MODERNO OPERANDO SISTEMAS

TERCERA EDICION

SOLUCIONES PROBLEMAS

ANDREW S. TANENBAUM
Vrije Universiteit
Amsterdam, Holanda

PRENTICE HALL
UPPER SADDLE RIVER, NJ 07458

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 1

1. La multiprogramación es el cambio rápido de la CPU entre múltiples procesos
esses en la memoria. Se usa comúnmente para mantener la CPU ocupada mientras uno o
más procesos están haciendo E / S.

2. La cola de entrada es la técnica de lectura en trabajos, por ejemplo, de tarjetas,
en el disco, de modo que cuando finalicen los procesos en ejecución,
habrá trabajo esperando a la CPU. El spooling de salida consta de primero
copiar archivos imprimibles en el disco antes de imprimirlos, en lugar de imprimir di-
directamente a medida que se genera la salida. La cola de entrada en una computadora personal no es
muy probablemente, pero la cola de salida sí lo es.

3. La razón principal de la multiprogramación es darle a la CPU algo que hacer
mientras espera que se complete la E / S. Si no hay DMA, la CPU está completamente ocupada.
haciendo E / S, por lo que no hay nada que ganar (al menos en términos de
zation) por multiprogramación. No importa cuántas E / S haga un programa, el
La CPU estará 100% ocupada. Esto, por supuesto, supone que el mayor retraso es la espera.
mientras se copian los datos. Una CPU podría hacer otro trabajo si las E / S fueran lentas para
otras razones (llegar en una línea serial, por ejemplo).

4. Todavía está vivo. Por ejemplo, Intel fabrica Pentium I, II y III y 4 CPU
con una variedad de propiedades diferentes, incluida la velocidad y el consumo de energía.
Todas estas máquinas son arquitectónicamente compatibles. Se diferencian solo en
precio y rendimiento, que es la esencia de la idea familiar.

5. Una pantalla de texto monocromática de 25 × 80 caracteres requiere un búfer de 2000 bytes. los
El mapa de bits de color de 24 bits de 1024 × 768 píxeles requiere 2,359,296 bytes. En 1980 estos
dos opciones habrían costado $ 10 y $ 11,520, respectivamente. Para corriente
precios, verifique cuánto RAM cuesta actualmente, probablemente menos de $ 1 / MB.

6. Considere la equidad y el tiempo real. La equidad requiere que cada proceso sea
asignó sus recursos de una manera justa, sin que ningún proceso obtenga más que lo justo
compartir. Por otro lado, el tiempo real requiere que los recursos se asignen según
sobre los momentos en que los diferentes procesos deben completar su ejecución. Un real
El proceso de tiempo puede obtener una parte desproporcionada de los recursos.

7. Las opciones (a), (c) y (d) deben restringirse al modo kernel.

8. Puede tomar 20, 25 o 30 mseg para completar la ejecución de estos programas.

dependiendo de cómo el sistema operativo los programe. Si *P0* y *P1* son programado en la misma CPU y *P2* está programado en la otra CPU, tomar 20 mses. Si *P0* y *P2* están programados en la misma CPU y *P1* está programado en la otra CPU, tomará 25 mseg. Si *P1* y *P2* están programados en la misma CPU y *P0* está programado en la otra CPU, tomará 30 mseg. Si los tres están en la misma CPU, tardará 35 ms.

2

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 1

9. Cada nanosegundo, una instrucción emerge de la tubería. Esto significa el la máquina está ejecutando mil millones de instrucciones por segundo. No importa en todas las etapas que tiene la tubería. Una tubería de 10 etapas con 1 nseg por stage también ejecutaría mil millones de instrucciones por segundo. Todo lo que importa es la frecuencia con la que una instrucción terminada aparece al final del proceso.

10. Tiempo medio de acceso =
0,95 × 2 nsec (la palabra es caché)
+ 0.05 × 0.99 × 10 nseg (la palabra está en la RAM, pero no en la caché)
+ 0.05 × 0.01 × 10,000,000 nseg (palabra en disco solamente)
= 5002,395 nseg
= 5,002395 μseg

11. El manuscrito contiene 80 × 50 × 700 = 2,8 millones de caracteres. Esto es, de Por supuesto, imposible encajar en los registros de cualquier CPU actualmente disponible y es demasiado grande para una caché de 1 MB, pero si tal hardware estuviera disponible, el El manuscrito se puede escanear en 2,8 ms de los registros o 5,8 ms de el caché. Hay aproximadamente 2700 bloques de datos de 1024 bytes, por lo que la extracción del disco requeriría unos 27 segundos, y de la cinta 2 minutos 7 segundos. Por supuesto, estos tiempos son solo para leer los datos. Procesamiento y reescribir los datos aumentaría el tiempo.

12. Quizás. Si la persona que llama recupera el control e inmediatamente sobrescribe los datos, cuando finalmente se produzca la escritura, se escribirán los datos incorrectos. Sin embargo, si el El controlador primero copia los datos en un búfer privado antes de regresar, luego la persona que llama se puede permitir que continúe inmediatamente. Otra posibilidad es permitir llamador para continuar y darle una señal cuando el búfer puede ser reutilizado, pero esto es complicado y propenso a errores.

13. Una instrucción trap cambia el modo de ejecución de una CPU del modo de usuario al modo kernel. Esta instrucción permite que un programa de usuario invoque funciones ciones en el kernel del sistema operativo.

14. Una trampa es causada por el programa y está sincronizada con él. Si el programa es ejecutar una y otra vez, la trampa siempre ocurrirá exactamente en la misma posición en el flujo de instrucciones. Una interrupción es causada por un evento externo y su la sincronización no es reproducible.

15. La tabla de procesos es necesaria para almacenar el estado de un proceso que está actualmente suspendido, listo o bloqueado. No es necesario en un solo sistema de proceso tem porque el proceso único nunca se suspende.

16. Al montar un sistema de archivos, los archivos ya están en el directorio del punto de montaje. inaccesible, por lo que los puntos de montaje normalmente están vacíos. Sin embargo, un administrador del sistema
El istrador podría querer copiar algunos de los archivos más importantes que normalmente se encuentran en el directorio montado al punto de montaje para que se puedan encontrar en su ruta normal en caso de emergencia cuando se estaba reparando el dispositivo montado.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 1

3

17. Una llamada al sistema permite que un proceso de usuario acceda y ejecute el sistema operativo. funciones dentro del kernel. Los programas de usuario utilizan llamadas al sistema para invocar operaciones
servicios del sistema de ing.

18. La horquilla puede fallar si no quedan ranuras libres en la mesa de proceso (y posiblemente si no queda memoria ni espacio de intercambio). Exec puede fallar si el nombre de archivo proporcionado no existe o no es un archivo ejecutable válido. La desvinculación puede fallar si el archivo se desvinculado no existe o el proceso de llamada no tiene la autoridad para desvincularlo.

19. Si la llamada falla, por ejemplo porque *fd* es incorrecta, puede devolver −1. Puede También fallan porque el disco está lleno y no es posible escribir el número de bytes solicitados. En una terminación correcta, siempre devuelve *nbytes* .

20. Contiene los bytes: 1, 5, 9, 2.

21. Hora de recuperar el archivo =
1 * 50 ms (tiempo para mover el brazo sobre la pista # 50)
+ 5 ms (tiempo para que el primer sector gire debajo de la cabeza)
+ 10/100 * 1000 ms (lectura 10 MB)
= 155 ms

22. Los archivos especiales de bloques consisten en bloques numerados, cada uno de los cuales se puede leer o
escrito independientemente de todos los demás. Es posible buscar a cualquier bloque y empieza a leer o escribir. Esto no es posible con archivos especiales de caracteres.

23. Las llamadas al sistema no tienen realmente nombres, salvo en el sentido de la documentación. Cuando el procedimiento de la biblioteca *lee* trampas en el kernel, pone el número de llamada al sistema en un registro o en la pila. Este número se utiliza para indexar en un mesa. Realmente no se usa ningún nombre en ninguna parte. Por otro lado, el nombre del procedimiento de la biblioteca es muy importante, ya que eso es lo que aparece en el programa.

24. Sí, especialmente si el kernel es un sistema de paso de mensajes.

25. En lo que respecta a la lógica del programa, no importa si una llamada a una El procedimiento de brary da como resultado una llamada al sistema. Pero si el rendimiento es un problema, si un
La tarea se puede realizar sin una llamada al sistema, el programa se ejecutará más rápido. Cada llamada al sistema implica un tiempo de sobrecarga para cambiar del contexto del usuario al contexto del kernel. Además, en un sistema multiusuario, el sistema operativo El tem puede programar otro proceso para que se ejecute cuando se complete una llamada al sistema, ralentizando aún más el progreso en tiempo real de un proceso de llamada.

26. Varias llamadas de UNIX no tienen contraparte en la API de Win32:
Enlace: un programa Win32 no puede hacer referencia a un archivo con un nombre alternativo o verlo en más de un directorio. Además, intentar crear un vínculo es conveniente forma de probar y crear un bloqueo en un archivo.

4

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 1

Montar y desmontar: un programa de Windows no puede hacer suposiciones sobre nombres de ruta estándar porque en sistemas con varias unidades de disco, la unidad parte del nombre de la ruta puede ser diferente.

Chmod: Windows usa listas de control de acceso

Matar: los programadores de Windows no pueden acabar con un programa que se comporta mal que no cooperando.

27. Cada arquitectura de sistema tiene su propio conjunto de instrucciones que puede ejecutar. Por tanto, un Pentium no puede ejecutar programas SPARC y un SPARC no puede ejecutar lindos programas Pentium. Además, las diferentes arquitecturas difieren en la arquitectura del bus. utilizado (como VME, ISA, PCI, MCA, SBus, ...) así como el tamaño de palabra del CPU (generalmente de 32 o 64 bits). Debido a estas diferencias en el hardware, no es factible construir un sistema operativo que sea completamente portátil. Muy El sistema operativo portátil constará de dos capas de alto nivel --- una máquina- capa dependiente y una capa independiente de la máquina. El dependiente de la máquina La capa aborda las especificidades del hardware y debe implementarse por separado. radamente para cada arquitectura. Esta capa proporciona una interfaz uniforme en la que se construye la capa independiente de la máquina. La capa independiente de la máquina debe implementarse solo una vez. Para ser altamente portátil, el tamaño de la máquina- La capa dependiente debe mantenerse lo más pequeña posible.

28. La separación de políticas y mecanismos permite a los diseñadores de sistemas operativos implementar pequeño número de primitivas básicas en el kernel. Estos primitivos son sim- plificados, porque no dependen de ninguna política específica. Ellos pueden entonces Ser utilizado para implementar mecanismos y políticas más complejos en el usuario. nivel.

29. Las conversiones son sencillas:

(a) Un micro año es $10^{-6} \times 365 \times 24 \times 3600 = 31.536$ segundos

(b) 1000 metros o 1 km.

(c) Hay 2^{40} bytes, que son 1.099.511.627.776 bytes.

(d) Es 6×10^{24} kg.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 2

1. Es concebible la transición de bloqueado a funcionamiento. Supongamos que un proceso se bloquea en E / S y la E / S finaliza. Si la CPU está inactiva, el proceso ess podría pasar directamente de bloqueado a en ejecución. La otra transición que falta, de listo a bloqueado, es imposible. Un proceso listo no puede hacer E / S ni otra cosa que podría bloquearlo. Solo un proceso en ejecución puede bloquear.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 2

5

2. Podría tener un registro que contenga un puntero a la tabla de proceso actual entrada. Cuando se completara la E / S, la CPU almacenaría el estado actual de la máquina en la entrada de la tabla de procesos actual. Luego iría al vector de interrupción para el interrumpir el dispositivo y buscar un puntero a otra entrada de la tabla de proceso (el procedimiento de vicio). A continuación, se iniciaría este proceso.

3. Generalmente, los lenguajes de alto nivel no permiten el tipo de acceso a la CPU. vajilla que se requiere. Por ejemplo, se puede requerir un manejador de interrupciones para habilitar y deshabilitar el servicio de interrupción de un dispositivo en particular, o para manipular datos dentro del área de pila de un proceso. Además, las rutinas del servicio de interrupción deben lindo lo más rápido posible.

4. Hay varias razones para usar una pila separada para el kernel. Dos de ellos son los siguientes. Primero, no desea que el sistema operativo se bloquee porque un programa de usuario mal escrito no permite suficiente espacio en la pila. En segundo lugar, si el kernel deja datos de pila en el espacio de memoria de un programa de usuario al regresar de una llamada al sistema, un usuario malintencionado podría utilizar estos datos

para obtener información sobre otros procesos.
5. Si cada trabajo tiene un 50% de espera de E / S, tardará 20 minutos en completarse en el ausencia de competencia. Si se ejecuta secuencialmente, el segundo terminará 40 minutos después de que comience el primero. Con dos trabajos, el uso aproximado de la CPU ción es 1 - 0,5 2 . Por lo tanto, cada uno obtiene 0.375 minutos de CPU por minuto de real hora. Para acumular 10 minutos de tiempo de CPU, un trabajo debe ejecutarse durante 10 / 0.375 minutos, o aproximadamente 26,67 minutos. Por lo tanto, ejecutando secuencialmente los trabajos terminan
a los 40 minutos, pero corriendo en paralelo terminan a los 26,67 minutos.
6. Sería difícil, si no imposible, mantener la coherencia del sistema de archivos. Cenar- plantean que un proceso cliente envía una solicitud al proceso 1 del servidor para actualizar un archivo. Este proceso actualiza la entrada de caché en su memoria. Poco después, otra er proceso cliente envía una solicitud al servidor 2 para leer ese archivo. Desafortunadamente, si el archivo también se almacena en caché allí, el servidor 2, en su inocencia, volverá obsoleto datos. Si el primer proceso escribe el archivo en el disco después de almacenarlo en caché, y el servidor 2 comprueba el disco en cada lectura para ver si su copia en caché está actualizada fecha, se puede hacer que el sistema funcione, pero son precisamente todos estos discos ac- ceses que el sistema de almacenamiento en caché está tratando de evitar.
7. No. Si un proceso de un solo subproceso está bloqueado en el teclado, no se puede bifurcar.
8. Un hilo de trabajo se bloqueará cuando tenga que leer una página web desde el disco. Si se están utilizando subprocesos a nivel de usuario, esta acción bloqueará todo el proceso, destruyendo el valor del multihilo. Por lo tanto, es esencial que los hilos del kernel se utilizan para permitir que algunos subprocesos se bloqueen sin afectar a los demás.
9. Sí. Si el servidor está completamente vinculado a la CPU, no es necesario tener varios hilos. Simplemente agrega una complejidad innecesaria. Como ejemplo, considere un tele- número de asistencia de directorio telefónico (como 555-1212) para un área con 1 millón

---

6
SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 2
personas. Si cada registro (nombre, número de teléfono) tiene, digamos, 64 caracteres, toda la base de datos ocupa 64 megabytes y se puede guardar fácilmente en el servidor memoria para proporcionar una búsqueda rápida.
10. Cuando se detiene un hilo, tiene valores en los registros. Deben ser salvados al igual que cuando se detiene el proceso, se deben guardar los registros. Multipro- Los hilos de programación no son diferentes a los procesos de multiprogramación, por lo que cada hilo necesita su propia área de guardado de registro.
11. Los hilos de un proceso cooperan. No son hostiles entre sí. Si rendimiento es necesario para el bien de la aplicación, entonces un hilo cederá. Después todos, normalmente es el mismo programador quien escribe el código para todos ellos.
12. Los subprocesos a nivel de usuario no pueden ser reemplazados por el reloj a menos que todo el proceso
El cuanto de ess se ha agotado. Los subprocesos de nivel de kernel se pueden adelantar individualmente dualmente. En el último caso, si un hilo se ejecuta demasiado, el reloj interrumpirá el proceso actual y, por lo tanto, el hilo actual. El kernel es libre de elegir una hilo diferente del mismo proceso para ejecutar a continuación si así lo desea.
13. En el caso de un solo subproceso, los aciertos de caché tardan 15 ms y los fallos de caché tardan 90 mseg. La media ponderada es de 2 / 3 x 15 + 1 / 3 x 90. Así, la re- media quest tarda 40 ms y el servidor puede hacer 25 por segundo. Para un multiproceso servidor, toda la espera del disco se superpone, por lo que cada solicitud toma 15 mseg, y el servidor puede manejar 66 2/3 solicitudes por segundo.
14. La mayor ventaja es la eficiencia. No se necesitan trampas en el kernel para

cambiar de hilo. La mayor desventaja es que si un hilo se bloquea, el
bloques de proceso de neumáticos.
15. Sí, se puede hacer. Después de cada llamada a *pthread create* , el programa principal
podría hacer un pthreadjoin para esperar hasta que el hilo recién creado haya salido antes
creando el siguiente hilo.
16. Los punteros son realmente necesarios porque el tamaño de la variable global es
desconocido. Podría ser cualquier cosa, desde un carácter hasta una matriz de puntos flotantes
números. Si se almacenara el valor, habría que dar el tamaño a
crear global , lo cual está bien, pero de qué tipo debería el segundo parámetro
de *set global* , y de qué tipo debería ser el valor de *read global* ?
17. Puede suceder que el sistema de ejecución esté precisamente en el punto de bloqueo o
desbloqueando un hilo y está ocupado manipulando las colas de programación. Esta
Sería un momento muy inoportuno para que el manejador de interrupciones del reloj comenzara
inspeccionar esas colas para ver si era el momento de hacer el cambio de hilo, ya que
podría estar en un estado inconsistente. Una solución es colocar una bandera cuando la ejecución
se ingresa el sistema de tiempo. El manejador del reloj vería esto y establecería su propia bandera,
luego regresa. Cuando finalizara el sistema de tiempo de ejecución, verificaría la bandera del reloj,
ver que se produjo una interrupción del reloj, y ahora ejecute el manejador del reloj.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 2
7
18. Sí, es posible, pero ineficaz. Un hilo que quiere hacer una llamada al sistema primero
establece un temporizador de alarma, luego realiza la llamada. Si la llamada se bloquea, el temporizador
vuelve
control al paquete de subprocesos. Por supuesto, la mayoría de las veces la llamada no
bloquear, y el temporizador debe borrarse. Por tanto, cada llamada al sistema que pueda bloquear
debe ejecutarse como tres llamadas al sistema. Si los temporizadores se apagan prematuramente, todos
se pueden desarrollar tipos de problemas. Esta no es una forma atractiva de construir un
paquete de hilos.
19. El problema de inversión de prioridad ocurre cuando un proceso de baja prioridad está en su
región crítica y de repente un proceso de alta prioridad está listo y es
programado. Si usa la espera ocupada, se ejecutará para siempre. Con nivel de usuario
subprocesos, no puede suceder que un subproceso de baja prioridad sea reemplazado repentinamente
Permitir la ejecución de un hilo de alta prioridad. No hay preferencia. Con nivel de kernel
hilos puede surgir este problema.
20. Con la programación por turnos funciona. Tarde o temprano *L* se ejecutará, y eventualmente
aliado saldrá de su región crítica. El punto es, con la programación prioritaria, *L*
nunca llega a correr en absoluto; con round robin, obtiene un intervalo de tiempo normal
cally, por lo que tiene la oportunidad de salir de su región crítica.
21. Cada subproceso llama a los procedimientos por sí solo, por lo que debe tener su propia pila para el
variables locales, direcciones de retorno, etc. Esto es igualmente cierto para el nivel de usuario
subprocesos como para los subprocesos a nivel de kernel.
22. Sí. La computadora simulada podría ser multiprogramada. Por ejemplo,
mientras el proceso *A se* está ejecutando, lee alguna variable compartida. Entonces un simulacro
El tic del reloj se produce y se ejecuta el proceso *B.* También lee la misma variedad
poder. Luego agrega 1 a la variable. Cuando se ejecuta el proceso *A* , si también agrega uno
a la variable, tenemos una condición de carrera.
23. Sí, todavía funciona, pero todavía está ocupado esperando, por supuesto.
24. Ciertamente funciona con programación preventiva. De hecho, fue diseñado para
Ese caso. Cuando la programación no es preventiva, puede fallar. Considere el caso
en cuyo *turno* es inicialmente 0 pero el proceso 1 se ejecuta primero. Se repetirá para siempre

y nunca suelte la CPU.
25. Para hacer una operación de semáforo, el sistema operativo primero deshabilita las interrupciones. Luego lee el valor del semáforo. Si está haciendo una bajada y el sema-
phore es igual a cero, coloca el proceso de llamada en una lista de procesos bloqueados asociado con el semáforo. Si está subiendo , debe verificar si hay alguna los procesos están bloqueados en el semáforo. Si uno o más procesos están bloqueados ed, uno de ellos se elimina de la lista de procesos bloqueados y se ejecuta nable. Cuando se hayan completado todas estas operaciones, las interrupciones pueden habilitado de nuevo.

8
SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 2
26. Asociados con cada semáforo de conteo hay dos semáforos binarios, *M* , utilizado para la exclusión mutua, y *B* , utilizado para el bloqueo. También asociado con cada semáforo de conteo es un contador que contiene el número de up s menos el número de down s, y una lista de procesos bloqueados en ese semáforo. Para im-
plemento hacia abajo , un proceso primero obtiene acceso exclusivo a los semáforos, mostrador, y la lista haciendo una abajo en *M* . Luego disminuye el contador. Si se es cero o más, solo hace un up en *M* y sale. Si *M* es negativo, el proceso ess se coloca en la lista de procesos bloqueados. Luego se hace un up en *M* y un abajo se realiza en *B* para bloquear el proceso. Para implementar arriba , primero *M* está abajo ed para obtener la exclusión mutua, y luego se incrementa el contador. Si es mas que cero, nadie estaba bloqueado, así que todo lo que hay que hacer es hasta *M* . Si, sin embargo, el contador ahora es negativo o cero, se debe eliminar algún proceso de la lista. Finalmente, se realiza una subida en *B* y *M* en ese orden.
27. Si el programa opera en fases y ninguno de los procesos puede ingresar al siguiente fase hasta que ambos hayan terminado con la fase actual, tiene perfecto sentido para utilizar una barrera.
28. Con los hilos del kernel, un hilo puede bloquearse en un semáforo y el kernel puede ejecutar algún otro hilo en el mismo proceso. En consecuencia, no hay problema utilizando semáforos. Con subprocesos a nivel de usuario, cuando un subproceso se bloquea en un semáforo, el kernel cree que todo el proceso está bloqueado y no lo ejecuta Nunca más. En consecuencia, el proceso falla.
29. Es muy costoso de implementar. Cada vez que cualquier variable que aparezca en un predicado en el que algún proceso está esperando cambios, el sistema de tiempo de ejecución debe reevaluar el predicado para ver si el proceso se puede desbloquear. Con los monitores Hoare y Brinch Hansen, los procesos solo se pueden despertar en un primitiva de señal .
30. Los empleados se comunican pasando mensajes: pedidos, comida y bolsas en este caso. En términos de UNIX , los cuatro procesos están conectados por tuberías.
31. No conduce a condiciones de carrera (nunca se pierde nada), pero es eficaz ocupado esperando.
32. Tardará *nT* seg.
33. En casos simples, puede ser posible determinar si la E / S limitará mirando el código fuente. Por ejemplo, un programa que lee todos sus archivos de entrada. en búferes al principio probablemente no estará vinculado a E / S, pero es un problema que lee y escribe de forma incremental en varios archivos diferentes (como una ler) es probable que esté vinculado a E / S. Si el sistema operativo proporciona una facilidad como como el comando *ps de* UNIX que puede indicarle la cantidad de tiempo de CPU utilizado por un programa, puede comparar esto con el tiempo total para completar la ejecución de el programa. Esto es, por supuesto, más significativo en un sistema en el que está

el único usuario.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 2

9

34. Para varios procesos en una canalización, el padre común podría pasar a la borrar la información del sistema sobre el flujo de datos. Con esta información el El sistema operativo podría, por ejemplo, determinar qué proceso podría suministrar salida a un proceso de bloqueo en una llamada de entrada.

35. La eficiencia de la CPU es el tiempo útil de la CPU dividido por el tiempo total de la CPU. Cuando $Q \geq T$, el ciclo básico es que el proceso se ejecute para T y se someta a una interruptor de proceso para S. Por tanto (a) y (b) tienen una eficiencia de $T / (S + T)$. Cuando el cuanto es más corto que T, cada ejecución de T requerirá el proceso T / Q interruptores, desperdiciando un tiempo de ST / Q. La eficiencia aquí es entonces

$$\frac{T + ST / Q}{T}$$

que se reduce a $Q / (Q + S)$, que es la respuesta a (c). Para (d), simplemente sub- sustituya Q por S y encuentre que la eficiencia es del 50%. Finalmente, para (e), cuando $Q \rightarrow 0$ la eficiencia pasa a 0.

36. El trabajo más corto primero es la forma de minimizar el tiempo medio de respuesta.

$0 < X \leq 3$: X, 3, 5, 6, 9.
$3 < X \leq 5$: 3, X, 5, 6, 9.
$5 < X \leq 6$: 3, 5, X, 6, 9.
$6 < X \leq 9$: 3, 5, 6, X, 9.
$X > 9$: 3, 5, 6, 9, X.

37. Para round robin, durante los primeros 10 minutos cada trabajo obtiene 1/5 de la CPU. A al final de los 10 minutos, C termina. Durante los siguientes 8 minutos, cada trabajo obtiene 1/4 de la CPU, después de lo cual D acabados. Entonces cada uno de los tres restantes Jobs obtiene 1/3 de la CPU durante 6 minutos, hasta que B termina, y así sucesivamente. La aleta- Los tiempos de cambio para los cinco trabajos son 10, 18, 24, 28 y 30, para un promedio de 22 minutos. Para la programación prioritaria, B se ejecuta primero. Después de 6 minutos está terminado. Los otros trabajos terminan a los 14, 24, 26 y 30, durante un promedio de 18,8 minutos. Si los trabajos se ejecutan en el orden A a E, terminan en 10, 16, 18, 22 y 30, por un promedio de 19,2 minutos. Finalmente, el trabajo más corto primero produce tiempos de acabado de 2, 6, 12, 20 y 30, durante un promedio de 14 minutos.

38. La primera vez que obtiene 1 cuanto. En ejecuciones sucesivas obtiene 2, 4, 8 y 15, por lo que debe intercambiarse 5 veces.

39. Se podría hacer una verificación para ver si el programa esperaba información y no cualquier cosa con él. Un programa que no esperaba entrada y no procesó no obtendría ningún impulso de prioridad especial.

40. La secuencia de predicciones es 40, 30, 35 y ahora 25.

10

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 2

41. La fracción de CPU utilizada es 35/50 + 20/100 + 10/200 + x / 250. Ser - estar programable, debe ser menor que 1. Por lo tanto, x debe ser menor que 12,5 mseg.

42. La programación de dos niveles es necesaria cuando la memoria es demasiado pequeña para contener todos los Procesos listos. Algunos de ellos se guardan en la memoria y se hace una elección. de ese conjunto. De vez en cuando, se ajusta el conjunto de procesos internos. Esta El algoritmo es fácil de implementar y razonablemente eficiente, ciertamente mucho mejor

que, digamos, round robin sin tener en cuenta si un proceso estaba en la memoria o no.

43. Cada llamada de voz se ejecuta 200 veces por segundo y usa hasta 1 mseg por ráfaga, por lo que cada la llamada de voz necesita 200 mseg por segundo o 400 mseg para los dos. los el video se ejecuta 25 veces por segundo y usa hasta 20 milisegundos cada vez, para un total de 500 milisegundos por segundo. Juntos consumen 900 ms por segundo, por lo que queda tiempo y el sistema es programable.

44. El kernel puede programar procesos por cualquier medio que desee, pero dentro de cada proceso ejecuta subprocesos estrictamente en orden de prioridad. Dejando que el proceso del usuario establezca la prioridad de sus propios hilos, el usuario controla la política pero el kernel maneja el mecanismo.

45. El cambio significaría que después de que un filósofo dejara de comer, ninguno de sus los vecinos podrían ser elegidos a continuación. De hecho, nunca serían elegidos. Cenar- plantean que el filósofo 2 terminó de comer. Haría *pruebas* para los filósofos 1 y 3, y ninguno se iniciaría, a pesar de que ambos tenían hambre y ambos tenedores disponibles. Del mismo modo, si el filósofo 4 terminó de comer, el filósofo 3 no se iniciaría. Nada lo pondría en marcha.

46. Si un filósofo bloquea, los vecinos pueden ver más tarde que tiene hambre al comprobar su estado, en *prueba* , para que pueda ser despertado cuando las horquillas estén disponibles.

47. Variación 1: los lectores tienen prioridad. Ningún escritor puede empezar cuando un lector está actuando. tivo. Cuando aparece un nuevo lector, puede comenzar inmediatamente a menos que un escritor sea actualmente activo. Cuando un escritor termina, si los lectores esperan, todos están comenzó, independientemente de la presencia de escritores en espera. Variación 2: Escritores tienen prioridad. Ningún lector puede comenzar cuando un escritor está esperando. Cuando el último ac- finaliza el proceso productivo, se inicia un escritor, si lo hay; de lo contrario, todos los Se inician los lectores (si los hay). Variación 3: versión simétrica. Cuando un lector es activos, los nuevos lectores pueden comenzar inmediatamente. Cuando un escritor termina, un nuevo El escritor tiene prioridad, si uno está esperando. En otras palabras, una vez que hemos comenzado leyendo, seguimos leyendo hasta que no queden lectores. Del mismo modo, una vez que han comenzado a escribir, todos los escritores pendientes pueden ejecutarse.

48. Un posible script de shell podría ser

```
Si [ ! –F números]; luego echo 0> números; fi
cuenta = 0
```

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 2

11

```
while (prueba $ count! = 200)
hacer
count = 'expr $ count + 1'
n = 'cola –1 números'
expr $ n + 1 >> números
hecho
```

Ejecute el script dos veces simultáneamente, iniciándolo una vez en segundo plano (usando &) y nuevamente en primer plano. Luego examine los *números de* archivo . Va a probablemente empiece pareciendo una lista ordenada de números, pero en algún momento perderá su orden, debido a la condición de carrera creada al ejecutar dos ies del guión. La carrera se puede evitar teniendo cada copia del guión.

Pruebe y establezca un bloqueo en el archivo antes de ingresar al área crítica, y desbloquee al salir del área crítica. Esto se puede hacer así:

si en los números numbers.lock
luego
n = 'cola –1 números'
expr $ n + 1 >> números
rm numeros.lock
fi
Esta versión simplemente saltará un turno cuando el archivo sea inaccesible, solución variante
pueden poner el proceso en reposo, esperar ocupados o contar solo bucles en
que la operación es exitosa.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 3

1. Es un accidente. El registro base es 16,384 porque el programa pasó a
cargarse en la dirección 16,384. Podría haberse cargado en cualquier lugar. El límite
el registro es 16,384 porque el programa contiene 16,384 bytes. Podría haber
sido cualquier longitud. Que la dirección de carga coincide exactamente con el programa
la longitud es pura coincidencia.

2. Se debe copiar casi toda la memoria, lo que requiere que cada palabra sea
leer y luego reescribir en una ubicación diferente. La lectura de 4 bytes tarda 10 nseg,
por lo que leer 1 byte tarda 2,5 nseg y escribir otros 2,5 nseg, por un
total de 5 nseg por byte compactado. Esta es una velocidad de 200.000.000 bytes / seg.
Para copiar 128 MB (2^{27} bytes, que es aproximadamente $1{,}34 \times 10^8$ bytes), la computadora
necesita 2^{27} / 200,000,000 seg, que es aproximadamente 671 ms. Este número es ligeramente
pesimista porque si el agujero inicial en la parte inferior de la memoria es de *k* bytes,
esos *k* bytes no necesitan copiarse. Sin embargo, si hay muchos agujeros y

muchos segmentos de datos, los agujeros serán pequeños, por lo que *k* será pequeño y el error
en el cálculo también será pequeño.

3. El mapa de bits necesita 1 bit por unidad de asignación. Con $2^{27}/n$ unidades de asignación, esto es
$2^{24}/n$ bytes. La lista enlazada tiene $2^{27}/2^{16}$ o 2^{11} nodos, cada uno de 8 bytes, para una
total de 2^{14} bytes. Para *n* pequeña , la lista enlazada es mejor. Para *n* grande , el mapa de bits
es mejor. El punto de cruce se puede calcular equiparando estas dos fórmulas.
las y resolviendo para *n* . El resultado es 1 KB. Para *n* menor de 1 KB, un enlace
la lista es mejor. Para *n* mayor de 1 KB, es mejor un mapa de bits. Por supuesto, el
la suposición de segmentos y agujeros que se alternan cada 64 KB es muy poco realista.
Además, necesitamos *n* <= 64 KB si los segmentos y agujeros son 64 KB.

4. El primer ajuste ocupa 20 KB, 10 KB, 18 KB. El mejor ajuste requiere 12 KB, 10 KB y 9 KB.
El peor ajuste ocupa 20 KB, 18 KB y 15 KB. El siguiente ajuste ocupa 20 KB, 18 KB y 9
KB.

5. Para un tamaño de página de 4 KB, los pares (página, desplazamiento) son (4, 3616), (8, 0) y (14,
2656). Para un tamaño de página de 8 KB, son (2, 3616), (4, 0) y (7, 2656).

6. Construyeron una MMU y la insertaron entre el 8086 y el bus. Así todo
8086 direcciones físicas entraron en la MMU como direcciones virtuales. La MMU
luego los asignó a direcciones físicas, que fueron al autobús.

7.

(a) *M* tiene que ser de al menos 4096 para garantizar un error de TLB por cada acceso a un ele-
de X. Dado que *N* solo afecta la cantidad de veces que se accede a *X* , cualquier valor
de *N* servirá.

(b) *M* aún debe ser al menos 4.096 para garantizar un error de TLB para cada acceso a un
elemento de *X* . Pero ahora *N* debería ser mayor que 64K para superar el TLB,
es decir, *X* debe superar los 256 KB.

8. El espacio total de direcciones virtuales para todos los procesos combinados es *nv* , por lo que

se necesita mucho espacio de almacenamiento para las páginas. Sin embargo, una cantidad *r* puede estar en la RAM, por lo que
la cantidad de almacenamiento en disco requerida es solo *nv* - *r* . Esta cantidad es mucho más de lo que nunca se necesita en la práctica porque rara vez habrá *n* procesos activos en funcionamiento y, aún más raramente, todos ellos necesitarán el máximo memoria virtual bajada.

9. La tabla de página contiene $2^{32}/2^{13}$ entradas, lo cual es 524.288. Cargando la página la tabla tarda 52 mseg. Si un proceso obtiene 100 mseg, esto consta de 52 mseg para cargando la tabla de páginas y 48 ms para ejecutar. Así, el 52% del tiempo se dedica cargando tablas de páginas.

10.

(a) Necesitamos una entrada para cada página, o $2^{24} = 16 \times 1024 \times 1024$ entradas, ya que hay 36 = 48 - 12 bits en el campo de número de página.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 3

13

(b) Las direcciones de instrucción llegarán al 100% en el TLB. Las páginas de datos tendrán un Tasa de aciertos de 100 hasta que el programa haya pasado a la siguiente página de datos. Desde un La página de 4 KB contiene 1024 enteros largos, habrá un error de TLB y un acceso adicional a la memoria por cada 1.024 referencias de datos.

11.

(a) Una tabla de páginas de varios niveles reduce el número de páginas reales de la página. tabla que necesita estar en la memoria debido a su estructura jerárquica. De hecho, en un programa con mucha instrucción y localidad de datos, solo necesitamos el tabla de páginas de nivel superior (una página), una página de instrucciones y una página de datos.

(b) Asigne 12 bits para cada uno de los tres campos de página. El campo de compensación requiere 14 bits para direccionar 16 KB. Eso deja 24 bits para los campos de la página. Ya que cada entrada tiene 4 bytes, una página puede contener 2 entradas de tabla de 2^{12} páginas y, por lo tanto, fore requiere 12 bits para indexar una página. Asignar 12 bits para cada uno de los Los campos de página abordarán los 2^{38} bytes.

12. Se utilizan veinte bits para los números de página virtual, dejando 12 para los conjunto. Esto produce una página de 4 KB. Veinte bits para la página virtual implica 2^{20} páginas.

13. El número de páginas depende del número total de bits en *una* , *b,* y *c* combined. No importa cómo se dividan entre los campos.

14. Para una tabla de páginas de un nivel, hay $2^{32}/2^{12}$ o páginas 1M necesario. Por lo tanto, la La tabla de páginas debe tener 1 millón de entradas. Para la paginación de dos niveles, la tabla de la página principal
tiene 1K entradas, cada una de las cuales apunta a una tabla de la segunda página. Solo dos de estos son usados. Por lo tanto, en total solo se necesitan tres entradas en la tabla de páginas, una en el tabla de nivel superior y una en cada una de las tablas de nivel inferior.

15. El tiempo de instrucción efectivo es $1 h + 5 (1 - h)$, donde *h* es la tasa de aciertos. Si nosotros igualar esta fórmula con 2 y resolver para *h* , encontramos que *h* debe ser al menos 0,75.

16. El bit *R* nunca se necesita en el TLB. La mera presencia de una página ahí significa que se ha hecho referencia a la página; de lo contrario, no estaría allí. Por lo tanto, la bit es completamente redundante. Cuando la entrada se vuelve a escribir en la memoria, cómo-siempre, se establece el bit *R* en la tabla de páginas de memoria.

17. Una memoria asociativa compara esencialmente una clave con el contenido de múltiples se registra simultáneamente. Para cada registro debe haber un conjunto de comparadores que comparan cada bit en el contenido del registro con la clave que se busca.

El número de puertas (o transistores) necesarios para implementar tal dispositivo es un
función lineal del número de registros, por lo que expandir el diseño se
pensativo linealmente.

14

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 3

18. Con páginas de 8 KB y un espacio de direcciones virtuales de 48 bits, la cantidad de páginas es $2^{48}/2^{13}$, que es 2^{35} (alrededor de 34 mil millones).

19. La memoria principal tiene $2^{28}/2^{13} = 32.768$ páginas. Una tabla hash de 32K tendrá un longitud media de la cadena de 1. Para bajar de 1, tenemos que pasar al siguiente tamaño, 65.536 entradas. Distribuir 32,768 entradas en 65,536 espacios de mesa dará una longitud media de la cadena de 0,5, lo que garantiza una búsqueda rápida.

20. Esto probablemente no sea posible excepto en el caso inusual y poco útil de un programa cuyo curso de ejecución es completamente predecible en la compilación tiempo de la estación. Si un compilador recopila información sobre las ubicaciones en el código de las llamadas a los procedimientos, esta información se puede utilizar en el momento del enlace para reorganizar el código objeto de modo que los procedimientos se ubiquen cerca del código que llama ellos. Esto haría más probable que un procedimiento estuviera en el mismo page como el código de llamada. Por supuesto, esto no ayudaría mucho para los procedimientos. llamado desde muchos lugares del programa.

21.

(a) Cada referencia tendrá un error de página a menos que el número de marcos de página sea 512, la longitud de toda la secuencia.

(b) Si hay 500 marcos, asigne las páginas 0 a 498 a marcos fijos y varíe solo un cuadro.

22. Los marcos de página para FIFO son los siguientes:

x0172333300
xx017222233
xxx01777722
xxxx0111177

Los marcos de página para LRU son los siguientes:

x0172327103
xx017232710
xxx01773271
xxxx0111327

FIFO produce errores de seis páginas; LRU rinde siete.

23. La primera página con un bit 0 se elegirá, en este caso *D* .

24. Los contadores son

Página 0: 0110110
Página 1: 01001001
Página 2: 00110111
Página 3: 10001011

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 3

15

25. La secuencia: 0, 1, 2, 1, 2, 0, 3. En LRU, la página 1 será reemplazada por la página 3. En el reloj, la página 1 será reemplazada, ya que todas las páginas estarán marcadas y la actual sor está en la página 0.

26. La edad de la página es 2204 - 1213 = 991. Si $\tau = 400$, definitivamente está fuera de

el conjunto de trabajo y no se hizo referencia recientemente, por lo que será desalojado. los
$\tau = 1000$ La situación es diferente. Ahora la página entra dentro del conjunto de trabajo
(apenas), por lo que no se quita.
27. La latencia de búsqueda más rotación es de 20 mseg. Para páginas de 2 KB, el tiempo de transferencia
es 1,25 mseg, para un total de 21,25 mseg. La carga de 32 de estas páginas llevará
680 mseg. Para páginas de 4 KB, el tiempo de transferencia se duplica a 2,5 ms, por lo que
el tiempo total por página es de 22,50 mseg. La carga de 16 de estas páginas tarda 360 ms.
28. NRU elimina la página 2. FIFO elimina la página 3. LRU elimina la página 1. Segundo
chance quita la página 2.
29. Fragmento *B* ya que el código tiene localidad más espacial que Fragmento *A* . los
El bucle interno causa solo una falla de página por cada otra iteración del externo
lazo. (Solo habrá 32 fallas de página). Aparte (Fragmento *A*): Dado que un marco
es 128 palabras, una fila de la matriz *X* ocupa la mitad de una página (es decir, 64 palabras).
Los ajustes de matriz completa en $64 \times 32 / 128 = 16$ marcos. El bucle interno del código
recorre filas consecutivas de *X* para una columna determinada. Así, todos los demás
la referencia a $X[i][j]$ provocará un error de página. El número total de errores de página
será $64 \times 64 / 2 = 2{,}048$.
30. El tambor de paginación PDP-1 tenía la ventaja de no tener latencia rotacional. Esta
guardaba media rotación cada vez que se escribía memoria en el tambor.
31. El texto tiene ocho páginas, los datos son cinco páginas y la pila es de cuatro páginas.
El programa no encaja porque necesita 17 páginas de 4096 bytes. Con un 512-
página de bytes, la situación es diferente. Aquí el texto es 64 páginas, los datos son 33
páginas, y la pila es de 31 páginas, para un total de 128 páginas de 512 bytes, que cabe.
Con el tamaño de página pequeño está bien, pero no con el grande.
32. Si las páginas se pueden compartir, sí. Por ejemplo, si dos usuarios de un sistema de tiempo compartido
están ejecutando el mismo editor al mismo tiempo y el texto del programa se comparte
en lugar de copiarse, algunas de esas páginas pueden estar en el conjunto de trabajo de cada usuario en
al mismo tiempo.
33. El programa está obteniendo 15.000 fallos de página, cada uno de los cuales utiliza 2 ms de
Tiempo de procesamiento. Juntos, la sobrecarga de fallas de página es de 30 segundos. Esto significa que
de los 60 segundos utilizados, la mitad se dedicó a la sobrecarga de fallas de página y la otra mitad a la ejecución
el programa. Si ejecutamos el programa con el doble de memoria, obtenemos la mitad
tantos fallos de página de memoria y solo 15 segundos de sobrecarga de fallo de página, por lo que
el tiempo total de ejecución será de 45 segundos.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 3
34. Funciona para el programa si el programa no se puede modificar. Funciona para el
datos si los datos no se pueden modificar. Sin embargo, es común que el programa
no se puede modificar y es extremadamente raro que los datos no se puedan modificar. Si
el área de datos en el archivo binario se sobrescribió con páginas actualizadas, el siguiente
vez que se inició el programa, no tendría los datos originales.
35. La instrucción podría situarse a horcajadas en el límite de una página, provocando dos errores de página
para buscar la instrucción. La palabra recuperada también podría abarcar el límite de una página,
generando dos fallas más, para un total de cuatro. Si las palabras deben estar alineadas en
memoria, la palabra de datos puede causar sólo un fallo, pero una instrucción para cargar un

La palabra de 32 bits en la dirección 4094 en una máquina con una página de 4 KB es legal en algunos máquinas (incluido el Pentium).
36. No. La tecla de búsqueda utiliza tanto el número de segmento como el número de página virtual. ber, por lo que la página exacta se puede encontrar en una sola coincidencia.
37. Estos son los resultados:

Habla a	**¿Culpa?**
(una)	
(14, 3)	
No (o 0xD3 o 1110 0011)	
(si)	
N / A	Fallo de protección: escribir para leer / ejecutar segmento
(C)	
N / A	Fallo de página
(re)	
N / A	Fallo de protección: saltar al segmento de lectura / escritura

38. No se necesita soporte de memoria virtual general cuando los requisitos de memoria de todas las aplicaciones son bien conocidas y controladas. Algunos ejemplos son procesadores de propósito cial (por ejemplo, procesadores de red), procesadores integrados y supercomputadoras (por ejemplo, diseño de alas de avión). En estas situaciones, deberíamos Considere siempre la posibilidad de utilizar más memoria real. Si el operativo El sistema no tenía que admitir memoria virtual, el código sería mucho más pler y más pequeño. Por otro lado, algunas ideas de la memoria virtual pueden todavía ser explotados de forma rentable, aunque con diferentes requisitos de diseño. por ejemplo, el aislamiento del programa / subproceso podría estar pagándose en la memoria flash

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 4

1. Estos sistemas cargaron el programa directamente en la memoria y comenzaron a ejecutarse en palabra 0, que era el número mágico. Para evitar intentar ejecutar el encabezado como código, el número mágico era una instrucción BRANCH con una dirección de destino justo encima del encabezado. De esta forma fue posible leer el archivo binario di- directamente en el espacio de direcciones del nuevo proceso y ejecutarlo en 0, sin siquiera saberlo ver qué tan grande era el encabezado.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 4

17

2. El sistema operativo se preocupa por la longitud del registro cuando los archivos se pueden estructurar como registros con claves en una posición específica dentro de cada registro y es posible pedir un registro con una clave determinada. En ese caso, el sistema debe saber cómo grandes son los registros para que pueda buscar la clave en cada uno.
3. Para empezar, si no hubiera abierto , en cada lectura sería necesario especifique el nombre del archivo que se abrirá. Entonces el sistema tendría que buscar el i-nodo, aunque podría almacenarse en caché. Un problema que rápidamente surge es cuándo vaciar el i-nodo de nuevo al disco. Sin embargo, podría agotarse. Eso sería un poco torpe, pero podría funcionar.
4. No. Si desea volver a leer el archivo, acceda aleatoriamente al byte 0.
5. Sí. La llamada de cambio de nombre no cambia la hora de creación ni la hora de la última modificación, pero la creación de un nuevo archivo hace que obtenga la hora actual la hora de creación y la hora de la última modificación. Además, si el disco está lleno, el la copia puede fallar.
6. La parte mapeada del archivo debe comenzar en un límite de página y ser un número integral de páginas de extensión. Cada página asignada utiliza el archivo en sí como

almacén de respaldo. La memoria no asignada utiliza un archivo temporal o una partición como respaldo Tienda.
7. Utilice nombres de archivo como */ usr / ast / file* . Si bien parece una ruta jerárquica
name, en realidad es solo un nombre que contiene barras inclinadas.
8. Una forma es agregar un parámetro adicional a la llamada al sistema de lectura que indica qué dirección para leer. En efecto, cada lectura tiene el potencial de hacer un
buscar dentro del archivo. Las desventajas de este esquema son (1) un parámetro extra
eter en cada llamada de lectura , y (2) exigir al usuario que realice un seguimiento de dónde
el puntero de archivo es.
9. El componente dotdot mueve la búsqueda a */ usr* , por lo que *../ast la* coloca en */ usr / ast* .
Por tanto, *../ast/x* es lo mismo que */ usr / ast / x* .
10. Dado que el almacenamiento desperdiciado se encuentra *entre* las unidades de asignación (archivos), no dentro
ellos, esto es fragmentación externa. Es precisamente análogo al externo
fragmentación de la memoria principal que ocurre con un sistema de intercambio o un sistema
tem utilizando segmentación pura.
11. Se necesitan 9 ms para iniciar la transferencia. Para leer 2^{13} bytes a una velocidad de transferencia de 2^{23}
bytes / seg requiere 2^{-10}
s (977 mseg), para un total de 9,977 mseg. Escribiéndolo
la vuelta tarda otros 9,977 mseg. Por lo tanto, copiar un archivo tarda 19,954 ms. A
la mitad compacta de un disco de 16 GB implicaría copiar 8 GB de almacenamiento, que
son 2^{20} archivos. A 19,954 ms por archivo, esto tarda 20,923 segundos, que son 5,8 horas.
Claramente, compactar el disco después de cada eliminación de archivos no es una gran idea.

18
SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 4
12. Si se hace bien, sí. Durante la compactación, cada archivo debe organizarse de modo que todos de sus bloques son consecutivos, para un acceso rápido. Windows tiene un programa que
desfragmenta y reorganiza el disco. Se anima a los usuarios a ejecutarlo periódicamente.
cally para mejorar el rendimiento del sistema. Pero dado el tiempo que lleva correr
una vez al mes puede ser una buena frecuencia.
13. Una cámara fotográfica digital graba varias fotografías en secuencia en un
medio de almacenamiento no volátil (por ejemplo, memoria flash). Cuando se reinicia la cámara,
el medio se vacía. A partir de entonces, las imágenes se graban de una en una en
secuencia hasta que el medio esté lleno, momento en el que se cargan en un
disco. Para esta aplicación, un sistema de archivos contiguo dentro de la cámara (por ejemplo, en
el medio de almacenamiento de imágenes) es ideal.
14. Encuentra la dirección del primer bloque en la entrada del directorio. Luego sigue el
cadena de punteros de bloque en el FAT hasta que haya localizado el bloque que necesita. Eso
luego recuerda este número de bloque para la siguiente llamada al sistema de lectura .
15. El bloque indirecto puede contener 256 direcciones de disco. Junto con los 10 directos
direcciones de disco, el archivo máximo tiene 266 bloques. Dado que cada bloque tiene 1 KB,
el archivo más grande es de 266 KB.
16. Debe haber una forma de indicar que los punteros del bloque de direcciones contienen datos, en lugar de
que punteros. Si queda un poco en algún lugar entre los atributos,
puede ser usado. Esto deja los nueve indicadores para los datos. Si los punteros son k bytes
cada uno, el archivo almacenado puede tener hasta 9 k bytes de longitud. Si no queda nada entre
los atributos, la primera dirección de disco puede contener una dirección no válida para marcar el
siguientes bytes como datos en lugar de punteros. En ese caso, el archivo máximo es

8 *k* bytes.
17. Elinor tiene razón. Tener dos copias del i-nodo en la tabla al mismo tiempo
es un desastre, a menos que ambos sean de solo lectura. El peor de los casos es cuando ambos están siendo
actualizado simultáneamente. Cuando los i-nodos se vuelven a escribir en el disco,
el que se escriba al final borrará los cambios realizados por el otro,
y los bloques de disco se perderán.
18. Los enlaces físicos no requieren espacio adicional en disco, solo un contador en el i-nodo para
Mantenga un registro de cuántos hay. Los enlaces simbólicos necesitan espacio para almacenar
nombre del archivo al que apunta. Los enlaces simbólicos pueden apuntar a archivos en otros ma-
chines, incluso a través de Internet. Los enlaces duros están restringidos a apuntar a archivos
dentro de su propia partición.
19. El mapa de bits requiere B bits. La lista libre requiere bits DF . La lista gratuita
requiere menos bits si $DF < B$. Alternativamente, la lista gratuita es más corta si
$F / B < 1 / D$, donde F / B es la fracción de bloques libres. Para el disco de 16 bits
vestidos, la lista libre es más corta si el 6% o menos del disco está libre.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 4
19
20. El comienzo del mapa de bits se ve así:
(a) Después de escribir el archivo B : 1111 1111 1111 0000
(b) Después de eliminar el archivo A : 1000 0001 1111 0000
(c) Después de escribir el archivo C : 1111 1111 1111 1100
(d) Después de eliminar el archivo B : 1111 1110 0000 1100
21. No es un problema grave en absoluto. La reparación es sencilla; solo lleva tiempo.
El algoritmo de recuperación es hacer una lista de todos los bloques en todos los archivos y
tome el complemento como la nueva lista gratuita. En UNIX, esto se puede hacer escaneando
uniendo todos los i-nodos. En el sistema de archivos FAT, el problema no puede ocurrir entre
porque no hay una lista libre. Pero incluso si lo hubiera, todo eso tendría que ser
hacer para recuperarlo es escanear el FAT en busca de entradas libres.
22. Es posible que la tesis de Ollie no esté respaldada con tanta fiabilidad como él quisiera. Una copia de seguridad
El programa puede pasar sobre un archivo que está actualmente abierto para escritura, ya que el estado de
los datos de dicho archivo pueden ser indeterminados.
23. Deben realizar un seguimiento de la hora del último volcado en un archivo en el disco. En cada
dump, se adjunta una entrada a este archivo. En el momento del volcado, se lee el archivo y
hora de la última entrada anotada. Cualquier archivo cambiado desde ese momento se vuelca.
24. En (a) y (b), 21 no estaría marcado. En (c), no habría ningún cambio. En
(d), no se marcaría 21.
25. Muchos archivos UNIX son cortos. Si todo el archivo cabe en el mismo bloque que el i-
nodo, solo se necesitaría un acceso al disco para leer el archivo, en lugar de dos, como
es actualmente el caso. Incluso para archivos más largos habría una ganancia, ya que uno
se necesitarían menos accesos al disco.
26. No debería suceder, pero debido a un error en algún lugar podría suceder. Significa
que algún bloque ocurre en dos archivos y también dos veces en la lista libre. El primero
El paso para reparar el error es eliminar ambas copias de la lista gratuita. Siguiente a
Se debe adquirir el bloque libre y copiar allí el contenido del bloque enfermo.
Finalmente, la ocurrencia del bloque en uno de los archivos debe cambiarse a
consulte la copia recién adquirida del bloque. En este punto, el sistema es una vez
de nuevo consistente.

27. El tiempo necesario es $h + 40 \times (1 - h)$. La trama es solo una línea recta.
28. A 15.000 rpm, el disco tarda 4 ms en girar una vez. El acceso medio
el tiempo (en mseg) para leer k bytes es entonces $8 + 2 + (k / 262144) \times 4$. Para bloques de 1 KB, 2 KB y 4 KB, los tiempos de acceso son 10.015625 mseg, 10.03125 mseg,
y 10.0625 mseg, respectivamente (apenas diferente). Estos dan tasas de
aproximadamente 102,240 KB / seg, 204,162 KB / seg y 407,056 KB / seg, respectivamente.

---

29. Si todos los archivos fueran de 1 KB, cada bloque de 2 KB contendría un archivo y 1 KB de espacio desperdiciado. No se permite intentar poner dos archivos en un bloque porque el
La unidad utilizada para realizar un seguimiento de los datos es el bloque, no el semibloque. Esto lleva a 50% de espacio desperdiciado. En la práctica, cada sistema de archivos tiene archivos grandes, así como muchos pequeños, y estos archivos usan el disco de manera mucho más eficiente. Por ex-
amplio, un archivo de 32,769 bytes usaría 17 bloques de disco para almacenamiento, dado un espacio eficiencia de 32768/34816, que es aproximadamente del 94%.
30. El bloque más grande es 32,768. Con 32,768 de estos bloques, el archivo más grande
sería 1 GB.
31. Restringe la suma de todas las longitudes de los archivos para que no sean más grandes que el disco. Esta no es una limitación muy seria. Si los archivos fueran colectivamente mayores que
el disco, no habría lugar para almacenarlos todos en el disco.
32. El i-nodo tiene 10 punteros. El bloque indirecto único contiene 256 punteros.
El bloque indirecto doble es bueno para 256^2 punteros. El triple bloqueo indirecto
es bueno para 256^3 punteros. Sumando estos, obtenemos un tamaño de archivo máximo de 16.843.018 bloques, que son aproximadamente 16,06 GB.
33. Se necesitan las siguientes lecturas de disco:
directorio para /
i-nodo para */ usr*
directorio para */ usr*
i-nodo para */ usr / ast*
directorio para */ usr / ast*
i-nodo para */ usr / ast / cursos*
directorio para */ usr / ast / cursos*
i-nodo para */ usr / ast / cursos / os*
directorio para */ usr / ast /ourses / os*
i-nodo para */usr/ast/courses/os/handout.t*
En total, se requieren 10 lecturas de disco.
34. Algunas ventajas son las siguientes. En primer lugar, no se desperdicia espacio en disco en i-nodos no utilizados.
En segundo lugar, no es posible quedarse sin i-nodos. En tercer lugar, hay menos movimiento del disco necesario ya que el i-nodo y los datos iniciales se pueden leer en una sola operación.
Algunas desventajas son las siguientes. Primero, las entradas del directorio ahora necesitarán un disco de 32 bits
dirección en lugar de un número de i-nodo de 16 bits. En segundo lugar, se utilizará un disco completo incluso para archivos que no contienen datos (archivos vacíos, archivos de dispositivo). En tercer lugar, el sistema de archivos
Las verificaciones de integridad del sistema serán más lentas debido a la necesidad de leer un bloque para cada i-nodo y porque los i-nodos estarán dispersos por todo el disco.
Cuarto, los archivos cuyo tamaño ha sido cuidadosamente diseñado para ajustarse al tamaño del bloque ya no se ajusta al tamaño del bloque debido al i-nodo, arruinando el rendimiento.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 4

21

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 5

1. En la figura, vemos un controlador con dos dispositivos. La razón por la que un solo
Se espera que el controlador maneje múltiples dispositivos es para eliminar la necesidad de
tener un controlador por dispositivo. Si los controladores se vuelven casi libres, entonces
Sería más sencillo construir el controlador en el propio dispositivo. Este diseño
También permiten múltiples transferencias en paralelo y así dan un mejor rendimiento.

2. Fácil. El escáner genera un máximo de 400 KB / seg. La red inalámbrica funciona
a 6,75 MB / s, por lo que no hay ningún problema.

3. No es una buena idea. El bus de memoria es seguramente más rápido que el bus de E / S, de otro modo
sabio por qué molestarse con él? Considere lo que sucede con una memoria normal
búsqueda. El bus de memoria termina primero, pero el bus de E / S todavía está ocupado. Si la CPU
espera hasta que finalice el bus de E / S, ha reducido el rendimiento de la memoria al de
el bus de E / S. Si solo intenta el bus de memoria para la segunda referencia,
falla si se trata de una referencia de dispositivo de E / S. Si hubiera alguna forma de instaurar
abortar instantáneamente la referencia del bus de E / S anterior para probar la segunda, la
la mejora puede funcionar, pero nunca existe esa opción. Con todo, es un
mala idea.

4. (a) Modo palabra a la vez: $1000 \times [(t_1 + t_2) + (t_1 + t_2) + (t_1 + t_2)]$
Donde el primer término es para adquirir el bus y enviar el comando al
controlador de disco, el segundo término es para transferir la palabra y el tercer término
es para el reconocimiento. En total, un total de $3000 \times (t_1 + t_2)$ nseg.
(b) Modo de ráfaga: $(t_1 + t_2) + t_1 + 1000$ veces $t_2 + (t_1 + t_2)$
donde el primer término es para adquirir el bus y enviar el comando al
controlador de disco, el segundo término es para que el controlador de disco adquiera el bus,
el tercer término es para la transferencia de ráfagas, y el cuarto término es para adquirir la
bus y haciendo el reconocimiento. En total, un total de $3 t_1 + 1002 t_2$.

5. Una interrupción requiere colocar 34 palabras en la pila. Regresando del interior
terrupt requiere obtener 34 palabras de la pila. Solo esta sobrecarga es 680
nsec. Por tanto, el número máximo de interrupciones por segundo no es más de
alrededor de 1,47 millones, suponiendo que no haya trabajo por cada interrupción.

6. La tasa de ejecución de una CPU moderna está determinada por el número de instrucciones
que terminan por segundo y tienen poco que ver con la duración de una instrucción
toma. Si una CPU puede terminar mil millones de instrucciones / seg, es una máquina de 1000 MIPS.
chine, incluso si una instrucción tarda 30 nseg. Por lo tanto, generalmente hay poca
tentar a hacer que las instrucciones terminen rápidamente. Manteniendo la interrupción hasta el último
La instrucción que se ejecuta actualmente termina puede aumentar la latencia de las interrupciones.
apreciablemente. Además, se requiere cierta administración para hacerlo bien.

22

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 5

7. Podría haberse hecho al principio. Una razón para hacerlo al final es que
el código del procedimiento del servicio de interrupción es muy corto. Primero emitiendo
otro carácter y luego reconocer la interrupción, si otra interrupción
ocurre inmediatamente, la impresora funcionará durante la interrupción, lo que
imprime un poco más rápido. Una desventaja de este enfoque es que muere un poco más
tiempo en el que se pueden desactivar otras interrupciones.

8. Sí. La PC apilada apunta a la primera instrucción no obtenida. Todas las instrucciones

anteriores han sido ejecutadas y la instrucción apuntada y su
los sucesores no han sido ejecutados. Esta es la condición para una interpretación precisa
se rompe. Las interrupciones precisas no son difíciles de lograr en una máquina con un solo
tubería. El problema surge cuando las instrucciones se ejecutan fuera de orden,
que no es el caso aquí.

9. La impresora imprime 50 × 80 × 6 = 24.000 caracteres / min, que son 400 caracteres.
ters / seg. Cada carácter usa 50 μseg de tiempo de CPU para la interrupción, por lo que
En cada segundo, la sobrecarga de interrupción es de 20 mseg. Usando interrupción
E / S impulsadas, los 980 milisegundos de tiempo restantes están disponibles para otros trabajos. En
En otras palabras, la sobrecarga de interrupción cuesta solo el 2% de la CPU, lo que
apenas afectan al programa en ejecución.

10. UNIX lo hace de la siguiente manera. Hay una tabla indexada por número de dispositivo, con
cada entrada de la tabla es una estructura C que contiene punteros a las funciones para
ing, cierre, lectura y escritura y algunas otras cosas desde el dispositivo. A
instalar un nuevo dispositivo, se debe realizar una nueva entrada en esta tabla y los punteros
rellenado, a menudo en el controlador de dispositivo recién cargado.

11. (a) Controlador de dispositivo.
(b) Controlador de dispositivo.
(c) Software independiente del dispositivo.
(d) Software a nivel de usuario.

12. Un paquete debe copiarse cuatro veces durante este proceso, que toma 4.1
mseg. También hay dos interrupciones, que representan 2 mseg. Finalmente, el
el tiempo de transmisión es de 0,83 mseg, para un total de 6,93 mseg por 1024 bytes. los
La velocidad máxima de datos es por lo tanto 147,763 bytes / seg, o aproximadamente el 12% de la 10
nominal
capacidad de red megabit / seg. (Si incluimos la sobrecarga del protocolo, las cifras
empeorar.)

13. Si la impresora se asignó tan pronto como apareció la salida, un proceso podría
atar la impresora imprimiendo algunos caracteres y luego irse a dormir por un
semana.

14. El nivel 2 de RAID no solo puede recuperarse de unidades averiadas, sino también de
errores transitorios no detectados. Si una unidad entrega un solo bit defectuoso, el nivel de RAID
2 corregirá esto, pero el nivel RAID 3 no.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 5

23

15. La probabilidad de 0 fallas, P_0, es $(1 - p)^k$. La probabilidad de 1 falla, P_1,
es $kp(1 - p)^{k-1}$
. La probabilidad de una falla de RAID es entonces $1 - P_0 - P_1$. Esta
es $1 - (1 - p)^k - kp(1 - p)^{k-1}$
.

16. Rendimiento de lectura: los niveles RAID 0, 2, 3, 4 y 5 permiten lecturas paralelas
servicio una solicitud de lectura. Sin embargo, el nivel RAID 1 permite además dos lecturas
misiones para continuar simultáneamente. Rendimiento de escritura: todos los niveles de RAID
Vide un rendimiento de escritura similar. Espacio superior: no hay espacio superior
en el nivel 0 y 100% de sobrecarga en el nivel 1. Con palabra de datos de 32 bits y seis paridad
unidades, la sobrecarga de espacio es de aproximadamente 18,75% en el nivel 2. Para una palabra de
datos de 32 bits,
la sobrecarga de espacio en el nivel 3 es de aproximadamente 3,13%. Finalmente, asumiendo 33
unidades en
niveles 4 y 5, la sobrecarga de espacio es del 3,13% en ellos. Fiabilidad: no hay

soporte de confiabilidad en el nivel 0. Todos los demás niveles de RAID pueden sobrevivir a un disco choque. Además, en los niveles 3, 4 y 5, un solo error de bit aleatorio en una palabra puede ser detectado, mientras que en el nivel 2, un solo error de bit aleatorio en una palabra se puede detectado y corregido.

17. Se genera un campo magnético entre dos polos. No solo es difícil hacer que la fuente de un campo magnético sea pequeña, pero también el campo se propaga rápidamente, lo que conduce a problemas mecánicos al tratar de mantener la superficie de un magnético medio cerca de una fuente magnética o sensor. Un láser semiconductor generado borra la luz en un lugar muy pequeño, y la luz se puede manipular ópticamente para iluminar un punto muy pequeño a una distancia relativamente grande de la fuente.

18. La principal ventaja de los discos ópticos es que tienen una grabación mucho más alta densidades que los discos magnéticos. La principal ventaja de los discos magnéticos es que son un orden de magnitud más rápidos que los discos ópticos.

19. Posiblemente. Si la mayoría de los archivos se almacenan en sectores lógicamente consecutivos, podría ser Vale la pena intercalar los sectores para dar tiempo a los programas para procesar los datos. recién recibido, de modo que cuando se emita la próxima solicitud, el disco esté en el lugar correcto. Si esto vale la pena depende en gran medida del tipo de los programas se ejecutan y cuán uniforme es su comportamiento.

20. Quizás sí y quizás no. El intercalado doble es efectivamente un sesgo de cilindro de dos sectores. Si la cabeza puede realizar una búsqueda de pista a pista en menos de dos tiempos de sector, que no se necesita ningún sesgo adicional del cilindro. Si no puede, entonces Se necesita un sesgo de cilindro adicional para evitar perder un sector después de una búsqueda.

21.

(a) La capacidad de una zona es pistas × cilindros × sectores / cilindro × bytes / sección.
Capacidad de la zona 1: 16 × 100 × 160 × 512 = 131072000 bytes
Capacidad de la zona 2: 16 × 100 × 200 × 512 = 163840000 bytes
Capacidad de la zona 3: 16 × 100 × 240 × 512 = 196608000 bytes
Capacidad de la zona 4:16 × 100 × 280 × 512 = 229376000 bytes
Suma = 131072000 + 163840000 + 196608000 + 229376000 = 720896000

(b) Una tasa de rotación de 7200 significa que hay 120 rotaciones / seg. En el 1 mseg tiempo de búsqueda pista a pista, se cubren 0,120 de los sectores. En la zona 1, el La cabeza del disco pasará por encima de 0,120 × 160 sectores en 1 mseg, por lo que la pista óptima el sesgo para la zona 1 es de 19,2 sectores. En la zona 2, la cabeza del disco pasará 0.120 × 200 sectores en 1 mseg, por lo que la desviación óptima de la pista para la zona 2 es de 24 segundos.
tors. En la zona 3, la cabeza del disco pasará por 0,120 × 240 sectores en 1 mseg, por tanto, la desviación óptima de la pista para la zona 3 es de 28,8 sectores. En la zona 4, la cabeza del disco pasará más de 0.120 × 280 sectores en 1 mseg, por lo tanto, la desviación de pista óptima para la zona 3 es 33,6 sectores.

(c) La velocidad máxima de transferencia de datos será cuando los cilindros en el exterior- la mayoría de las zonas (zona 4) se están leyendo / escribiendo. En esa zona, en un segundo, 280 sectores se leen 120 veces. Por lo tanto, la velocidad de datos es 280 × 120 × 512 = 17,203,200 bytes / seg.

22. La capacidad de la unidad y las tasas de transferencia se duplican. El tiempo de búsqueda y el promedio

El retardo rotacional es el mismo.
23. Una consecuencia bastante obvia es que ningún sistema operativo existente
funcionan porque todos miran allí para ver dónde están las particiones del disco. Chang-
El formato de la tabla de particiones hará que todos los sistemas operativos
fallar. La única forma de cambiar la tabla de particiones es cambiar simultáneamente
todos los sistemas operativos para utilizar el nuevo formato.
24. (a) 10 + 12 + 2 + 18 + 38 + 34 + 32 = 146 cilindros = 876 mseg.
(b) 0 + 2 + 12 + 4 + 4 + 36 +2
= 60 cilindros = 360 mseg.
(c) 0 + 2 + 16 + 2 + 30 + 4 + 4
= 58 cilindros = 348 mseg.
25. En el peor de los casos, una solicitud de lectura / escritura no se atiende durante casi dos
escaneos en el algoritmo del elevador, mientras que es como máximo un escaneo de disco completo en
el
algoritmo modificado.
26. Hay una carrera pero no importa. Dado que la escritura estable en sí ya ha
completado, el hecho de que la RAM no volátil no se haya actualizado solo
significa que el programa de recuperación sabrá qué bloque se estaba escribiendo.
Leerá ambas copias. Al encontrarlos idénticos, no cambiará ninguno, lo que
es la acción correcta. El efecto del accidente justo antes de la RAM no volátil
se actualizó solo significa que el programa de recuperación tendrá que hacer dos discos
lee más de lo que debería.
27. Sí, el disco permanece consistente incluso si la CPU falla durante una recuperación
procedimiento. Considere la figura 5-31. No hay recuperación involucrada en (a) o (e).
Suponga que la CPU falla durante la recuperación en (b). Si la CPU falla antes
el bloque de la unidad 2 se ha copiado completamente a la unidad 1, la situación
sigue siendo el mismo que antes. El procedimiento de recuperación posterior detectará una
Error de ECC en la unidad 1 y vuelva a copiar el bloque de la unidad 2 a la unidad 1. Si la CPU
se bloquea después de que el bloque de la unidad 2 se haya copiado a la unidad 1, la situación es

---

igual que en el caso (e). Suponga que la CPU falla durante la recuperación en (c).
Si la CPU falla antes de que el bloque de la unidad 1 se haya copiado completamente en
unidad 2, la situación es la misma que en el caso (d). El posterior pro-
cedure detectará un error ECC en la unidad 2 y copiará el bloque de la unidad 1 a
unidad 2. Si la CPU falla después de que el bloque de la unidad 1 se haya copiado a la unidad
2, la situación es la misma que en el caso (e). Finalmente, suponga que la CPU falla
durante la recuperación en (d). Si la CPU falla antes de que el bloque de la unidad 1 haya sido
copiado completamente en la unidad 2, la situación sigue siendo la misma que antes. El sub-
El siguiente procedimiento de recuperación detectará un error ECC en la unidad 2 y nuevamente
copie el bloque de la unidad 1 a la unidad 2. Si la CPU falla después del bloque de
la unidad 1 se ha copiado en la unidad 2, la situación es la misma que en el caso (e).
28. Dos milisegundos 60 veces por segundo son 120 milisegundos / segundo, o el 12% de la CPU
29.
(a) Con un cristal de 500 MHz, el contador se puede disminuir cada 2 nseg. Entonces,
para un tic cada milisegundo, el registro debe ser 1000000/2 = 500,000.
(b) Para obtener un tic del reloj cada 100 μseg, el valor del registro de retención debe ser 50 000.
30. En el momento 5000:
Hora actual = 5000; Siguiente señal = 8; Encabezado → 8 → 4 → 3 → 14 → 8.
En el momento 5005:

Hora actual = 5005; Siguiente señal = 3; Encabezado → 3 → 4 → 3 → 14 → 8.
En el momento 5013:
Hora actual = 5013; Siguiente señal = 2; Encabezado 2 → 14 → 8.
En el momento 5023:
Hora actual = 5023; Siguiente señal = 6; Encabezado → 6 → 4 → 5.
31. El número de segundos en un año medio es 365,25 × 24 × 3600. Este número es
31.557.600. El contador se reinicia después de 2^{32} segundos a partir del 1 de enero.
1970. El valor de 2^{32} / 31,557,600 es 136,1 años, por lo que la envoltura se producirá en
2106.1, que es a principios de febrero de 2106. Por supuesto, para entonces, todas las computadoras
ser de al menos 64 bits, por lo que no sucederá en absoluto.
32. Desplazarse por la ventana requiere copiar 59 líneas de 80 caracteres o 4720 caracteres.
acters. Copiar 1 carácter (16 bytes) toma 800 nseg, por lo que toda la ventana
tarda 3.776 ms. Escribir 80 caracteres en la pantalla toma 400 nseg, por lo que
desplazarse y mostrar una nueva línea toma 4.176 ms. Esto da alrededor de 239,5
líneas / seg.
33. Suponga que el usuario le pide inadvertidamente al editor que imprima miles de
líneas. Luego pulsa DEL para detenerlo. Si el controlador no descartó la salida, la salida
podría continuar durante varios segundos después de la DEL, lo que haría que el usuario
presione DEL una y otra vez y se frustre cuando no pasó nada.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 5
34. Las 25 líneas de caracteres, cada una de 8 píxeles de alto, requieren 200 escaneos para dibujar.
Hay 60 pantallas por segundo, o 12.000 exploraciones / seg. A 63,6 μseg / escaneo, el
El haz se mueve horizontalmente 763 ms por segundo, dejando 237 ms para escritura.
ing en la RAM de video. Por tanto, la RAM de vídeo está disponible el 23,7% del tiempo.
35. La velocidad máxima que puede mover el mouse es de 200 mm / seg, que es 2000
mickey / seg. Si cada informe es de 3 bytes, la velocidad de salida es de 6000 bytes / seg.
36. Con un sistema de color de 24 bits, solo se pueden representar 2^{24} colores. Esto no es todo
de ellos. Por ejemplo, supongamos que un fotógrafo toma fotografías de 300 latas
de pintura azul pura, cada una con una cantidad de pigmento ligeramente diferente. El primero
podría estar representado por el valor (R, G, B) (0, 0, 1). El próximo podría ser
representado por (0, 0, 2), y así sucesivamente. Dado que la coordenada B tiene solo 8 bits,
no hay forma de representar 300 valores diferentes de azul puro. Algunos de los
las fotografías deberán representarse con el color incorrecto. Otro ejemplo es
el color (120,24, 150,47, 135,89). No se puede representar, solo aproximadamente
apareado por (120, 150, 136).
37.
(a) Cada píxel ocupa 3 bytes en RGB, por lo que el espacio de la tabla es 16 × 24 × 3 bytes,
que es 1152 bytes.
(b) A 100 nseg por byte, cada carácter tarda 115,2 μseg. Esto da una salida
velocidad de aproximadamente 8681 caracteres / seg.
38. Reescribir la pantalla de texto requiere copiar 2000 bytes, lo que se puede hacer en
20 μsegundos. Reescribir la pantalla gráfica requiere copiar 1024 × 768 × 3
= 2,359,296 bytes, o aproximadamente 23,6 mseg.
39. En Windows, el sistema operativo llama a los propios procedimientos del controlador. En X
Windows, nada
ing así sucede. X simplemente recibe un mensaje y lo procesa internamente.
40. El primer parámetro es fundamental. En primer lugar, las coordenadas son relativas a
alguna ventana, por lo que se necesita *hdc* para especificar la ventana y, por lo tanto, el origen.
En segundo lugar, el rectángulo se recortará si cae fuera de la ventana, por lo que

se necesitan las coordenadas de la ventana. En tercer lugar, el color y otras propiedades del rectangle se toman del contexto especificado por *hdc* . Es bastante esencial.

41. El tamaño de la pantalla es 400 × 160 × 3 bytes, que es 192.000 bytes. A 10 fps esto es 1.920.000 bytes / seg o 15.360.000 bits / seg. Esto consume el 15% del Fast Ethernet.

42. El ancho de banda en un segmento de red se comparte, por lo que 100 usuarios que solicitan diferentes

Diferentes datos simultáneamente en una red de 1 Mbps verán cada uno un efecto de 10 Kbps velocidad efectiva. Con una red compartida, un programa de TV puede ser multidifusión, por lo que los paquetes de video solo se transmiten una vez, sin importar cuántos usuarios haya y debería funcionar bien. Con 100 usuarios navegando por la Web, cada usuario obtendrá 1/100 del ancho de banda, por lo que el rendimiento puede degradarse muy rápidamente.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 5

27

43. Si $n = 10$, la CPU aún puede realizar su trabajo a tiempo, pero la energía utilizada cae apreciablemente. Si la energía consumida en 1 segundo a máxima velocidad es E , entonces funcionando a máxima velocidad durante 100 mseg y luego inactivo durante 900 mseg utiliza $E / 10$. Correr a 1/10 de velocidad durante un segundo completo usa $E / 100$, un ahorro de 9 $E / 100$. El porcentaje de ahorro al cortar el voltaje es del 90%.

44. El sistema de ventanas utiliza mucha más memoria para su visualización y utiliza vir- memoria tual más que el modo de texto. Esto hace que sea menos probable que el El disco estará inactivo durante un período de tiempo suficiente para que se active automáticamente. apagado.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 6

1. En los Estados Unidos, considere una elección presidencial en la que tres o más candidatos están intentando la nominación de algún partido. Después de todas las elecciones primarias terminados, cuando los delegados lleguen a la convención del partido, podría suceder pluma que ningún candidato tiene mayoría y que ningún delegado está dispuesto a cambiar su voto. Este es un punto muerto. Cada candidato tiene algunos recursos (votos) pero necesita más para hacer el trabajo. En países con múltiples políticas partidos en el parlamento, puede suceder que cada partido apoye a un versión diferente del presupuesto anual y que es imposible armar un mayoría para aprobar el presupuesto. Esto también es un punto muerto.

2. El espacio en disco de la partición de spool es un recurso finito. Cada bloque que entra de facto reclama un recurso y cada nuevo que llega quiere más recursos. Si el espacio en cola es, digamos, 10 MB y la primera mitad de diez 2 MB llegan trabajos, el disco estará lleno y no se pueden almacenar más bloques, por lo que tenemos un punto muerto. El interbloqueo se puede evitar permitiendo que un trabajo comience a imprimirse antes de que esté completamente enrollado y reservando el espacio así liberado para el resto de ese trabajo. De esta manera, un trabajo se imprimirá hasta su finalización, luego el siguiente uno puede hacer lo mismo. Si los trabajos no pueden comenzar a imprimirse hasta que estén completamente

en cola, es posible un interbloqueo.

3. Sí. No hace ninguna diferencia en absoluto.

4. Suponga que hay tres procesos, A , B y C , y dos tipos de recursos, R y S . Supóngase además que hay una instancia de R y dos instancia de S . Considere el siguiente escenario de ejecución: A solicita R y lo obtiene; B re misiones S y obtiene; C solicita S y lo obtiene (hay dos instancias de S); B re misiones R y está bloqueado; A solicita S y se bloquea. En esta etapa los cuatro las condiciones se mantienen. Sin embargo, no existe un punto muerto. Cuando C termina, uno en

postura de *S* se libera que se asigna a *A* . Ahora *A* puede completar su ejecución
y liberar *R* que se puede asignar a *B* , que luego puede completar su ex-
ecuación. Estas 4 condiciones son suficientes si hay 1 recurso de cada tipo.

28

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 6

5. Sí, existen gráficos ilegales. Dijimos que un recurso solo puede estar en manos de un
proceso único. Un arco desde un cuadrado de recursos a un círculo de proceso indica
que el proceso es propietario del recurso. Así, un cuadrado con arcos que van de él a
dos o más procesos significa que todos esos procesos contienen el recurso,
que viola las reglas. En consecuencia, cualquier gráfico en el que varios arcos
dejar un cuadrado y terminar en diferentes círculos viola las reglas a menos que haya
múltiples copias de los recursos. Arcos de cuadrados a cuadrados o de círculos
a los círculos también violan las reglas.

6. Considere tres procesos, *A* , *B* y *C* y dos recursos *R* y *S* . Suponga que *A* es
esperando por *I* que está en manos de *B* , *B* está esperando por *S en* manos de *A* y *C* está esperando
para *R* en poder de *A* . Los tres procesos, *A* , *B* y *C* están bloqueados. Sin embargo,
solo *A* y *B* pertenecen a la cadena circular.

7. Una parte de todos esos recursos podría reservarse para que la utilicen únicamente los procesos.
propiedad del administrador, por lo que siempre puede ejecutar un shell y
gramos necesarios para evaluar un punto muerto y tomar decisiones sobre qué proceso
esses to kill para hacer que el sistema sea utilizable nuevamente.

8. Recuperación mediante preferencia: una vez que se completen los procesos *P2* y *P3* , procese
P1 puede ser forzado a apropiarse de 1 unidad de *RS3* . Esto hará que A = (0 2 1 3 2),
y permita que se complete el proceso *P4* . Una vez que *P4 se* completa y libera su
fuentes *P1* pueden completar. Recuperación mediante reversión: Revertir *P1* al
Estado controlado antes de adquirir *RS3* . Recuperación a través del proceso de matar
esses: Mata a *P1* .

9. El proceso está pidiendo más recursos de los que tiene el sistema. No hay
manera concebible en que puede obtener estos recursos, por lo que nunca puede terminar, incluso si no
otros procesos quieren cualquier recurso en absoluto.

10. El modelo que se muestra en la Figura 6-8 requiere conocimiento de cuándo un proceso
arrienda sus recursos. Por otro lado, decidir si un estado es seguro o
inseguro no requiere este conocimiento. Una consecuencia de esto es que el
El modelo de la Figura 6-8 se puede utilizar para identificar las cajas como la
limitada por I_1 , I_2 , I_5 e I_6 que garantizan que el sistema eventualmente
punto muerto, un estado inseguro significa que no hay garantía de que un punto muerto
no ocurrió.

11. Sí. Haz todo en tres dimensiones. El eje *z* mide el número
número de instrucciones ejecutadas por el tercer proceso.

12. El método solo se puede utilizar para guiar la programación si el instante exacto en
qué recurso se va a reclamar se conoce de antemano. En la práctica, este
Rara vez es el caso.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 6

29

13. Hay estados que no son seguros ni estancados, pero que conducen a
estados bloqueados. Como ejemplo, suponga que tenemos cuatro recursos: cintas,
trazadores, escáneres y CD-ROM, como en el texto, y tres procesos que compiten
ing por ellos. Podríamos tener la siguiente situación:

Tiene
Necesidades
Disponible
A: 2000
1020
0121
B: 1000
0131
C: 0121
1010
Este estado no está bloqueado porque aún pueden ocurrir muchas acciones, por ejemplo bastante, *A* todavía puede conseguir dos impresoras. Sin embargo, si cada proceso solicita su requisitos restantes, tenemos un punto muerto.

14. No. Se niega un recurso disponible a un proceso de solicitud en un sistema que usa el algoritmo del banquero, si existe la posibilidad de que el sistema se bloquee concediendo esa solicitud. Ciertamente es posible que el sistema no tenga bloqueado si esa solicitud fue concedida.

15. Aparte de forzar una ejecución secuencial de procesos, no es posible evitar puntos muertos si la información sobre las necesidades máximas de recursos de todos procesos no está disponible de antemano. Considere el ejemplo de dos procesos que quieren grabar un documento escaneado en un CD-ROM. Cuando el proceso *B* re-solicita la grabadora de CD-ROM, es imposible determinar si la concesión de esta búsqueda conducirá a un estado inseguro, porque no se sabe si *A* necesitará el Grabador de CD-ROM más tarde y *B* necesitará el escáner más tarde.

16. Una solicitud de *D* no es segura, pero una de *C* es segura.

17. El sistema está libre de interbloqueo. Suponga que cada proceso tiene un recurso. Hay un recurso gratuito. Cualquiera de los procesos puede solicitarlo y obtenerlo, en el que caso de que pueda terminar y liberar ambos recursos. En consecuencia, el punto muerto es im-posible.

18. Si un proceso tiene *m* recursos, puede terminar y no puede estar involucrado en un bloquear. Por lo tanto, el peor de los casos es cuando cada proceso tiene *m* - 1 recursos y necesita otro. Si queda un recurso, un proceso puede terminar ish y liberar todos sus recursos, dejando que el resto termine también. Por lo tanto, el con-La opción para evitar el punto muerto es $r \geq p\,(\,m - 1) + 1$.

19. No. *D* todavía puede terminar. Cuando termina, devuelve suficientes recursos para permitir *E* (o *A*) para terminar, y así sucesivamente.

20. Con tres procesos, cada uno puede tener dos unidades. Con cuatro procesos, el La distribución de unidades será (2, 2, 1, 1), lo que permitirá que los dos primeros procesos terminar. Con cinco procesos, la distribución será (2, 1, 1, 1, 1), que aún permite que termine el primero. Con seis, cada uno con una unidad de cinta y queriendo en otro, tenemos un punto muerto. Por tanto, para $n < 6$, el sistema está libre de interbloqueo.

21. La comparación de una fila en la matriz con el vector de recursos disponibles requiere *m* operaciones. Este paso debe repetirse en el orden de *n* veces para encontrar un proceso. que puede terminar y marcarse como terminado. Marcando así un proceso como hecho asume el orden de *mn* pasos. Repitiendo el algoritmo para todos los *n* procesos significa que el número de pasos es entonces mn^2 . Por tanto, $a = 1$ y $b = 2$.

22. La matriz de necesidades es la siguiente:
01002

02100
10300
00111
Si *x* es 0, tenemos un punto muerto inmediatamente. Si *x* es 1, el proceso *D* puede ejecutarse hasta terminación. Cuando termina, el vector disponible es 1 1 2 2 1.
afortunadamente ahora estamos estancados. Si *x* es 2, después de que *D se* ejecuta, el vector disponible es
1 1 3 2 1 y *C* pueden funcionar. Después de que termine y devuelva sus recursos, la disponibilidad capaz vector es 2 2 3 3 1, que permitirá *B* para ejecutar y completa, y entonces *A* a
ejecutar y completar. Por lo tanto, el valor más pequeño de *x* que evita un punto muerto es 2.

23. Considere un proceso que necesita copiar un archivo enorme de una cinta a una impresora. Serporque la cantidad de memoria es limitada y el archivo completo no cabe en este
memoria, el proceso tendrá que recorrer las siguientes declaraciones hasta que
se ha impreso todo el archivo:
Adquirir unidad de cinta
Copie la siguiente parte del archivo en la memoria (tamaño de memoria limitado)
Liberar unidad de cinta
Adquirir impresora
Imprimir archivo desde la memoria
Liberar impresora
Esto alargará el tiempo de ejecución del proceso. Además, dado que
la impresora se libera después de cada paso de impresión, no hay garantía de que todas las partes del archivo se imprimirá en páginas continuas.

24. Suponga que el proceso *A* solicita los registros en el orden *a* , *b* , *c* . Si el proceso *B* también pide *un* primero, uno de ellos lo obtendrá y el otro bloqueará. Esta
La situación siempre está libre de interbloqueo ya que el ganador ahora puede correr hasta el final sin interferencias. De las otras cuatro combinaciones, algunas pueden conducir a
interbloqueo y algunos están libres de interbloqueo. Los seis casos son los siguientes:
a B C
punto muerto libre
acb
punto muerto libre
bac
posible punto muerto
bca
posible punto muerto

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 6

31

taxi
posible punto muerto
cba
posible punto muerto
Dado que cuatro de los seis pueden conducir a un punto muerto, existe una probabilidad de 1/3 de evitar un
punto muerto y una probabilidad de 2/3 de conseguir uno.

25. Sí. Supongamos que todos los buzones de correo están vacíos. Ahora *A* envía a *B* y espera para una respuesta, *B* envía a *C* y espera una respuesta, y *C* envía a *A* y espera
una respuesta. Ahora se cumplen todas las condiciones para un bloqueo de comunicaciones.

26. Para evitar esperas circulares, numere los recursos (las cuentas) con su cuenta

números. Después de leer una línea de entrada, un proceso bloquea la cuenta de número inferior.
contar primero, luego, cuando obtiene el bloqueo (lo que puede implicar una espera), bloquea el
otro. Dado que ningún proceso espera nunca una cuenta inferior a la que
listo tiene, nunca hay una espera circular, por lo tanto, nunca un punto muerto.
27. Cambie la semántica para solicitar un nuevo recurso de la siguiente manera. Si un proceso
solicita un nuevo recurso y está disponible, obtiene el recurso y conserva lo
ya lo ha hecho. Si el nuevo recurso no está disponible, todos los recursos existentes se
publicado. Con este escenario, el punto muerto es imposible y no hay peligro
que se adquiere el nuevo recurso pero se pierden los existentes. Por supuesto, el proceso
ess solo funciona si es posible liberar un recurso (puede liberar un escáner
entre páginas o una grabadora de CD entre CD).
28. Le daría una calificación F (reprobatoria). ¿Qué hace el proceso? Ya que claramente
necesita el recurso, simplemente vuelve a preguntar y bloquea de nuevo. Esto no es mejor que
permanecer bloqueado. De hecho, puede ser peor ya que el sistema puede realizar un seguimiento de
cuánto tiempo han estado esperando los procesos de la competencia y asignan un recién liberado
recurso al proceso que ha estado esperando por más tiempo. Al cronometrar periódicamente
y volver a intentarlo, un proceso pierde su antigüedad.
29. Un interbloqueo ocurre cuando un conjunto de procesos se bloquea esperando un evento.
que solo algún otro proceso en el conjunto puede causar. Por otro lado, proc-
los esses en un livelock no están bloqueados. En cambio, continúan ejecutando la verificación
para que se cumpla una condición que nunca se hará realidad. Así, además
a los recursos que tienen, los procesos en livelock continúan consumiendo
precioso tiempo de CPU. Finalmente, la inanición de un proceso ocurre debido a la
presencia de otros procesos, así como un flujo de nuevos procesos entrantes
que terminan con una prioridad más alta que el proceso que se está muriendo de hambre. A diferencia de los muertos
bloqueo o livelock, la inanición puede terminar por sí sola, por ejemplo, cuando el proceso existente
los esses con mayor prioridad terminan y no hay nuevos procesos con mayor prioridad
llegar.
30. Si ambos programas solicitan Woofer primero, las computadoras se morirán de hambre con el
menos secuencia: solicitar woofer, cancelar solicitud, solicitar woofer, cancelar
búsqueda, y así sucesivamente. Si uno de ellos pide la caseta del perro y el otro pide
el perro, tenemos un punto muerto, que es detectado por ambas partes y luego

---

roto, pero simplemente se repite en el siguiente ciclo. De cualquier manera, si ambos
Los ordenadores han sido programados para ir tras el perro o la caseta del perro primero, ya sea
sobreviene el hambre o el estancamiento. Realmente no hay mucha diferencia entre
los dos aquí. En la mayoría de los problemas de estancamiento, la inanición no parece ser un problema grave.
porque la introducción de retrasos aleatorios generalmente lo hará muy poco probable. Que ap-
proach no funciona aquí.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 7

1. La televisión NTSC estándar tiene aproximadamente 640 × 480 píxeles. A 8 bits / píxel y 30
cuadros / seg obtenemos un ancho de banda de 73 Mbps. Apenas lo hace con uno
canal. Dos canales sería demasiado.
2. Según la tabla, HDTV es 1280 × 720 frente a 640 × 480 para la TV normal. Tiene
tres veces más píxeles y, por lo tanto, necesita tres veces el ancho de banda. los
La razón por la que no necesita cuatro veces más ancho de banda es que la relación de aspecto
de HDTV es diferente de la TV convencional para coincidir con la de la película de 35 mm

ter.
3. Para avanzar en cámara lenta, es suficiente que se muestre cada cuadro
dos o más veces seguidas. No se necesita ningún archivo adicional. Retroceder
lento es tan malo como retroceder rápidamente, por lo que se necesita un archivo adicional.
4. Hay 32,768 magnitudes posibles. Por ejemplo, suponga que la señal
varía de -32,768 voltios a +32,767 voltios y el valor almacenado para cada muestra
ple es la señal redondeada al número más cercano de milivoltios, como un signo
Entero de 16 bits. Una señal de 16.0005 voltios tendría que registrarse como
16.000 o como 16.001. El error porcentual aquí es 1/320 por ciento. Sin embargo, sup
plantean la señal es de 0.0005 voltios. Esto se registra en 0 o 1. En este último
caso, el error es del 50%. Por lo tanto, el ruido de cuantificación afecta más a las amplitudes bajas.
que altas amplitudes. Los conciertos de flauta serán más fuertes que el rock and roll
debido a sus menores amplitudes.
5. Se podría implementar un esquema de expansión / compresión de volumen de la siguiente manera.
Un bit de la salida está reservado para señalar que la señal grabada está
panded. Los 15 bits restantes se utilizan para la señal. Cuando el orden superior 5
los bits de la señal de 20 bits no son 00000, el bit de expansión es 0 y los otros 15
Los bits contienen los 15 bits de orden superior de los datos muestreados. Cuando el orden superior
5 bits de la señal son 00000, el bit de expansión está activado y el de 20 bits
la señal de amplitud se desplaza a la izquierda 5 bits. Al final del oyente el proceso inverso
tiene lugar. Este esquema aumenta ligeramente el ruido de cuantificación para señales fuertes.
nals (debido a una señal de 15 bits en lugar de una señal de 16 bits), pero la disminuye para
señales silenciosas, cuando el efecto de la cuantificación es más notable. La mayor
La desventaja es que esto no es un estándar y no funcionaría con los
Reproductores de CD, pero podría funcionar para música en línea reproducida con un complemento
especial

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 7
33
que utilizó este esquema en ambos extremos. Una versión más sofisticada podría usar 2
bits para indicar cuatro regímenes de expansión diferentes para diferentes niveles de señal.
6. La diferencia no causa ningún problema. El algoritmo DCT se utiliza para
codificar fotogramas I en un esquema similar a JPEG. Los macrobloques se utilizan en P-
fotogramas para localizar macrobloques que aparecieron en fotogramas anteriores. Los dos
las cosas no tienen nada que ver entre sí y no entran en conflicto.
7. No, no lo hacen. El algoritmo de compensación de movimiento encontrará cada macro-
bloque en el cuadro anterior en algún desplazamiento de su ubicación actual. Por en-
codificando el hecho de que el macrobloque actual debe tomarse del anterior
fotograma en una posición (Δx , Δy) de la actual, no es necesario trans-
mit el bloque en sí de nuevo.
8. Los procesos que soportan las tres secuencias de video ya usan 0.808 del
Tiempo de CPU, por lo que quedan 192 mseg por segundo para el audio. Proceso de audio
ess *A se* ejecuta 33.333 veces / seg, el proceso de audio *B se* ejecuta 25 veces / seg y
el proceso *C se* ejecuta 20 veces / seg, para un total de 78,333 ejecuciones / seg. Estas 78.333 carreras
puede usar 192 mseg, por lo que cada ejecución puede usar 192 / 78.333 o 2.45 mseg.
9. El primer proceso usa 0.400 de la CPU. El segundo usa 0.375 del
UPC. Juntos usan 0,775. El límite de RMS para dos procesos es
$2 \times (2^{0,5} - 1)$, que es 0,828, por lo que se garantiza que el valor eficaz funciona.
10. Desde 0,65 <ln 2, RMS siempre puede programar las películas, sin importar cuántas
existen. Por tanto, RMS no limita el número de películas.
11. La secuencia que comienza en $t = 150$ es *A* 6, *B* 5, *C* 4, *A* 7, *B* 6, *A* 5 y *C* 5. Cuando

C 5 termina en $t = 235$ no hay trabajo por hacer hasta $t = 240$ cuando *A* y *B se* vuelven
listo, por lo que el sistema permanece inactivo durante 5 mseg. La elección de ejecutar *B* 5 antes
C 4 es arbitrario. También se permite la otra forma.
12. Un lector de DVD está bien para verlo en casa, pero el tiempo de búsqueda alto de la opción actual
icos sistemas de grabación limitan su utilidad a proporcionar un único flujo de
datos. Las unidades de DVD no pueden admitir múltiples transmisiones con diferentes horas de inicio
o funciones de control similares a las de una videograbadora, como pausa, rebobinado y avance rápido
para
diferentes usuarios. Con la tecnología actual, los datos tendrían que almacenarse en búfer en
una memoria extremadamente grande. Los discos duros son simplemente mejores.
13. Si la espera en el peor de los casos es de 6 minutos, se debe iniciar una nueva transmisión cada 6
minutos. Para
Película de 180 minutos, se necesitan 30 secuencias.
14. La velocidad de datos es de 0,5 MB / seg. Un minuto de video usa 30 MB. Para ir por-
Ward o hacia atrás 1 minuto cada uno requiere 60 MB.
15. Tenemos $2 \times \Delta T \times 2 \times 10_{6} \leq! 50 \times 2_{20} \times 8$. Entonces $\Delta T \leq 105$ seg.

34
SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 7
16. HDTV no hace ninguna diferencia. Todavía hay 216.000 fotogramas en el
película. El desperdicio de cada fotograma es aproximadamente la mitad de un bloque de disco, o 0,5
KB. por
toda la película, esta pérdida es de 108 KB.
17. Hay alguna pérdida para cada cuadro. Cuantos más fotogramas tengas, más pérdida
tienes. NTSC tiene una velocidad de fotogramas más alta, por lo que tiene un poco más de pérdida. Pero
dados los números involucrados, esta pérdida no es una fracción significativa del total
Espacio del disco.
18. El efecto principal de HDTV son los cuadros más grandes. Los marcos grandes tienden a hacer
La desventaja de los bloques pequeños es menos grave porque los marcos grandes se pueden leer en
eficientemente. Por lo tanto, el argumento del rendimiento del disco a favor de bloques grandes
disminuye. Además, si los marcos no se dividen en bloques (ya que no
aquí), tener I-frames que son una fracción sustancial de un bloque es una seria
problema. A menudo puede ocurrir que un bloque esté parcialmente lleno y un marco en I grande
aparece a continuación, desperdiciando una gran cantidad de espacio en el bloque actual. Sobre el
En conjunto, ir a HDTV favorece el modelo de bloque pequeño.
19. En promedio, se puede desperdiciar 1 KB por bloque. Una película PAL de 2 horas requiere
180.000 fotogramas. Por lo tanto, el desperdicio de espacio en disco total promedio para esta película es
180.000 KB.
20. Cada bloque de índice de cuadros puede almacenar hasta 2048/8 = 512 entradas. Desde cada uno
El marco puede ser tan grande como 255 × 2 KB = 510 KB, el tamaño de archivo más grande puede ser:
512 × 510KB = 261.120KB.
21. El búfer es suficientemente grande si el número de fotogramas I es 4 o menos. La probabilidad
La capacidad de obtener exactamente *k* fotogramas I es $C(24, k)\, I_{k}\, B_{24-k}$, donde *I* es 0.1 y *B* es
0,9. Las probabilidades de obtener exactamente 0, 1, 2, 3 y 4 I-frames son 0.0798.
0,213, 0,272, 0,221 y 0,129, respectivamente. La suma de estos es 0,915. Esta
significa que hay una probabilidad de falla del 0.085 u 8.5%. Esto es demasiado grande para aceptarlo.
22. El búfer debe ser lo suficientemente grande para contener cinco fotogramas, ya que solo el 5% del
las pistas tienen más de cinco I-frames.
23. Independientemente del formato, el número de flujos simultáneos es 3 × 60 / 15 = 12.
24. Para obtener el punto de reproducción en el medio del búfer, necesitamos poder leer
y almacenar tres flujos a la vez. Cuando la película se reanuda a los 12 minutos, comenzamos

almacenando los streams que están actualmente en 15 min y 20 min. Después de 3 minutos, hemos almacenado de 15 a 18 min y de 20 a 23 min. En ese punto dejamos lo privado stream y comience a mostrar desde el búfer. Después de 2 minutos adicionales, tienen entre 15 y 25 minutos almacenados y el punto de reproducción es de 17 minutos. En este punto solo cargue el búfer del flujo ahora a los 25 min. En 3 minutos, tenemos entre 15 y 28 minutos en el búfer y el punto de reproducción es de 20 min. Hemos logrado nuestro objetivo. Ser- porque no estamos sincronizados con las transmisiones de video bajo demanda cercanas, esta es la lo mejor que podemos hacer.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 7

35

25. Una alternativa es tener un archivo separado para cada idioma. Esta alternativa minimiza el uso de RAM pero desperdicia grandes cantidades de espacio en disco. Si el espacio en disco es baratos y el objetivo es admitir tantas transmisiones posibles a la vez, esto proach es atractivo. Otra alternativa es almacenar la pista de audio para cada idioma por separado y realice una búsqueda adicional por cuadro para obtener el audio. Esta El esquema hace un uso eficiente del espacio en disco, pero introduce búsquedas adicionales y, por lo tanto, ralentiza el rendimiento.

26. La constante de normalización, C , es 0.36794, por lo que las probabilidades son 0.368, 0.184, 0.123, 0.092, 0.074, 0.061, 0.053 y 0.046.

27. Un disco de 14 GB contiene 14×2^{30} o 15.032.385.536 bytes. Si estos son uniformemente dividido en 1000 cilindros, cada cilindro contiene 15.032.385, que es solo suficiente para un videoclip de 30 segundos. Por tanto, cada clip ocupa un cilindro. los Entonces, la pregunta es qué fracción del peso total está representada por los 10 clips de 1000. Sumando 1, 1/2, ... 1/10, obtenemos 2.92895. Multiplicar esto por 0.134 obtenemos 0.392, por lo que el brazo pasa casi el 40% de su tiempo dentro los 10 cilindros del medio.

28. Para cuatro elementos, la ley de Zipf produce probabilidades de 0.48, 0.24, 0.16 y 0.12. Las razones de estas probabilidades también describen la utilización relativa de las unidades. para la figura 7-0 (a). Para los otros tres arreglos de trazado de bandas, todos los se usa igualmente, asumiendo que todos los que pagan por una película la ven hasta el final. Sin embargo, el resultado en un momento determinado puede ser diferente. Si todo el mundo en la ciudad quiere empezar a ver una película a las 8 AM del La disposición de la figura 7-0 (b) golpearía inicialmente con más fuerza el primer disco, luego el siguiente disco 15 minutos después, etc. Los arreglos de la Fig. 7-0 (c) o (d) no ser afectado de esta manera.

29. PAL se ejecuta a 25 fotogramas por segundo, por lo que los dos usuarios tienen una diferencia de 150 fotogramas. Fusionar ellos en 3 min significa cerrar la brecha en 50 fotogramas / min. Uno va 25 cuadros / min más rápido y uno va 25 cuadros / min más lento. La velocidad de fotogramas normal es 1500 fotogramas / min, por lo que la velocidad hacia arriba o hacia abajo es 25/1500 o 1/60, que es aproximadamente 1,67%.

30. Para NTSC, con 30 fotogramas / seg, una ronda es 33,3 mseg. El disco gira 180 veces / seg, por lo que la latencia de rotación promedio es media rotación o 2,8 mseg. MPEG-2 se ejecuta a unos 500.000 bytes / seg. O unos 16.667 bytes / cuadro. A 320 MB / seg, el tiempo de transferencia de una trama es de aproximadamente 51 microsegundos. Así la búsqueda,

la latencia de rotación y los tiempos de transferencia suman aproximadamente 5,8 mseg. Cinco corrientes
por lo tanto, consuma 29 mseg de los 33,3 mseg, que es el máximo.
31. El tiempo de búsqueda promedio va de 3.0 mseg a 2.4 mseg, por lo que el tiempo por operación
ción se reduce a 5,2 mseg. Esto agrega una corriente más, lo que hace seis en total.

36
SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 7
32. Las solicitudes se ordenan, con base en el cilindro solicitado, como: (40, 210), (32, 300), (34, 310), (36, 500). La hora de finalización de cada solicitud es, en orden: 21 ms, 27 mseg, 33 mseg, 39 mseg. Por tanto, el sistema no cumplirá la última fecha límite, si se utiliza el algoritmo anterior.
33. Seis corrientes. Las bandas son inútiles. Cada operación de disco todavía tarda 5,2 ms en ponga el brazo sobre los datos. Si el tiempo de transferencia es 51 µseg o 13 µseg no hace mucha diferencia.
34. Para el primer lote de cinco solicitudes, la crítica es para el cilindro 676, cuarto en la lista, pero con un plazo de t = 712 mseg. Entonces cada solicitud debe ser servido en 3 mseg o menos para que el cuarto se haga en t = 712 mseg.
SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 8
1. Tanto USENET como SETI @ home podrían describirse como distribuidos en un área amplia sistemas. Sin embargo, USENET es en realidad más primitivo que el esquema de Fig.8-1c, ya que no requiere ninguna infraestructura de red más que conexiones punto a punto entre pares de máquinas. Además, dado que no procesar el trabajo más allá de lo necesario para garantizar la adecuada difusión de artículos de noticias, podría debatirse si realmente es un sistema distribuido de el tipo que nos ocupa en este capítulo. SETI @ home es más típico ejemplo de un sistema distribuido de área amplia; los datos se distribuyen a nodos remotos que luego devuelve los resultados de los cálculos al nodo coordinador.
2. Dependiendo de los detalles de cómo las CPU están conectadas a la memoria, una de ellas pasa primero, por ejemplo, toma el autobús primero. Completa su memoria operación, luego ocurre la otra. No es predecible cual va primero, pero si el sistema ha sido diseñado para una coherencia secuencial, debe no importa.
3. Una máquina de 200 MIPS emitirá 200 millones de referencias de memoria / seg, consumiendo 200 millones de ciclos de bus o la mitad de la capacidad del bus. Solo se necesitan dos CPU para consumir todo el bus. El almacenamiento en caché reduce el número de misiones / seg a 20 millones, lo que permite que 20 CPU compartan el bus. Para obtener 32 CPU en el bus, cada uno podía solicitar no más de 12,5 millones de solicitudes / seg. Si sólo 12,5 millones de los 200 millones de referencias de memoria salen en el bus, la tasa de errores de caché debe ser 12,5 / 200 o 6,25%. Esto significa la tasa de aciertos es 93,75%.
4. Las CPU 000, 010, 100 y 110 se cortan de las memorias 010 y 011.
5. Cada CPU gestiona completamente sus propias señales. Si se genera una señal desde el teclado y el teclado no están asignados a ninguna CPU en particular (el caso habitual), de alguna manera, la señal debe enviarse a la CPU correcta para que la maneje.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 8
37
6. Aquí hay una posible solución:
enterregion:

```
BLOQUEO TST
| Prueba el valor de bloqueo
ENTERREGIÓN DE JNE
| Si no es cero, inténtalo de nuevo.
REGISTRO TSL, BLOQUEO
| Copie el bloqueo para registrarse y configure el bloqueo en 1
REGISTRO CMP, # 0
| ¿Fue cerradura cero?
ENTERREGIÓN DE JNE
| Si era distinto de cero, se estableció el bloqueo, así que bucle
RETIRADO
| Volver a la persona que llama; región crítica ingresada
```

7. Como se indica en el texto, tenemos poca experiencia (y herramientas) para escribir mucho aplicaciones de escritorio paralelas. Aunque las aplicaciones de escritorio a veces multiproceso, los subprocesos se utilizan a menudo para simplificar la programación de E / S y por lo tanto, no son subprocesos de procesamiento intensivo. La única aplicación de escritorio El área que tiene alguna posibilidad de paralelización a gran escala son los videojuegos, donde muchos aspectos del juego requieren un cálculo significativo (paralelo). UNA Un enfoque más prometedor es paralelizar el sistema operativo y los servicios de biblioteca. vicios. Ya hemos visto ejemplos de esto en hardware y sistemas operativos actuales. señales. Por ejemplo, las tarjetas de red ahora tienen procesadores paralelos integrados (procesadores de red) que se utilizan para acelerar el procesamiento de paquetes y ofrecer servicios de red de nivel superior a velocidades de línea (p. ej., cifrado, detección de intrusiones ción, etc.). Como otro ejemplo, considere los potentes procesadores que se encuentran en tarjetas de video utilizadas para descargar el procesamiento de video de la CPU principal y ofrecer API de gráficos de nivel superior para aplicaciones (por ejemplo, Open GL). Uno puede imaginar reemplazando estas tarjetas de propósito especial con procesadores multinúcleo de un solo chip. Además, a medida que aumenta el número de núcleos, se puede aplicar el mismo enfoque básico se utiliza para paralelizar otros sistemas operativos y servicios bibliotecarios comunes.

8. Probablemente los bloqueos en las estructuras de datos sean suficientes. Es difcil imaginar algo fragmento de código podría hacer eso es crítico y no involucra algunos datos del kernel estructura. Toda adquisición y liberación de recursos utiliza estructuras de datos, por ejemplo amplio. Si bien no se puede probar, es muy probable que bloquee la estructura de datos las tarifas son suficientes.

9. Se necesitan 16 ciclos de bus para mover el bloque y funciona en ambos sentidos para cada TSL . Así, cada 50 ciclos de bus, 32 de ellos se desperdician moviendo el bloque de caché. En consecuencia, el 64% del ancho de banda del bus se desperdicia en transferencias de caché.

10. Sí, lo sería, pero el tiempo entre encuestas podría terminar siendo muy largo, degradante actuación. Pero sería correcto, incluso sin un máximo.

11. Es tan bueno como TSL . Se utiliza cargando previamente un 1 en el registro para usado. Entonces ese registro y la palabra de memoria se intercambian atómicamente. Después la instrucción, la palabra de memoria está bloqueada (es decir, tiene un valor de 1). Su anterior Nuestro valor ahora está contenido en el registro. Si estaba bloqueado previamente, el

la palabra no se ha cambiado y la persona que llama debe repetirse. Si fue previamente desbloqueado, ahora está bloqueado.

12. El bucle consta de una instrucción TSL (5 nsec), un ciclo de bus (10 nsec) y un JMP de nuevo a la instrucción TSL (5 nsec). Por lo tanto, en 20 nseg, 1 ciclo de bus se cuestionado ocupando 10 nseg. El bucle consume el 50% del bus.

13. *A* es el proceso que acaba de seleccionar. Bien puede haber otros en la misma CPU.

14. La programación de afinidad tiene que ver con colocar el hilo correcto en la CPU correcta. Hacerlo bien podría reducir las pérdidas de TLB, ya que estas se guardan dentro de cada UPC. Por otro lado, no tiene ningún efecto sobre las fallas de página, ya que si una página está en memoria para una CPU, está en memoria para todas las CPU.
15. (a) 2 (b) 4 (c) 8 (d) 5 (e) 3 (f) 4.
16. En una cuadrícula, el peor de los casos son los nodos en esquinas opuestas que intentan comunicarse. Sin embargo, con un toro, las esquinas opuestas están a solo dos saltos de distancia. Lo peor El caso es una esquina tratando de hablar con un nodo en el medio. Para k impar , se necesita $(k - 1) / 2$ saltos para ir de una esquina a la mitad horizontalmente y otra $(k - 1) / 2$ saltos para ir al centro verticalmente, para un total de $k - 1$. Para k pares , el medio es un cuadrado de cuatro puntos en el medio, por lo que el peor de los casos es de un esquina al punto más distante en ese cuadrado de cuatro puntos. Se necesitan $k / 2$ saltos para obtener allí horizontalmente y también $k / 2$ verticalmente, por lo que el diámetro es k .
17. La red se puede cortar en dos mediante un plano que pasa por el medio, lo que da dos sistemas, cada uno con una geometría de $8 \times 8 \times 4$. Hay 64 enlaces que se ejecutan entre entre las dos mitades, para un ancho de banda de bisección de 64 Gbps.
18. Si solo consideramos el tiempo de la red, obtenemos 1 nseg por bit o un retraso de 512 nsec por paquete. Para copiar 64 bytes 4 bytes a la vez, se necesitan 320 nseg en cada lateral, o 640 nseg en total. Sumando el tiempo de conexión de 512 nsec, obtenemos 1132 nsec total. Si se necesitan dos copias adicionales, obtenemos 1792 nsec.
19. Si consideramos solo el tiempo de conexión, una red de 1 Gbps entrega 125 MB / seg. Mover 64 bytes en 1152 nsec es 55,6 MB / seg. Mover 64 bytes en 1792 nsec es 35,7 MB / seg.
20. En una máquina de memoria compartida, basta con pasar el puntero al mensaje desde la CPU que ejecuta el envío a la CPU que ejecuta la recepción , con pos-
Traducciones posibles entre direcciones de memoria física y virtual. En un multicomputadora sin memoria compartida, una dirección en la dirección de una CPU espacio no tiene significado para otra CPU, por lo que el contenido real del búfer de envío er tienen que ser transmitidos como paquetes y luego reensamblados en el búfer del proceso de recepción. Para el programador, los procesos parecen idénticos, pero el el tiempo requerido será mucho mayor en el multicomputador.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 8
39
21. El tiempo para mover k bytes por E / S programadas es $20\ k$ nseg. El tiempo de DMA es $2000 + 5\ k$ nseg. Igualando estos y resolviendo para k obtenemos el punto de equilibrio apunte a 133 bytes.
22. Claramente, sucede algo incorrecto si una llamada al sistema se ejecuta de forma remota. Tratar-
La lectura de un archivo en la máquina remota no funcionará si el archivo no está allí. Además, configurar una alarma en la máquina remota no enviará una señal a la máquina que llama. Una forma de manejar las llamadas al sistema remoto es atraparlas y enviarlos de vuelta al sitio de origen para su ejecución.
23. Primero, en una red de transmisión, se podría realizar una solicitud de transmisión. Segundo, un base de datos centralizada de quién tiene qué página se podría mantener. En tercer lugar, cada La página podría tener una base de operaciones, indicada por los k bits superiores de su anuncio virtual. vestido; la base de operaciones podría realizar un seguimiento de la ubicación de cada una de sus páginas.
24. En esta división, el nodo 1 tiene A , E y G , el nodo 2 tiene B y F , y el nodo 3 tiene C , D , H y yo . El corte entre los nodos 1 y 2 ahora contiene AB y EB para un peso de 5. El corte entre los nodos 2 y 3 ahora contiene CD , CI , FI y FH para un peso de 14. El corte entre los nodos 1 y 3 ahora contiene EH y

GH para un peso de 8. La suma es 27.
25. La tabla de archivos abiertos se mantiene en el kernel, por lo que si un proceso tiene archivos abiertos,
cuando se descongela e intenta usar uno de sus archivos, el nuevo kernel no
saber sobre ellos. Un segundo problema es la máscara de señal, que también se almacena
en el kernel original. Un tercer problema es que si hay una alarma pendiente,
se apaga en la máquina equivocada. En general, el kernel está lleno de fragmentos
de información sobre el proceso, y deben migrarse correctamente como
bien.
26. Las máquinas virtuales no tienen nada que ver con las particiones de disco. El hipervisor puede
tomar una sola partición de disco y dividirla en subparticiones y dar a cada
máquina virtual uno de ellos. En principio, puede haber cientos de ellos. Eso
puede dividir estáticamente el disco en *n* piezas o hacerlo a pedido como
se solicitan bloques.
27. Las tablas de páginas solo pueden ser modificadas por el sistema operativo invitado, no por la aplicación.
programas de cationes en el huésped. Cuando el sistema operativo invitado termine de modificar el
tablas, debe volver al modo de usuario emitiendo una instrucción sensible como
REGRESO DE LA TRAMPA. Esta instrucción atrapará y dará al hipervisor
controlar. A continuación, podría examinar las tablas de páginas en el sistema operativo invitado para ver si
había sido modificado. Si bien esto podría funcionar, todas las tablas de páginas deberían
ser verificado en cada sistema creado por una aplicación de invitado, es decir, cada vez
el sistema operativo invitado volvió al modo de usuario. Podría haber miles de estos transi-
ciones por segundo, por lo que no es probable que sea tan eficiente como usar páginas de solo lectura
para la tabla de páginas.

---

28. Podría traducir todo el programa por adelantado. La razón para no hacer eso
es que muchos programas tienen grandes fragmentos de código que nunca se ejecutan. Por
traduciendo bloques básicos bajo demanda, nunca se traduce ningún código no utilizado. UNA
La desventaja potencial de la traducción bajo demanda es que puede ser ligeramente
menos eficiente para seguir iniciando y deteniendo el traductor, pero este efecto es
probablemente pequeño.
29. Sí, por supuesto. Linux se ha paravirtualizado precisamente porque la fuente
el código está disponible. Windows no se ha paravirtualizado porque la fuente
el código no está disponible.
30. No, esas diferencias pueden significar que trasladar el hipervisor de la plataforma a
plataforma requiere un poco de ajuste, pero la máquina emulada es la misma en
todos los casos, por lo que el dispositivo virtual debería funcionar en todas partes.
31. Los nodos Ethernet deben poder detectar colisiones entre paquetes, por lo que
El retraso de la paginación entre los dos nodos más separados debe ser menor
que la duración del paquete más corto a enviar. De lo contrario, el remitente puede
transmitir completamente un paquete y no detectar una colisión aunque el paquete
sufre una colisión cerca del otro extremo del cable.
32. No solo la máquina necesita memoria para el sistema operativo normal (invitado)
tem y todas sus aplicaciones, pero también necesita memoria para la función de hipervisor
ciones y estructuras de datos necesarias para ejecutar instrucciones sensibles en nombre de
el SO invitado. Los hipervisores de tipo 2 tienen el costo adicional del host operativo
sistema. Además, cada máquina virtual tendrá su propio sistema operativo,

por lo que habrá N copias del sistema operativo almacenadas en la memoria. Una forma de recuperar
duce el uso de memoria sería identificar "código compartido" y sólo mantener uno
copia de este código en la memoria. Por ejemplo, una empresa de alojamiento web puede ejecutar
múltiples VM, cada una con una versión idéntica del sistema operativo Linux
código tem y una copia idéntica del código del servidor web Apache. En este caso,
el segmento de código se puede compartir entre VM, aunque las regiones de datos
debe ser privado.
33. El middleware se ejecuta en diferentes sistemas operativos, por lo que el código es claramente
diferente porque las llamadas al sistema integrado son diferentes. Lo que tienen en
common está produciendo una interfaz común para la capa de aplicación anterior
ellos. Si la capa de aplicación solo realiza llamadas a la capa de Middleware y
sin llamadas al sistema, entonces todas las versiones pueden tener el mismo código fuente. Si
también hacen verdaderas llamadas al sistema, estas serán diferentes.
34. Los servicios más adecuados son
(a) Conexión no confiable.
(b) Flujo de bytes confiable.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 8
41
35. Se mantiene jerárquicamente. Hay un servidor mundial para *.edu* que
conoce todas las universidades y un servidor *.com* que conoce todas las
nombres que terminan en *.com* . Por lo tanto, para buscar *cs.uni.edu* , una máquina buscaría primero
hasta *uni* en la *.edu* servidor, y luego ir allí para preguntar por *cs* , y así sucesivamente.
36. Una computadora puede tener muchos procesos esperando conexiones entrantes.
Estos pueden ser el servidor web, el servidor de correo, el servidor de noticias y otros. Algunos
Se necesita una forma para que sea posible dirigir una conexión entrante a algunos
proceso particular. Eso se hace haciendo que cada proceso escuche un
Puerto. Se ha acordado que los servidores web escucharán el puerto 80, por lo que
las próximas conexiones dirigidas al servidor web se envían al puerto 80. El número
ber en sí fue una elección arbitraria, pero hubo que elegir algunos números.
37. Los dispositivos de E / S físicos aún presentan problemas porque no migran con
la máquina virtual, sin embargo, sus registros pueden mantener un estado que es crítico para la
buen funcionamiento del sistema. Piense en operaciones de lectura o escritura en dispositivos
(por ejemplo, el disco) que se han emitido pero aún no se han completado. Red
La E / S es particularmente difícil porque otras máquinas continuarán enviando
paquetes al hipervisor, sin saber que la máquina virtual se ha movido. Incluso
si los paquetes se pueden redirigir al nuevo hipervisor, la máquina virtual se
no responde durante el período de migración, que puede ser largo porque la entrada
Tire de la máquina virtual, incluido el sistema operativo invitado y todos los procesos.
ejecutar en él debe trasladarse a la nueva máquina. Como resultado, los paquetes pueden
experimenta grandes retrasos o incluso pérdida de paquetes si el dispositivo / hipervisor almacena en
búfer
Desbordamiento.
38. Pueden. Por ejemplo, *www.intel.com* no dice nada sobre dónde está el servidor
es.
39. Una forma sería que el servidor web empaquetara toda la página, incluida la
todas las imágenes, en un archivo zip grande y enviarlo todo la primera vez para que
solo se necesita una conexión. Una segunda forma sería utilizar una conexión
Protocolo sin función como UDP. Esto eliminaría la sobrecarga de conexión,
pero requeriría que los servidores y navegadores realicen su propio control de errores.
40. Tener el valor de una lectura depende de si un proceso está en el

La misma máquina que el último escritor no es en absoluto transparente. Esto aboga por hacer
los cambios solo son visibles para el proceso que realiza los cambios. En el otro
Por otro lado, tener un solo administrador de caché por máquina es más fácil y económico de
plemento. Tal gerente se vuelve mucho más complicado si tiene que
mantener múltiples copias de cada archivo modificado, con el valor devuelto de
pendiente de quién está haciendo la lectura.
41. La memoria compartida funciona con páginas completas. Esto puede llevar a un intercambio falso, en
qué acceso a variables no relacionadas que se encuentran en la misma página
Causa paliza. Poner cada variable en una página separada es un desperdicio. Transmisión exterior-

42
SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 8
El acceso basado en objetos elimina estos problemas y permite un grano más fino de
En g.
42. Hash en cualquiera de los campos de la tupla cuando se inserta en la tupla
el espacio no ayuda porque el *de* puede tener parámetros sobre todo formales. Uno
optimización que trabaja siempre está observando que todos los campos de la *salida* y *en*
se escriben. Por lo tanto, se conoce la firma de tipo de todas las tuplas en el espacio de tuplas,
y el tipo tupla necesaria en una *en* también se conoce. Esto sugiere crear un
subespacio de tupla para cada tipo de firma. Por ejemplo, todos los (int, int, int)
las tuplas van a un espacio, y todas las tuplas (string, int, float) van a una
espacio diferente. Cuando se ejecuta un *in* , solo el subespacio coincidente debe ser
buscado.
SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 9
1. ha llegado el momento de la morsa dijo que se hablara de muchas cosas
de barcos y zapatos y lacre de coles y reyes
de por qué el mar está hirviendo y si los cerdos tienen alas
pero espera un poco las ostras lloraron antes de que tengamos nuestra charla
porque algunos de nosotros estamos sin aliento y todos estamos gordos
sin prisa dijo el carpintero le agradecieron mucho por eso
Desde *A través del espejo* (Tweedledum y Tweedledee).
2. La restricción es que no hay dos celdas que contengan las mismas dos letras, de lo contrario
el descifrado sería ambiguo. Por tanto, cada uno de los 676 elementos de la matriz
contiene uno diferente de los 676 digramas. El número de combinaciones diferentes
ciones es así 676! Este es un número muy grande.
3. El remitente elige una clave aleatoria y la envía al cifrado de terceros de confianza.
ted con la clave secreta que comparten. Luego, el tercero de confianza descifra
la clave aleatoria y la vuelve a cifrar con la clave secreta que comparte con el receptor.
Luego, este mensaje se envía al receptor.
4. Una función como $y = x^k$ es fácil de calcular, pero tomar la k -ésima raíz de y está lejos
más difícil.
5. *A* y *B* eligen claves aleatorias *Ka* y *Kb* y los envían a *C* cifra con *C* s'
Llave pública. *C* elige una clave aleatoria *K* y la envía a *A* cifrada usando *Ka* y
a *B* cifrado mediante *Kb* .
6. Matriz de protección total: 1000 × 100 = 100.000 unidades de espacio.
ACL:
10 × 100 × 2 (1% de objetos accesibles en todos los dominios; 100 entradas / ACL)
+ 100 × 2 × 2 (10% de objetos accesibles en dos dominios; dos entradas / ACL)

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 9

43

+ 890 × 1 × 2 (89% de objetos accesibles en un dominio; una entrada / ACL)
= 4,180 unidades de espacio

El espacio necesario para almacenar una lista de capacidades será el mismo que para
ACL.

7. Para que todo el mundo pueda leer un archivo *excepto* una persona, acceda a las listas de control son la única posibilidad. Para compartir archivos privados, listas de control de acceso o capacidades bilidades se pueden utilizar. Para hacer públicos los archivos, las listas de control de acceso son más fáciles, pero

También puede ser posible poner una capacidad para el archivo o archivos en un conocido
colocar en un sistema de capacidad.

8. Aquí está la matriz de protección:

Objeto
Dominio
Notas PPP
prog1
proyecto.t
splash.gif
asw
Leer
Leer
Ejecutivo
Leer
Escribir
Leer
Escribir
gmw
Leer
Escribir
Leer
Escribir
usuarios
Leer
Leer
Escribir
desarrollar
Leer
Leer
Ejecutivo

9. Las ACL son las siguientes:

Expediente
ACL
Notas PPP
gmw: RW; *: R
prog1
asw: RWX; desarrollo: RX; *: R
proyecto.t
asw: RW; usuarios: RW
splash.gif
asw: RW; desarrollo: R

Suponga que * significa todo.

10. El servidor verificará que la capacidad sea válida y luego generará una
capacidad. Esto es legal. Después de todo, el amigo puede simplemente regalar la capacidad
ya lo ha hecho. Darle el poder de regalar algo aún más débil no es
una amenaza a la seguridad. Si tiene la capacidad de regalar, digamos, poder de lectura / escritura, regalar energía de solo lectura no es un problema.

44

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 9

11. No. Eso sería escribir, lo que viola la propiedad * .

12. Un proceso que escribe en otro proceso es similar a un proceso que escribe en un archivo. En consecuencia, la propiedad * debería mantenerse. Un proceso podría escribir pero no anote. Proceso *B* podría enviar a *C* , *D* , y *E* , pero no a *una* .

13. En la foto original, los ejes R, G y B permiten valores integrales discretos de 0 a 255, inclusive. Esto significa que hay 2^{24} puntos válidos en color. espacio que puede ocupar un píxel. Cuando se quita 1 bit para el canal encubierto nel, solo se permiten los valores pares (asumiendo que el bit secreto se reemplaza por un 0 en todas partes). Por lo tanto, se cubre la mayor parte del espacio, pero la resolución de color es solo la mitad de bueno. En total, solo se puede representar 1/8 de los colores. los los colores no permitidos se asignan al color adyacente, todos cuyos valores son números pares, por ejemplo, los colores (201, 43, 97), (201, 42, 97), (200, 43, 96), y (200, 42, 97) ahora todos se asignan al punto (200, 42, 96) y no pueden ya se distingue.

14. La imagen contiene 1.920.000 píxeles. Cada píxel tiene 3 bits que se pueden usar, dada una capacidad bruta de 720.000 bytes. Si esto se duplica efectivamente debido a comprimiendo el texto antes de almacenarlo, la imagen puede contener texto ASCII ocupado- aproximadamente 1.440.000 bytes antes de la compresión. Por tanto, una sola imagen puede contener el valor de un disquete completo de datos ASCII. No hay expansión debido a la esteganografía. La imagen con los datos ocultos es del mismo tamaño que el original. imagen nal. La eficiencia es del 25%. Esto puede verse fácilmente por el hecho de que 1 bit de cada muestra de color de 8 bits contiene carga útil y la compresión comprime dos bits de texto ASCII por bit de carga útil. Por tanto, por píxel de 24 bits, Se están codificando efectivamente 6 bits de texto ASCII.

15. Los disidentes podrían firmar los mensajes con una clave privada y luego intentar dar a conocer ampliamente su clave pública. Esto podría ser posible si alguien sacarlo de contrabando fuera del país y luego publicarlo en Internet desde un país libre tratar.

16. (a) Ambos archivos tienen 2,25 MB.
(b) *Hamlet* , *Julio César* , *Rey Lear* , *Macbeth* y *Mercader de Venecia* .
(c) Hay seis archivos de texto almacenados en secreto, con un total de aproximadamente 722 KB.

17. Depende de la longitud de la contraseña. El alfabeto del cual las contraseñas Está construido tiene 62 símbolos. El espacio de búsqueda total es $62^5 + 62^6 + 62^7 + 62^8$, que es aproximadamente 2×10^{14} . Si se sabe que la contraseña tiene *k* caracteres, el espacio de búsqueda se reduce a solo 62^k . La proporción de éstos es por lo tanto $2 \times 10^{14} / 62^k$. Para *k* de 5 a 8, estos valores son 242,235, 3907, 63 y 1. En otras palabras, saber que la contraseña tiene solo cinco caracteres reduce el espacio de búsqueda al un factor de 242,235 porque no es necesario probar todas las contraseñas largas. Esta es una gran victoria. Sin embargo, saber que son ocho caracteres no ayuda mucho porque significa que se pueden omitir todas las contraseñas cortas (fáciles).

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 9

45

18. Trate de calmar al asistente. El algoritmo de cifrado de contraseña es público. Pasar- Las palabras son encriptadas por el programa de *inicio de sesión* tan pronto como se escriben, y la contraseña cifrada se compara con la entrada en el archivo de contraseña.

19. No, no es así. El estudiante puede averiguar fácilmente cuál es el número aleatorio

su superusuario es. Esta información se encuentra en el archivo de contraseña sin cifrar. Si esto es 0003, por ejemplo, luego intenta encriptar contraseñas potenciales como *Susan0003* , *Boston0003* , *IBMPC0003* , etc. Si otro usuario ha pasado palabra *Boston0004* , sin embargo, no la descubrirá.

20. Un mecanismo de cifrado requiere una forma de obtener el texto original del texto cifrado mediante un algoritmo de descifrado y una clave. El mecanismo UNIX hace uso de una función unidireccional que no se puede invertir.

21. Suponga que hay *m* usuarios en los sistemas. El cracker puede recoger el *m* valores de sal, asumidos todos diferentes aquí. El cracker tendrá que intentar cifrar ing cada contraseña adivinada *m* veces, una vez con cada una de las *m* sales utilizadas por el sistema. Por lo tanto, el tiempo del cracker para descifrar todas las contraseñas se multiplica por *m* .

22. Hay muchos criterios. Éstos son algunos de ellos:
Debe ser fácil e indoloro de medir (no muestras de sangre).
Debe haber muchos valores disponibles (no el color de ojos)
La característica no debe cambiar con el tiempo (no el color del cabello).
Debería ser difícil falsificar la característica (no el peso)

23. Si se puede confiar en todas las máquinas, funciona bien. Si no se puede confiar en alguno, el esquema se rompe, porque una máquina no confiable podría enviar un mensaje sabio a una máquina confiable pidiéndole que ejecute algún comando en nombre del superusuario. La máquina que recibe el mensaje no tiene forma de saber si el comando realmente se originó con el superusuario o con un estudiante.

24. Ambos utilizan funciones unidireccionales. UNIX almacena todas las contraseñas en el archivo de contraseñas está cifrado y el esquema de Lamport utiliza funciones unidireccionales para generar una secuencia de contraseñas.

25. No funcionaría utilizarlos hacia adelante. Si un intruso capturara a uno, saber cuál usar la próxima vez. Usarlos al revés evita este peligro.

26. No, no es factible. El problema es que no se comprueban los límites de la matriz. Las matrices no se alinean con los límites de la página, por lo que la MMU no es de ninguna ayuda. Además, hacer una llamada al kernel para cambiar la MMU en cada procedimiento la llamada sería prohibitivamente cara.

27. El compilador podría insertar código en todas las referencias de matriz para verificar los límites. Esta característica evitaría los ataques de desbordamiento del búfer. No se hace porque ralentizaría significativamente todos los programas. Además, en C no es ilegal para declarar una matriz de tamaño 1 como parámetro de procedimiento y luego hacer referencia

---

elemento 20, pero claramente la matriz real cuya dirección se ha pasado había mejor tener al menos 20 elementos.

28. Si las capacidades se utilizan para hacer posible tener una pequeña protección, red, no; de lo contrario, sí. Si un editor, por ejemplo, se inicia con solo el capacidades para que el archivo se edite y su archivo borrador, entonces no importa qué Hay trucos al acecho dentro del editor, todo lo que puede hacer es leer esos dos archivos. Sobre el Por otro lado, si el editor puede acceder a todos los objetos del usuario, Trojan los caballos pueden hacer su trabajo sucio, con capacidad o no.

29. Desde el punto de vista de la seguridad, sería ideal. Los bloques usados a veces son expuestos, filtrando información valiosa. Desde el punto de vista del desempeño, Poner a cero los bloques desperdicia tiempo de CPU, degradando así el rendimiento.

30. Para cualquier sistema operativo, todos los programas deben comenzar a ejecutarse en un dirección o tener una dirección inicial almacenada en una posición conocida en el programa encabezado del archivo. (a) El virus copia primero las instrucciones en la dirección de inicio normal.

vestido o la dirección en el encabezado a un lugar seguro, y luego inserta un salto a él-
self en el código o su propia dirección de inicio en el encabezado. (b) Cuando termine con
su propio trabajo, el virus ejecuta las instrucciones que tomó prestadas, seguidas de un
saltar a la siguiente instrucción que se habría ejecutado, o transferencias
arrastre a la dirección que encontró en el encabezado original.
31. Un registro de arranque maestro requiere solo un sector, y si el resto de la primera pista
es gratuito, proporciona espacio donde un virus puede ocultar el sector de arranque original como
así como una parte sustancial de su propio código. Los controladores de disco modernos leen y
almacena en búfer pistas enteras a la vez, por lo que no habrá retardos ni sonidos perceptibles
de búsquedas adicionales a medida que se leen los datos adicionales.
32. Los programas C tienen la extensión *.c* . En lugar de usar la llamada al sistema de acceso para probar
para obtener permiso de ejecución, examine el nombre del archivo para ver si termina en *.c* . Esta
el código lo hará

```
char * nombre de archivo;
int len;
nombre de archivo = dp-> dname;
len = strlen (nombre de archivo);
if (strcmp (& nombre de archivo [len - 2], ".c") == 0) infectar (s);
```

33. Probablemente no puedan decirlo, pero pueden adivinar que XORing una palabra dentro
el virus con el resto producirá un código de máquina válido. Sus computadoras pueden
solo pruebe cada palabra de virus a su vez y vea si alguna de ellas produce una máquina válida
código. Para ralentizar este proceso, Virgil puede utilizar un mejor algoritmo de cifrado.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 9

47

ritmo, como usar diferentes claves para las palabras pares e impares, y luego rotar
ing la primera palabra dejada por una cierta cantidad de bits determinada por una función hash
en las teclas, rotando la segunda palabra ese número de bits más uno, y así sucesivamente.
34. El compresor es necesario para comprimir otros programas ejecutables como parte de
el proceso de infectarlos.
35. La mayoría de los virus no quieren infectar un archivo dos veces. Puede que ni siquiera funcione.
Por lo tanto, es importante poder detectar el virus en un archivo para ver si está
listo infectado. Todas las técnicas utilizadas para dificultar el software antivirus.
para detectar virus también dificulta que el propio virus sepa qué archivos
sido infectado.
36. Primero, ejecutar el programa *fdisk* desde el disco duro es un error. Puede estar en
infectado y puede infectar el sector de arranque. Tiene que ser ejecutado desde el original.
CD-ROM o un disquete protegido contra escritura. En segundo lugar, los archivos restaurados pueden
infectado. Volver a colocarlos sin limpiarlos puede simplemente volver a instalar
virus.
37. Sí, pero el mecanismo es ligeramente diferente al de Windows. En UNIX, una com-
El virus panion se puede instalar en un directorio en la ruta de búsqueda antes del
uno en el que vive el programa real. El ejemplo más común es insertar un
programa *ls* en un directorio de usuario, que anula efectivamente */ bin / ls* porque es
encontrado primero.
38. Un gusano es un programa independiente que funciona solo. Un virus es un código
fragmento que se adjunta a otro programa. El gusano se reproduce haciendo
más copias del programa de gusanos. El virus se reproduce infectando a otros
programas.
39. Obviamente, ejecutar cualquier programa de una fuente desconocida es peligroso.
Los archivos autoextraíbles pueden ser especialmente peligrosos, porque pueden
alquilar varios archivos en varios directorios y el programa de extracción en sí

podría ser un caballo de Troya. Si hay una opción disponible, es mucho mejor obtener archivos en forma de un archivo ordinario, que luego puede extraer con herramientas tú confías.
40. No es posible escribir un programa de este tipo, porque si tal programa es posible sible, un cracker puede utilizar ese programa para eludir la comprobación de virus en el programa cargado de virus que escribe.
41. Se puede inspeccionar la dirección IP de origen de todos los paquetes entrantes. El segundo un conjunto de reglas eliminará todos los paquetes IP entrantes con direcciones IP de origen pertenecientes a spammers conocidos.
42. No importa. Si se usa relleno de cero, entonces S2 debe contener el prefijo verdadero como un entero sin signo en los *k* bits de orden inferior. Si se usa la extensión de signo, entonces S2 También debe estar extendido el letrero. Siempre que S2 contenga los resultados correctos de

cambiar una dirección verdadera, no importa qué hay en los bits superiores no utilizados de S2 .
43. Los navegadores existentes vienen precargados con las claves públicas de varios terceros como Verisign Corporation. Su negocio consiste en veri-
Fingir las claves públicas de otras empresas y crear certificados para ellas.
Estos certificados están firmados, por ejemplo, por la clave privada de Verisign. Ya que La clave pública de Verisign está integrada en el navegador, los certificados firmados con su Se puede verificar la clave vate.
44. Primero, Java no proporciona variables de puntero. Esto limita la capacidad de un proceso para sobrescribir una ubicación de memoria arbitraria. En segundo lugar, Java no permite que los usuarios Asignación de almacenamiento controlada (*malloc / free*). Esto simplifica el manejo de la memoria envejecimiento. En tercer lugar, Java es un lenguaje de tipo seguro, lo que garantiza que se utilice una variable
exactamente de la manera que se supone que debe basarse en su tipo.
45. Aquí están las reglas.

URL	**Firmante**	**Objeto**	**Acción**
www.appletsRus.com	AppletsRus	/ usr / me / appletdir / *	Leer
www.appletsRus.com	AppletsRUs	/ usr / tmp / *	Leer escribir
www.appletsRus.com	AppletsRUs	www.appletsRus; puerto: 5004	Conectar, Leer

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 10
1. Los archivos que se enumerarán son: *bonefish, quacker, seahorse* y *weasel* .
2. Imprime el número de líneas del archivo *xyz* que contienen la cadena " nd " en ellos.
3. La tubería es la siguiente:
cabeza –8 z | cola –1

La primera parte selecciona las primeras ocho líneas de *zy las* pasa a la *cola* ,
que solo escribe el último en la pantalla.
4. Están separados, por lo que la salida estándar se puede redirigir sin afectar
Error estándar. En una tubería, la salida estándar puede pasar a otro proceso, pero
el error estándar todavía se escribe en el terminal.
5. Cada programa se ejecuta en su propio proceso, por lo que se inician seis procesos nuevos.
6. Sí. La memoria del niño es una copia exacta de la de los padres, incluida la pila.
Por lo tanto, si las variables de entorno estaban en la pila de los padres, estarán en
la pila del niño también.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 10
49
7. Dado que los segmentos de texto se comparten, solo se deben copiar 36 KB. La máquina
puede copiar 80 bytes por microsec, por lo que 36 KB tarda 0,46 mseg. Agregar otro 1
mseg para entrar y salir del kernel, y todo toma aproximadamente
1,46 mseg.
8. Linux se basa en la copia en escritura. Le da al niño consejos a los padres
espacio de vestimenta, pero marca las páginas de los padres como protegidas contra escritura. Cuando el niño en-
intenta escribir en el espacio de direcciones de los padres, ocurre una falla y una copia de
la página de los padres se crea y se asigna al espacio de direcciones del niño.
9. Sí. Ya no puede funcionar, por lo que cuanto antes su memoria vuelva a estar libre
lista, mejor.
10. Los usuarios malintencionados podrían causar estragos en el sistema si pudieran enviar señales
a procesos arbitrarios no relacionados. Nada impediría que un usuario escribiera un
programa que consta de un bucle que envía una señal al proceso con PID *i* para todos
i de 1 al máximo PID. Muchos de estos procesos no estarían preparados
por la señal y sería asesinado por ella. Si quiere acabar con su propio proceso
esses, eso está bien, pero matar los procesos de su vecino no es un ac-
Aceptable.
11. Sería imposible usar Linux o Windows Vista, pero el Pentium difícilmente
Ware hace esto posible. Lo que se necesita es utilizar la función de segmentación.
las características del hardware, que no son compatibles con Linux o Windows
Vista. El sistema operativo podría colocarse en uno o más segmentos globales,
con llamadas a procedimientos protegidos para realizar llamadas al sistema en lugar de trampas. OS / 2
funciona de esta manera.
12. Generalmente, los demonios se ejecutan en segundo plano haciendo cosas como imprimir y
envío de correo electrónico. Dado que la gente no suele sentarse en el borde de sus sillas
esperando a que terminen, se les da baja prioridad, absorbiendo el exceso de CPU
tiempo que no necesitan los procesos interactivos.
13. Un PID debe ser único. Tarde o temprano, el mostrador se envolverá y se irá
volverá a 0. Entonces será tan ascendente a, por ejemplo, 15. Si sucede que
el proceso 15 se inició hace meses, pero aún se está ejecutando, no se puede asignar 15
a un nuevo proceso. Por lo tanto, después de elegir un PID propuesto utilizando el contador,
Se debe realizar una búsqueda en la tabla de procesos para ver si el PID ya está en uso.
14. Cuando el proceso finaliza, al padre se le dará el estado de salida de su hijo.
El PID es necesario para poder identificar al padre para que el estado de salida pueda ser
transferido al proceso correcto.
15. Si se establecen todos los bits de las *banderas de intercambio* , la llamada de clonación inicia una
hilo. Si se borran todos los bits, 1 la llamada es esencialmente una bifurcación .

50

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 10

16. Cada decisión de programación requiere buscar un mapa de bits para la matriz activa.
y buscando el primer bit establecido en la matriz, que se puede hacer en constante
tiempo, retirando una sola tarea de la cola seleccionada, nuevamente un tiempo constante
operación, o si el valor del mapa de bits es cero, intercambiando los valores del activo y
listas caducadas, nuevamente una operación de tiempo constante.

17. El programa cargado desde el bloque 0 tiene una longitud máxima de 512 bytes, por lo que puede-
no será muy complicado. La carga del sistema operativo requiere comprensión
el diseño del sistema de archivos para encontrar y cargar el sistema operativo. Dif-
diferentes sistemas operativos tienen sistemas de archivos muy diferentes; esta preguntando tambien
es de esperar que un programa de 512 bytes solucione todo esto. En cambio, el bloque 0
loader simplemente recupera otro cargador de una ubicación fija en la partición del disco.
Este programa puede ser mucho más largo y específico del sistema para que pueda encontrar y
cargar el sistema operativo.

18. Con texto compartido, se necesitan 100 KB para el texto. Cada uno de los tres procesos
necesita 80 KB para su segmento de datos y 10 KB para su pila, por lo que la memoria total
necesario es 370 KB. Sin texto compartido, cada programa necesita 190 KB, por lo que tres
de ellos necesitan un total de 570 KB.

19. Los procesos que comparten un archivo, incluida la posición actual del puntero del archivo, pueden
compartir un descriptor de archivo abierto, sin tener que actualizar nada en cada
otras tablas de descriptores de archivos privados. Al mismo tiempo, otro proceso puede
acceder al mismo archivo a través de un descriptor de archivo abierto separado, obtener una
puntero de archivo y moverse a través del archivo a su propia voluntad.

20. El segmento de texto no puede cambiar, por lo que nunca se debe paginar. Si sus marcos
son necesarios, simplemente pueden abandonarse. Las páginas siempre se pueden recuperar
desde el sistema de archivos. El segmento de datos no debe volver a paginarse al ex-
ecutable, porque es probable que haya cambiado desde que se incorporó.
Paginarlo arruinaría el archivo ejecutable. El segmento de la pila ni siquiera
presente en el archivo ejecutable.

21. Dos procesos podrían asignar el mismo archivo a sus espacios de direcciones en el mismo
hora. Esto les da una forma de compartir la memoria física. La mitad de lo compartido
La memoria podría usarse como un búfer de *A* a *B* y la mitad como un búfer de *B* a
Una . Para comunicarse, un proceso escribe un mensaje en su parte del
memoria, luego una señal al otro para indicar que hay un mensaje en espera
para ello. La respuesta podría usar el otro búfer.

22. La dirección de memoria 65,536 es el byte de archivo 0, por lo que la dirección de memoria 72,000
es el byte de archivo
6464.

23. Originalmente, se asignaron cuatro páginas del archivo: 0, 1, 2 y 3. La llamada
tiene éxito y, una vez hecho, solo las páginas 2 y 3 siguen mapeadas, es decir,
bytes 16,384 aunque 32,767

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 10

51

24. Es posible. Por ejemplo, cuando la pila crece más allá de la página inferior, un
se produce una falla de página y el sistema operativo normalmente asigna la siguiente
página. Sin embargo, si la pila se ha topado con el segmento de datos, el siguiente
La página no se puede asignar a la pila, por lo que el proceso debe terminarse antes
porque se ha quedado sin espacio de direcciones virtuales. Además, incluso si hay otro

página disponible en la memoria virtual, el área de paginación del disco puede estar llena, imposibilitando la asignación de un almacén de respaldo para la nueva página, lo que también terminar el proceso.

25. Es posible si los dos bloques no son amigos. Considere la situación de Figura 10-17 (e). Llegan dos nuevas solicitudes de ocho páginas cada una. En este punto las 32 páginas inferiores de la memoria son propiedad de cuatro usuarios diferentes, cada uno con ocho páginas. Ahora los usuarios 1 y 2 liberan sus páginas, pero los usuarios 0 y 3 retienen suyo. Esto produce una situación en la que se utilizan ocho páginas, ocho páginas libres, ocho páginas libres y ocho páginas utilizadas. Tenemos dos bloques adyacentes de igual tamaño que no se pueden fusionar porque no son amigos.

26. Pagar a una partición permite el uso de un dispositivo sin procesar, sin la sobrecarga de utilizando estructuras de datos del sistema de archivos. Para acceder al bloque *n* , el sistema operativo puede calcular su posición de disco simplemente agregándolo al bloque inicial de la partición. No es necesario pasar por todos los bloques indirectos que de otro modo sabio ser necesario.

27. Abrir un archivo por una ruta relativa al directorio de trabajo suele ser más complicado. conveniente para el programador o usuario, ya que se necesita un nombre de ruta más corto. Es también suele ser mucho más sencillo y requiere menos accesos al disco.

28. Los resultados son los siguientes:

(a) Se concede el bloqueo.

(b) Se concede el bloqueo.

(c) *C* está bloqueado, ya que los bytes 20 a 30 no están disponibles.

(d) *A* está bloqueado, ya que los bytes 20 a 25 no están disponibles.

(e) *B* está bloqueado, ya que el byte 8 no está disponible para bloqueo exclusivo.

En este punto ahora tenemos un punto muerto. Ninguno de los procesos podrá para correr de nuevo.

29. Surge la cuestión de qué proceso obtiene el bloqueo cuando está disponible. La solución más sencilla es dejarlo sin definir. Esto es lo que POSIX hace porque es el más fácil de implementar. Otro es exigir que las cerraduras estén concedidas en el orden en que fueron solicitadas. Este enfoque es más trabajo para el implementación, pero evita el hambre. Otra posibilidad más es dejar

Los procesos proporcionan una prioridad al solicitar un bloqueo, y utilizan estas prioridades para tomar una decisión.

---

30. Un enfoque es dar un error y negarse a realizar la búsqueda . Otro es hacer que la compensación sea negativa. Mientras no se use, no hay daño hecho. Solo si se intenta leer o escribir el archivo, debe aparecer un mensaje de error. se le dé salvia. Si la búsqueda es seguida por otra búsqueda que hace el desplazamiento positivo, no se da ningún error.

31. El propietario puede leerlo, escribirlo y ejecutarlo, y todos los demás (incluido el propietario) puede leerlo y ejecutarlo, pero no escribirlo.

32. Sí. Cualquier dispositivo de bloque capaz de leer y escribir un bloque arbitrario puede utilizarse para contener un sistema de archivos. Incluso si no hubiera forma de buscar un bloque, siempre es posible rebobinar la cinta y luego contar hacia adelante hasta el bloque solicitado. Un sistema de archivos de este tipo no sería un sistema de archivos de alto rendimiento.

tem, pero funcionaría. El autor ha hecho esto en un PDP-11 usando DECtapes y funciona.

33. No. El archivo todavía tiene un solo propietario. Si, por ejemplo, solo el propietario puede escribir en el archivo, la otra parte no puede hacerlo. Vincular un archivo a su directorio La historia no le otorga de repente ningún derecho que no tuviera antes. Es solo crea una nueva ruta para acceder al archivo.
34. Cuando se cambia el directorio de trabajo, usando la llamada al sistema chdir , el i-El nodo para el nuevo directorio de trabajo se recupera y se guarda en la memoria, en el i-tabla de nodos. El i-nodo para el directorio raíz también está ahí. En la estructura de usuario tura, se mantienen los indicadores a ambos. Cuando un nombre de ruta tiene que ser analizado, se inspecciona el primer carácter. Si es una " / ", el puntero a la raíz i-nodo se utiliza como el lugar de partida, de lo contrario, el puntero a la dirección de trabajo Se utiliza el i-nodo de tory.
35. El acceso al i-nodo del directorio raíz no requiere acceso al disco, por lo que tener lo siguiente:
1. Leer el directorio / para buscar "usr".
2. Lectura en el i-nodo para */ usr* .
3. Leyendo el directorio */ usr* para buscar " ast ".
4. Lectura en el i-nodo para */ usr / ast* .
5. Leer el directorio */ usr / ast* para buscar "trabajo".
6. Leer en el i-nodo para */ usr / ast / work* .
7. Leer el directorio */ usr / ast / work* para buscar " f ".
8. Lectura en el i-nodo para */ usr / ast / work / f* .

Así, en total, se necesitan ocho accesos al disco para ftech el i-nodo. memoria.
36. El i-nodo tiene 12 direcciones. El único bloque indirecto contiene 256. El doble El bloque indirecto ble conduce a 65.536, y el triple indirecto conduce a 16.777.216, para un total de 16.843.018 bloques. Esto limita el tamaño máximo de archivo a 12 + 256 + 65,536 + 16,777,218 bloques, que son aproximadamente 16 gigabytes.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 10
53
37. Cuando se cierra un archivo, el contador de su i-nodo en la memoria disminuye. Si es mayor que cero, el i-nodo no se puede eliminar de la tabla porque el archivo todavía está abierto en algún proceso. Solo cuando el contador llegue a cero i-node sea eliminado. Sin el recuento de referencias, el sistema no sabría cuándo eliminar el i-nodo de la tabla. Haciendo una copia separada del i-nodo cada vez que se abría el archivo no funcionaría porque los cambios realizados en una copia no sería visible en las demás.
38. Al mantener las colas de ejecución por CPU, se pueden tomar decisiones de programación localmente, sin ejecutar costosos mecanismos de sincronización para siempre acceder y actualizar una cola de ejecución compartida. Además, es más probable que todos los Las páginas de memoria seguirán estando en la caché1 si programamos un hilo en el mismo CPU donde ya se ejecutó.
39. Al forzar el contenido del archivo modificado en el disco cada 30 segundos, el daño causado por un choque está limitado a 30 segundos. Si *pdflush* no se ejecutó, un proceso podría escribir un archivo, luego salir con el contenido completo del archivo aún en la caché. De hecho, el usuario puede cerrar la sesión y volver a casa con el archivo todavía en el cache. Una hora más tarde, el sistema podría bloquearse y perder el archivo, todavía solo en el caché y no en el disco. Al día siguiente no tendríamos un usuario feliz.
40. Todo lo que tiene que hacer es establecer el recuento de enlaces en 1, ya que solo una entrada de directorio hace referencia
ences el i-nodo.
41. Generalmente es getpid , getuid , getgid o algo así. Todo lo que hacen es

buscar un entero de un lugar conocido y devolverlo. Cada otra llamada lo hace
más.
42. El archivo simplemente se elimina. Esta es la forma normal (en realidad, la única) de
eliminar un archivo.
43. Un disquete de 1,44 MB puede contener 1440 bloques de datos sin procesar. El bloque de arranque
superbloque, bloque descriptor de grupo, mapa de bits de bloque y mapa de bits de i-nodo de un
El sistema de archivos ext2 usa cada uno un bloque. Si se crean 8192 i-nodos de 128 bytes,
estos i-nodos ocuparían otros 1024 bloques, dejando solo 411 bloques
no usado. Se necesita al menos un bloque para el directorio raíz, dejando espacio para
410 bloques de datos de archivo. En realidad, el programa Linux *mkfs* es lo suficientemente inteligente
como para no
para hacer más i-nodos de los que posiblemente se puedan usar, por lo que la ineficiencia no es esta
malo. De forma predeterminada, se crearán 184 i-nodos que ocupan 23 bloques. Sin embargo,
Debido a la sobrecarga del sistema de archivos ext2, Linux normalmente usa el
Sistema de archivos MINIX 1 en disquetes y otros dispositivos pequeños.
44. A menudo es esencial tener a alguien que pueda hacer cosas que normalmente son recomendables.
licitado. Por ejemplo, un usuario inicia un trabajo que genera una cantidad infinita
de salida. Luego, el usuario cierra la sesión y se va de vacaciones de tres semanas a Lon-
don. Tarde o temprano el disco se llenará y el superusuario tendrá que fabricar
Ally mata el proceso y elimina el archivo de salida. Existen otros ejemplos similares.

---

45. Probablemente alguien tenía el archivo abierto cuando el profesor cambió el permiso
siones. El profesor debería haber eliminado el archivo y luego poner otra copia
en el directorio público. Además, debería utilizar un método mejor para distribuir
archivos, como una página web, pero eso está más allá del alcance de este ejercicio.
46. Si, por ejemplo, se otorgan derechos de superusuario a otro usuario con la llamada al sistema fsuid ,
ese usuario puede acceder a los archivos de superusuario, pero no podrá enviar señales, elimine
procesos o realizar otras operaciones que requieran privilegios de superusuario.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 11

1. Una base de datos genealógica puede resultar conveniente para registrar el nacimiento y la muerte
fechas de los antepasados utilizando el formato de hora estándar del sistema. De hecho, cualquier
La base de datos histórica podría usar esto.
2. Si las manijas contenían un número de secuencia, entonces cuando se cerró una manija y
luego continuó utilizándose, esto podría detectarse fácilmente comparando los
secuencia en cada asa a un número de secuencia en la tabla de asa. Cuarto para
el número de secuencia debería encontrarse tanto en el asa como en la tabla
entrada. Cada vez que se reutiliza una entrada de la tabla de identificadores, el número de secuencia es
incrementa, y el número de secuencia está incrustado en el identificador. Si estos-
número de secuencia es N bits, entonces un identificador particular tendría que cerrarse y
reabierto 2^N veces para evitar la detección. Así que incluso para una pequeña N mucha manija
Las condiciones de carrera pueden detectarse en los programas.
3. Una señal es manejada por un nuevo hilo en el contexto de algún proceso. Por ejemplo,
cuando se pulsa la tecla Salir o incluso cuando falla un hilo. Realmente no hace
cualquier sentido para captar una señal en el contexto de un hilo. Realmente tiene que ser por proceso
ess. Por lo tanto, el manejo de señales es realmente una actividad por proceso.
4. Tendría más sentido en los servidores. Las máquinas cliente tienen menos concurrentes
Procesos. Las bibliotecas compartidas solo tienen sentido si hay varios procesos
compartirlos. De lo contrario, es más eficiente vincular estáticamente las bibliotecas y
aceptar duplicación. La ventaja del enlace estático es que solo aquellos procedimientos

Se cargan los duros que realmente se necesitan. Con los archivos DLL puede haber
duros en la memoria que nadie está usando.
5. La razón más importante es que los procesos son multiproceso, por lo que un subproceso
podría modificar la pila del núcleo que pertenece a otro hilo mientras el núcleo
lo estaba usando. Además, el kernel tendría que determinar la parte superior del usuario-
modo pila antes de establecer su propio puntero de pila. Y tendría que haber
una forma de garantizar que haya suficiente espacio en la pila para el kernel
correr. Además, el núcleo a menudo se basa en el hecho de que el núcleo se apila
no se puede paginar. Entonces, la parte superior de la pila del modo de usuario (al menos) permitiría

---

las formas tienen que estar grabadas en la memoria. La ventaja de que el kernel comparte el
pila de modo de usuario es que la pila de kernel no tendría que consumir kernel
direcciones virtuales. Este beneficio podría lograrse asignando un kernel separado
nel stack, pero en modo de usuario en lugar de en modo kernel. Pero esto no funcionaría
en NT porque los subprocesos del kernel pueden adjuntarse a nuevos espacios de direcciones, y
perdería acceso a su pila en modo kernel si estuviera en el modo de usuario
ción del espacio de direcciones.
6. La tasa de aciertos en las entradas de la tabla de páginas almacenadas en caché en el TLB tiene un gran impacto en
rendimiento de sistema. La operación para recorrer las tablas de páginas y encontrar un error
La entrada es muy cara. Dado que el TLB tiene sólo un número limitado de
intentos, el uso de páginas de 4 MB aumenta considerablemente la cantidad de direcciones virtuales que se pueden
mapeado por el TLB a la vez.
7. Hay un límite de 32 operaciones porque solo hay 32 bits de derechos en el
mango de objeto.
8. No es posible porque los semáforos y los mutex son objetos ejecutivos y
las secciones críticas no lo son. Se administran principalmente en el espacio del usuario (pero tienen
un semáforo de respaldo cuando se necesita bloquear). El administrador de objetos no
los conocen y no tienen tiradores, como se indica en el texto.
Dado que WaitForMultipleObjects es una llamada al sistema, el sistema no puede realizar una
OR booleano de varias cosas, una de las cuales no sabe nada. La llamada
debe ser una llamada al sistema porque los semáforos y las exclusiones mutuas son objetos del núcleo. En
En resumen, no es posible tener ninguna llamada al sistema que mezcle objetos del kernel y
objetos de usuario como este. Tiene que ser uno o el otro.
9. (a) El último hilo sale.
(b) Un hilo ejecuta ExitProcess .
(c) Otro proceso con un mango para este lo mata.
10. Dado que las identificaciones se reutilizan de inmediato, un programa que identificó un proceso
que quería operar por ID podría encontrar que el proceso había muerto, y
la identificación se ha reutilizado entre el momento en que la encuentra y cuando la usa.
UNIX no reutiliza los ID de proceso hasta que se hayan utilizado todos los demás (32 000) ID.
Entonces, aunque teóricamente podría ocurrir el mismo problema, las probabilidades son muy bajas.
Windows evita este problema manteniendo la lista libre en orden FIFO, de modo que
Normalmente, los ID no se reutilizan durante mucho tiempo. Otra solución podría tener
sido agregar números de secuencia, como se sugirió para los identificadores de objetos para resolver un
problema similar con la reutilización de mangos ordinarios. En general, las aplicaciones deben
no identificar un proceso en particular solo por el ID, sino también por los tiempos de creación-

apisonar. Para realizar una operación en un proceso, la ID se usa para obtener un identificador. los
La marca de tiempo de creación se puede verificar una vez que se obtiene el identificador.

---

11. Como máximo unos pocos microsegundos. Se adelanta al hilo actual inmediatamente. Eso es solo una cuestión de cuánto tiempo lleva ejecutar el código del despachador para hacer el interruptor de hilo.

12. Que su prioridad se reduzca por debajo de la prioridad base podría usarse como un problema por utilizar un tiempo de CPU excesivo u otros recursos.

13. Para los subprocesos en modo kernel, sus pilas y la mayoría de las estructuras de datos que necesidad de acceder a permanecer accesible incluso después de cambiar los espacios de direcciones, porque
se comparten en todos los procesos. Si los hilos del modo de usuario cambiaran de anuncio espacios de vestuario, perderían el acceso a sus pilas y otros datos. Encontrar un solución para el modo de usuario, como el cambio de pilas durante una llamada de proceso cruzado, permitiría a los subprocesos crear procesos como límites de aislamiento y pasar el CPU entre ellos tanto como la CPU pasa del modo de usuario al kernel modo. Se pueden realizar llamadas a procedimientos entre procesos sin involucrar al programador para suspender a la persona que llama y despertar a la persona que llama en el proceso del servidor (y
viceversa al regresar).

14. Aunque los conjuntos de trabajo no se están recortando, los procesos aún se están modificando páginas respaldadas por archivos. Estos archivos se enviarán periódicamente al disco. también Las aplicaciones realizan descargas explícitas de páginas al disco para preservar los resultados y el mantenimiento.
mantener la coherencia en los archivos. Tanto el sistema de archivos como el registro están activos todos los
tiempo e intente asegurarse de que los volúmenes y las colmenas no pierdan datos si los datos iban a estrellarse.

15. El automapa se realiza tomando una de las entradas en el directorio de páginas y, en lugar de que apunte a una página de tabla de páginas, se apunta a sí mismo. Esta entrada es configurado en el directorio de la página para un proceso cuando se inicializa por primera vez. Desde el siempre se utiliza la misma entrada PDE, la dirección virtual del automapa será la lo mismo en todos los procesos.

16. Sí. Los VAD son la forma en que el administrador de memoria realiza un seguimiento de qué los vestidos están en uso y son gratuitos. Se necesita un VAD para una región reservada para evitar que un intento posterior de reservarlo o cometerlo tenga éxito.

17. (1) es una decisión política sobre cuándo y cómo recortar un conjunto de trabajo. (2) y (3) son requeridos. (4) es una decisión política sobre cuán agresivamente escribir sucio páginas al disco. (5) y (6) son obligatorios. (7) no es realmente una cuestión de política o requerido; el sistema nunca tiene que poner cero páginas, pero si el sistema es de otra manera inactivas, poner a cero las páginas siempre es mejor que simplemente ejecutar el ciclo inactivo.

18. No se mueve en absoluto. Una página solo entra en una de las listas cuando no está presente en cualquier conjunto de trabajo. Si todavía está en un conjunto de trabajo, no continúa cualquiera de las listas gratuitas.

---

19. No puede ir a la lista modificada, ya que contiene páginas que aún están en el mapa.

ped en y podría ser devuelto. Una página sin asignar no está en esa categoría.
Ciertamente no puede ir directamente a la lista libre porque esas páginas pueden ser abandonadas.
hecho a voluntad. Una página sucia no se puede abandonar a voluntad. En consecuencia,
primero debe escribirse de nuevo en el disco, luego puede ir a la lista libre.
20. Un evento de notificación significa que todos los hilos en espera serán ejecutables.
cuando se señala el evento. Esto podría resultar en muchos cambios de contexto adicionales
ya que los hilos intentan adquirir el evento solo para volver a bloquearse. En un solo proceso
Essor, si el tiempo de retención de bloqueo es mucho más corto que el cuanto, entonces es probable
que para cuando los otros subprocesos se ejecuten, el bloqueo se habrá liberado
por el hilo anterior para adquirirlo. Pero con tiempos de espera cortos, la probabilidad
de muchos subprocesos esperando el bloqueo es mucho menor de todos modos. Con un
multiprocesador, incluso tiempos de espera cortos pueden resultar en un contexto innecesario
cambia porque los subprocesos que se desbloquean pueden estar todos programados para
se ejecutan inmediatamente en diferentes procesadores pero se bloquean inmediatamente, ya que solo
uno
hilo logrará adquirir el bloqueo.
21. Hay dos registros. Los campos son los siguientes. Los valores antes de los dos puntos
son los campos de encabezado:
Registro 1 = 0, 8: (3, 50), (1, 22), (3, 24), (2, 53)
Registro 2 = 10, 10: (1, 60)
22. El hecho de que el bloque 66 sea contiguo a un recorrido existente no ayuda, ya que el
los bloques no están en el orden lógico de los archivos. En otras palabras, usar el bloque 66 como nuevo
El bloque no es mejor que usar el bloque 90. Las entradas en el MFT son:
0, 8: (4, 20), (2, 64), (3, 80), (1, 66)
23. Es un accidente. Los 16 bloques aparentemente comprimidos en 8 bloques. Podria
Han tenido 9 u 11 años con la misma facilidad.
24. Todos, excepto el SID del usuario, se pueden eliminar sin afectar la fuerza de
la seguridad.
25. No. Cuando se carga una DLL, puede ejecutar código dentro del proceso. Este codigo
puede acceder a toda la memoria del proceso y puede utilizar cualquiera de los identificadores, y
todas las credenciales en el token predeterminado del proceso. La única cosa impersona-
Lo que hace es permitir que el proceso adquiera nuevos identificadores utilizando las credenciales de
el cliente que está personificando. Permitir que se ejecute CUALQUIER código que no sea de confianza
dentro del proceso significa que cualquiera de las credenciales accesibles a ese proceso
ahora o en el futuro se pueden utilizar incorrectamente. Esto es algo a considerar la próxima vez
utiliza cualquier navegador configurado para descargar código de sitios que no son de confianza.
26. El programador siempre intentará poner el hilo en su procesador ideal, o
en caso contrario, vuelva a un procesador en el mismo nodo. Por lo tanto, incluso si el hilo es
actualmente se ejecuta en un procesador en un nodo diferente, al final de la

pagefault, de hecho, puede programarse nuevamente en su nodo preferido. Me caigo
Los procesadores de ese nodo continúan ocupados, es posible que se ejecute en un nodo diferente.
y pagar la multa por un acceso más lento a la página que acaba de recibir un error. Pero
es más probable que vuelva a su nodo preferido tarde o temprano, y
por lo tanto, aproveche la ubicación de esta página a largo plazo.
27. Las operaciones de metadatos en el sistema de archivos, como cambiar archivos de recuperación,
Cree momentos inoportunos en los que los bloqueos no darían lugar a ningún archivo de recuperación.
Tener la oportunidad de vaciar los búferes de datos internos permite que la aplicación
crear un estado coherente del que sea más fácil recuperarse. Permitiendo que la aplicación

ción para terminar operaciones complejas simplificaría la cantidad de lógica necesaria
al recuperarse.
En general, permitir que una aplicación se prepare para la instantánea del
ume le permite reducir la cantidad de estado transitorio que debe tratarse
cuando se recupere más tarde. La recuperación de fallos es generalmente difícil de probar porque
Es tan difícil evaluar todas las posibles combinaciones de condiciones de carrera que son
posible. Teóricamente es posible diseñar un software que pueda hacer frente a todos
tipo de escenarios de recuperación y falla, pero es muy difícil. Un ejemplo de
El software que enfrenta estos desafíos es el propio sistema de archivos NTFS. Un montón de
Se han invertido esfuerzos de diseño, análisis y prueba para que sea muy similar
que NTFS puede recuperarse de un bloqueo sin tener que ejecutar *chkdsk* (el
Equivalente de ventanas UNIX 's *fsck*). En sistemas de archivos empresariales muy grandes,
chkdsk para recuperarse después de un bloqueo puede llevar horas o días.
28. La última página de un archivo mapeado en memoria debe contener ceros después de la última
datos. De lo contrario, si un controlador de E / S para un disco solo sobrescribió parte de la página
con datos del disco, los datos anteriores en el resto de la página serían
expuesto. Esto no es un problema para las páginas anteriores del archivo porque el sistema
leerá una página completa del archivo antes de dar acceso al modo de usuario
código. Cuando las páginas de la pila se asignan a medida que la pila crece, estas deben contener
todos ceros, no cualquier dato aleatorio que pudiera haber dejado allí el
inquilino.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 12

1. La consideración aquí debe ser cuánto tiempo pasa la operación en el
kernel, es decir, usando datos del kernel y llamando a operaciones del lado del kernel, y si un
Se trata de un recurso compartido que requiere protección.
(a) La programación es típicamente un servicio de kernel pesado, sin importar qué tipo de kernel
nel estás corriendo. Esto se consideraría una operación de espacio de kernel.
(b) La impresión no depende del kernel. Si bien hay alguna E / S de dispositivo,
la mayoría de las operaciones de impresión están en la generación de gráficos para imprimir.
Esta es principalmente una acción de espacio de usuario.

(c) La recepción y respuesta a una consulta de descubrimiento de Bluetooth implica una gran cantidad de
E / S del dispositivo. Además, hay mucho que esperar y escuchar las solicitudes.
Esto se puede hacer en un servidor del sistema o en el propio kernel. La espera
involucrado lo convertiría en una operación basada en servidor para la mayoría de microkernel
plataformas.
(d) La pantalla es un recurso que requiere mucha administración y protección.
La mayoría de las actividades de gestión / protección las realizan servidores en micro-
granos. La visualización generalmente se realiza de esta manera.
(e) Esta operación tiene dos partes: detectar un mensaje que llega y reproducir un
sonido. Ambas partes involucran piezas de kernel, pero también involucran compartidas
recursos. Dado que se dedica mucho tiempo a la espera, ambas piezas quisieran:
Se colocará en un servidor.
(f) La interrupción de la ejecución es una operación pesada del kernel, que involucra a muchos
elementos de datos y llamadas al sistema. Esto ocurriría en el kernel.

2. Los micronúcleos tienen muchos ahorros de eficiencia. Suelen cargarse más rápido al arrancar
hora. Por lo general, ocupan menos memoria que sus contrapartes más grandes. Ellos
también tienden a ser flexibles, manteniendo un mínimo de operaciones de espacio de kernel en el kernel
nel. Esto también hace que el grano sea más ligero.

3. Los microkernels también sufren problemas de eficiencia. Utilizan mensaje
pasaje mucho más que otros diseños de kernel y por lo tanto invocan importantes
gastos generales para las operaciones del sistema. La funcionalidad del espacio de usuario también requiere
visitas al kernel para hacer las cosas; esto podría ser más eficiente para ser
implementado completamente en el kernel. Los micronúcleos requieren un control estricto sobre el
funciones que quedan en el kernel, por lo que el rendimiento se puede optimizar.
4. El enfoque del nanokernel está en la implementación que se puede hacer con poco
datos y espera mínima. La memoria dinámica requiere mucho registro tabular
datos para realizar un seguimiento de qué proceso / hilo tiene qué bloque de memoria asignado.
Además, dado que la memoria es un recurso compartido, la asignación de memoria podría
implica esperar para usar la memoria.
5. Una aplicación ciertamente podría ser multiproceso y cada uno de los subprocesos podría
trabajar con recursos que requirieron espera ocupada. Entonces, usando objetos activos para
varios subprocesos en una aplicación tendrían sentido. Dado que todos los objetos activos
se combinan para formar un solo hilo, la operación despertaría este hilo y
permitir que el hilo actúe en un evento a la vez.
6. La gente de Symbian cree sin duda que este tipo de seguridad es suficiente. los
La pregunta debe centrarse en esto: ¿hay alguna forma de que una solicitud firmada
permitir la instalación de otra aplicación sin firmar? En otras palabras,
¿Podría una aplicación permitir que otra aplicación se cargue y se ejecute? Otro
no se pudo instalar la aplicación, porque solo el proceso de instalación ha
misión para acceder a las partes del sistema de archivos involucradas en la instalación
(¿recuerdas la jaula de datos?), pero todavía hay áreas problemáticas. Por ejemplo,

---

uno podría imaginarse un programa Java que podría venderse a sí mismo como inocente,
pero descargue clases de Java que podrían realizar operaciones ilegales. Sin embargo,
Los dispositivos Symbian OS solo son compatibles con J2ME y esta plataforma hace muy poco en
la computadora host. Por lo tanto, se cree que el método de instalación actual
od de seguridad es suficiente.
7. Los servidores pueden descargar operaciones complicadas del kernel a la programación de espacio de usuario.
gramos. Esto tiene muchas ventajas. Los servidores pueden cargar y ejecutar solo cuando
son necesarios; en una arquitectura de kernel diferente, el código comparable
formas presentes en el kernel. Los servidores pueden minimizar la cantidad de tiempo
la operación ular pasa en modo kernel; otras arquitecturas incorporarían
el código del servidor en el kernel, pasando así todo el tiempo de implementación en el
núcleo. Finalmente, existe la posibilidad de una mayor concurrencia en el servicio de solicitud
vicio: porque varios servidores (que son procesos en sí mismos) están sirviendo
solicitudes, puede haber más actividad. En contraste con el servicio dentro
otras arquitecturas de kernel; no importa cuántos subprocesos del kernel estén asignados,
los servidores de microkernel representan más subprocesos posibles que se pueden ejecutar.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 13

1. Las mejoras en el hardware de las computadoras se han debido en gran medida a transiciones más pequeñas.
tors. Algunos factores que pueden limitar esto son: (a) las propiedades ondulatorias de la luz pueden
limitar las técnicas fotolitográficas convencionales para producir circuitos integrados
cuits, (b) la movilidad de átomos individuales en sólidos puede conducir a la degradación de
las propiedades de capas muy delgadas de semiconductores, aislantes y conductores

tors, y (c) la radiactividad de fondo puede romper los enlaces moleculares o afectar cargas almacenadas muy pequeñas. Ciertamente hay otros.

2. Para programas altamente interactivos, el modelo de eventos puede ser mejor. De estos, solo (b) es interactivo. Por tanto (a) y (c) son algorítmicos y (b) es evento impulsado.

3. Ponerlo allí ahorró algo de RAM y redujo el tiempo de carga a 0, pero la mayoría importante, facilitó a los desarrolladores de software de terceros el uso de la GUI, asegurando así una uniformidad de apariencia en todo el software.

4. Posiblemente la estadística sea redundante. Podría lograrse mediante una combinación de apertura , fstat y cierre . Sería muy difícil simular cualquiera de los demás.

5. Es posible. Lo que se necesita es un proceso a nivel de usuario, el servidor semáforo que gerentes semáforos. Para crear un semáforo, un usuario le envía un mensaje pidiendo un nuevo semáforo. Para usarlo, el proceso de usuario pasa la identidad de el semáforo a otros procesos. Luego pueden enviar mensajes a la sema-

---

phore server solicitando una operación. Si la operación se bloquea, no se envía respuesta hacia atrás, bloqueando así a la persona que llama.

6. El patrón es de 8 mseg de código de usuario, luego 2 mseg de código de sistema. Con el optimización, cada ciclo es ahora de 8 mseg de código de usuario y 1 mseg de sistema código. Por tanto, el ciclo se reduce de 10 mseg a 9 mseg. Multiplicar por 1000 ciclos de este tipo, un programa de 10 segundos ahora tarda 9 segundos.

7. El mecanismo de venta a los clientes es un edificio con estantes, empleados para almacenar los estantes, cajeros para manejar el pago, etc. La política es qué tipo de productos que vende la tienda.

8. Los nombres externos pueden ser tan largos como sea necesario y de longitud variable. Nombres internos
son generalmente de 32 bits o 64 bits y siempre de longitud fija. Los nombres externos necesitan no ser único. Dos nombres pueden apuntar al mismo objeto, por ejemplo, enlaces en el sistema de archivos UNIX . Los nombres internos deben ser únicos. Los nombres externos pueden ser jerárquico. Los nombres internos son generalmente índices en tablas y, por lo tanto, forman un espacio de nombre plano.

9. Si la nueva mesa es 2 veces más grande que la anterior, no se llenará rápidamente, reduciendo la cantidad de veces que se necesitará una tabla mejorada. Por otro lado, entonces es posible que no se necesite mucho espacio, por lo que puede desperdiciar memoria. Este es un clasico compensación de tiempo versus espacio.

10. Sería arriesgado hacer eso. Suponga que el PID está en la última entrada. En ese caso, salir del bucle dejaría *p* apuntando a la última entrada. Howev-er, si no se encontró el PID, *p* podría terminar apuntando a la última entrada o al uno más allá, dependiendo de los detalles del código compilado, que optimiza zations están activados, y así sucesivamente. Lo que podría funcionar con un compilador podría falla con uno diferente. Es mejor poner una bandera.

11. Podría hacerse, pero no sería una buena idea. Un controlador IDE o SCSI es muchas páginas. Tener un código condicional tan largo hace que el código fuente difícil de seguir. Sería mejor poner cada uno en un archivo separado y luego use el *Makefile* para determinar cuál incluir. O al menos,
La compilación adicional se puede utilizar para incluir un archivo de controlador u otro.

12. Sí. Hace que el código sea más lento. Además, más código significa más errores.

13. No fácilmente. Varias invocaciones al mismo tiempo podrían interferir con una otro. Podría ser posible si los datos estáticos estuvieran protegidos por un mutex, pero eso

significaría que una persona que llama a un procedimiento simple podría bloquearse inesperadamente.
14. Sí. El código se replica cada vez que se llama a la macro. Si se llama
muchas veces, el programa será mucho más grande. Este es un tpico de un tiempo-
compensación espacial. Un programa más grande y más rápido en lugar de un programa más pequeño y lento.
gramo. Sin embargo, en un caso extremo, el programa más grande podría no encajar en el
TLB, lo que hace que se agite y, por lo tanto, funcione más lento.

---

15. Comience por EXCLUSIVE ORing los 16 bits inferior y superior de la palabra juntos.
ella para formar un entero de 16 bits, *s* . Para cada bit, hay cuatro casos: 00 (resultados
en un 0), 01 (da como resultado un 1), 10 (da como resultado un 1) y 11 (da como resultado un 0). Así que si
el número de unos en *s* es impar, la paridad es impar; de lo contrario, es uniforme. Hacer una
tabla con 65.536 entradas, cada una de las cuales contiene un byte con el bit de paridad.
La macro se ve así:
#define paridad (w) bits [(w & 0xFFFF) ˆ ((w >> 16) & 0xFFFF)]
16. Sin circunstancias. El valor de color " comprimido " sería tan grande como el
original y, además, podría necesitarse una enorme paleta de colores. No hace
sentido en absoluto.
17. La paleta de colores de 8 bits de ancho contiene 256 entradas de 3 bytes cada una para un total de 768 bytes. El ahorro por píxel es de 2 bytes. Así, con más de 384 píxeles,
GIF gana. Una paleta de colores de 16 bits de ancho contiene 65.536 entradas de 3 bytes
cada uno, por 196,608 bytes. El ahorro aquí es de 1 byte por píxel. Así con más
que 196.608 píxeles, gana la compresión de 16 bits. Suponiendo una proporción de 4: 3, la
El punto de equilibrio es una imagen de 512 × 384 píxeles. Para VGA (640 × 480), 16-
el color de bits requiere menos datos que el color verdadero de 24 bits.
18. Para una ruta que está en la caché de nombre de ruta, no tiene ningún efecto porque el i-nodo está omitido de todos modos. Si no se lee, no importa si ya está en memoria.
ry. Para una ruta que no está en la caché de nombres pero que involucra un i-node anclado, entonces
La fijación ayuda ya que elimina la lectura de un disco.
19. Registrar la fecha de la última modificación, el tamaño y posiblemente un cálculo
una firma como una suma de comprobación o CRC puede ayudar a determinar si ha cambiado
desde la última referencia. Una advertencia: un servidor remoto podría proporcionar información falsa
información sobre un archivo, y la regeneración local de una firma calculada podría ser
necesario.
20. Al archivo se le puede asignar un número de versión o una suma de verificación y esta información
almacenado junto con la sugerencia. Antes de acceder a un archivo remoto, se debe realizar una verificación
hecho para asegurarse de que el nmero de versin o la suma de
archivo de alquiler.
21. Un sistema de archivos normalmente intentará escribir nuevos datos en el disco disponible más cercano.
bloque siguiente al último utilizado. Si se escriben dos archivos simultáneamente
Esto puede resultar en la intercalación de los bloques de datos en el disco, lo que resulta en
ambos archivos están fragmentados y, por lo tanto, son más difíciles de leer. Este efecto puede ser
mejorado almacenando datos en la memoria para maximizar el tamaño de las escrituras, o
escribir en archivos temporales y luego copiar cada salida en un archivo permanente
cuando el programa termina.

SOLUCIONES DE PROBLEMAS PARA EL CAPÍTULO 13

63

22. Brooks hablaba de grandes proyectos en los que la comunicación entre los los programadores ralentizan todo. Ese problema no ocurre con un proyecto unipersonal y para que la productividad pueda ser mayor.

23. Si un programador puede producir 1000 líneas de código por un costo de $ 100,000, una línea de código cuesta $ 100. En el Cap. 11, declaramos que Windows Vista constaba de 70 millones de líneas de código, lo que equivale a $ 7 mil millones. Eso parece horrible lote. Probablemente Microsoft haya logrado mejorar la productividad de los programadores. utilizando mejores herramientas para que un programador pueda producir varios miles de líneas de código por año. Por otro lado, los ingresos anuales de Microsoft rondan los 50 dólares. mil millones, por lo que es posible gastar $ 7 mil millones en Vista.

24. Suponga que la memoria cuesta $ 100 por 64 MB (compare los precios actuales). Entonces, una máquina de gama baja necesita $ 1600 en disco. Si el resto de la PC está $ 500, el costo total es de $ 2100. Esto es demasiado caro para los de gama baja. mercado.

25. Un sistema integrado puede ejecutar uno solo o una pequeña cantidad de programas. Si Todos los programas se pueden mantener cargados en la memoria en todo momento, es posible que no haya necesita un administrador de memoria o un sistema de archivos. Además, los conductores sería necesario solo para algunos dispositivos de E / S, y podría tener más sentido escriba los controladores de E / S como rutinas de biblioteca. Las rutinas de la biblioteca también pueden ser mejores compilados en programas individuales, en lugar de en bibliotecas compartidas, eliminando la necesidad de bibliotecas compartidas. Probablemente se podrían eliminar muchas otras características. inactivo en casos específicos.