MODERNO
OPERANTE
SISTEMAS

SEGUNDA EDICIÓN

SOLUCIONES A PROBLEMAS

ANDREW S. TANENBAUM

Universidad Libre
Ámsterdam, Países Bajos

SALÓN PRENTICE

Río Saddle superior, Nueva Jersey 07458

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 1

1. Un sistema operativo debe proporcionar a los usuarios una máquina extendida (es decir, virtual) y debe administrar los dispositivos de E/S y otros recursos del sistema.
2. La multiprogramación es el cambio rápido de la CPU entre múltiples procesos en la memoria. Se utiliza comúnmente para mantener la CPU ocupada mientras uno o más procesos realizan operaciones de E/S.
3. El spooling de entrada es la técnica de leer trabajos, por ejemplo, desde tarjetas, al disco, de modo que cuando finalicen los procesos que se están ejecutando en ese momento, habrá trabajo esperando a la CPU. El spooling de salida consiste en copiar primero los archivos imprimibles al disco antes de imprimirlos, en lugar de imprimirlos directamente a medida que se genera la salida. El spooling de entrada en una computadora personal no es muy probable, pero el spooling de salida sí lo es.
4. La razón principal de la multiprogramación es darle a la CPU algo que hacer mientras espera que se complete la operación de E/S. Si no hay DMA, la CPU está completamente ocupada realizando operaciones de E/S, por lo que no hay nada que ganar (al menos en términos de utilización de la CPU) con la multiprogramación. No importa cuántas operaciones de E/S realice un programa, la CPU estará ocupada al 100 por ciento. Por supuesto, esto supone que el retraso principal es la espera mientras se copian los datos. Una CPU podría hacer otro trabajo si la operación de E/S fuera lenta por otras razones (por ejemplo, al llegar a una línea serial).
5. Las computadoras de segunda generación no tenían el hardware necesario para proteger el sistema operativo de programas de usuario maliciosos.
6. Sigue vigente. Por ejemplo, Intel fabrica procesadores Pentium I, II y III, y cuatro CPU con distintas características, como velocidad y consumo de energía. Todas estas máquinas son compatibles arquitectónicamente. Se diferencian únicamente en

precio y rendimiento, que es la esencia de la idea de familia.

7. Un 25×La pantalla de texto monocromático de 80 caracteres requiere un búfer de 2000 bytes. El 1024 ×Un mapa de bits de 24 bits y 768 píxeles requiere 2.359.296 bytes. En 1980, estas dos opciones habrían costado 10 y 11.520 dólares, respectivamente. Para conocer los precios actuales, consulte cuánto cuesta la RAM actualmente, probablemente menos de 1 dólar por MB.

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

9. Los sistemas informáticos personales son siempre interactivos, a menudo con un solo usuario. Los sistemas mainframe casi siempre enfatizan el procesamiento por lotes o el uso compartido del tiempo con muchos usuarios. La protección es un problema mucho mayor en los sistemas mainframe, al igual que el uso eficiente de todos los recursos.

10. Cada nanosegundo, una instrucción emerge de la secuencia de comandos. Esto significa que la máquina está ejecutando mil millones de instrucciones por segundo. No importa en absoluto cuántas etapas tenga la secuencia de comandos. Una secuencia de comandos de 10 etapas con 1 nseg por segundo

Además, la etapa ejecutaría mil millones de instrucciones por segundo. Lo único que importa es la frecuencia con la que una instrucción terminada sale del proceso.

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

12. Lógicamente, no importa si el registro límite utiliza una dirección virtual o una dirección física. Sin embargo, el rendimiento de la primera es mejor. Si se utilizan direcciones virtuales, la adición de la dirección virtual y el registro base puede comenzar simultáneamente con la comparación y luego puede ejecutarse en paralelo. Si se utilizan direcciones físicas, la comparación no puede comenzar hasta que se complete la adición, lo que aumenta el tiempo de acceso.

13. Quizás. Si el autor de la llamada recupera el control y sobrescribe inmediatamente los datos, cuando finalmente se produzca la escritura, se escribirán los datos incorrectos. Sin embargo, si el controlador primero copia los datos a un búfer privado antes de regresar, entonces se puede permitir que el autor de la llamada continúe inmediatamente. Otra posibilidad es permitir que el autor de la llamada continúe y darle una señal cuando se pueda reutilizar el búfer, pero esto es complicado y propenso a errores.

14. Una trampa es causada por el programa y es sincrónica con él. Si el programa se ejecuta 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 tiempo no es reproducible.

15. Base = 40 000 y límite = 10 000. Una respuesta de límite = 50 000 es incorrecta para la forma en que se describió el sistema en este libro. Podría haberse implementado de esa manera, pero hacerlo habría requerido esperar hasta que se encontrara la dirección + El cálculo base se completó antes de iniciar la verificación del límite, lo que ralentizó la computadora.

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

17. Al montar un sistema de archivos, los archivos que ya se encuentran en el directorio del punto de montaje quedan inaccesibles, por lo que los puntos de montaje suelen estar vacíos. Sin embargo, un administrador de sistemas 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 esté comprobando o reparando el dispositivo montado.

18. Tenedorpuede fallar si no quedan espacios libres en la tabla de procesos (y posiblemente si no queda memoria o espacio de intercambio).EjecutivoPuede fallar si el nombre de archivo proporcionado no existe o no es un archivo ejecutable válido.Desconectarpuede fallar si el archivo que se va a desvincular no existe o el proceso que lo llama no tiene la autoridad para desvincularlo.

19. Si la llamada falla, por ejemplo porque*fin de semana*es incorrecta, puede devolver -1. También puede fallar porque el disco está lleno y no es posible escribir la cantidad de bytes solicitados. En una terminación correcta, siempre devuelve*nbytes*.

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

21. Los archivos especiales de bloques están compuestos por bloques numerados, cada uno de los cuales puede leerse o escribirse independientemente de los demás. Es posible buscar cualquier bloque y comenzar a leer o escribir. Esto no es posible con los archivos especiales de caracteres.

22. Las llamadas del sistema en realidad no tienen nombres, salvo en el sentido de documentación. Cuando el procedimiento de la biblioteca*leer*Traps al núcleo, coloca el número de la llamada al sistema en un registro o en la pila. Este número se utiliza para indexar en una tabla. En realidad no se utiliza ningún nombre en ninguna parte. Por otro lado, el nombre del procedimiento de la biblioteca es muy importante, ya que es lo que aparece en el programa.

23. Sí puede, especialmente si el núcleo es un sistema de paso de mensajes.

24. En lo que respecta a la lógica del programa, no importa si una llamada a un procedimiento de biblioteca da como resultado una llamada al sistema. Pero si el rendimiento es un problema, si una tarea se puede realizar sin una llamada al sistema, el programa se ejecutará más rápido. Cada llamada al sistema implica un tiempo adicional para cambiar del contexto del usuario al contexto del núcleo. Además, en un sistema

multiusuario, el sistema operativo puede programar otro proceso para que se ejecute cuando se completa una llamada al sistema, lo que ralentiza aún más el progreso en tiempo real de un proceso que realiza la llamada.

25. Varias llamadas UNIX no tienen equivalente en la API Win32:

Enlace: un programa Win32 no puede hacer referencia a un archivo con un nombre alternativo ni verlo en más de un directorio. Además, intentar crear un enlace es una forma conveniente de probar y crear un bloqueo en un archivo.

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

Chmod: Los programadores de Windows deben asumir que cada usuario puede acceder a todos los archivos.

Matar: Los programadores de Windows no pueden matar un programa que funciona mal y que no coopera.

26. Las conversiones son sencillas:

(a) Un microaño es 10-6×365×24×3600 = 31,536 seg.
(b) 1000 metros o 1 km.
(c) Hay 240bytes, que son 1.099.511.627.776 bytes.
(d) Son las 6×1024kilogramo.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 2

1. La transición de bloqueado a en ejecución es concebible. Supongamos que un proceso está bloqueado en la E/S y la E/S finaliza. Si la CPU está inactiva, el proceso podría pasar directamente de bloqueado a en ejecución. La otra transición faltante, de listo a bloqueado, es imposible. Un proceso listo no puede realizar E/S ni nada que pueda bloquearlo. Solo un proceso en ejecución puede bloquearse.

2. Podría tener un registro que contenga un puntero a la entrada actual de la tabla de procesos. Cuando se complete la operación de E/S, la CPU almacenará el estado actual de la máquina en la entrada actual de la tabla de procesos. Luego, irá al vector de interrupción del dispositivo que realiza la interrupción y buscará un puntero a otra entrada de la tabla de procesos (el procedimiento de servicio). Luego, se iniciará este proceso.

3. En general, los lenguajes de alto nivel no permiten el tipo de acceso al hardware de la CPU que se requiere. Por ejemplo, puede ser necesario un controlador de interrupciones para habilitar y deshabilitar el servicio de interrupciones de un dispositivo en particular, o para manipular datos dentro del área de pila de un proceso. Además, las rutinas de servicio de interrupciones deben ejecutarse lo más rápido posible.

4. Existen varias razones para utilizar una pila independiente para el núcleo. Dos de ellas son las siguientes. En primer lugar, no se desea que el sistema operativo se bloquee porque un programa de usuario mal escrito no deja suficiente espacio en la pila. En segundo lugar, si el núcleo deja datos de la pila en el

espacio de memoria de un programa de usuario al regresar de una llamada del sistema, un usuario malintencionado podría utilizar estos datos para obtener información sobre otros procesos.

5. Sería difícil, si no imposible, mantener la coherencia del sistema de archivos. Supongamos que un proceso cliente envía una solicitud al proceso servidor 1 para actualizar un archivo. Este proceso actualiza la entrada de caché en su memoria. Poco después, otro proceso cliente envía una solicitud al servidor 2 para leer ese archivo. Desafortunadamente, si el archivo también está almacenado en caché allí, el servidor 2, en su inocencia, devolverá datos obsoletos. Si el primer proceso escribe el archivo en el disco después de almacenarlo en caché, y el servidor 2 verifica el disco en cada lectura para ver si su copia en caché está actualizada, el sistema puede funcionar, pero son precisamente todos estos accesos al disco los que el sistema de almacenamiento en caché está tratando de evitar.

6. Cuando se detiene un subproceso, tiene valores en los registros. Deben guardarse, al igual que cuando se detiene el proceso, deben guardarse los registros. Los subprocesos de tiempo compartido no son diferentes a los procesos de tiempo compartido, por lo que cada subproceso necesita su propia área de almacenamiento de registros.

7. No. Si un proceso de un solo hilo está bloqueado en el teclado, no puede bifurcarse.

8. Un subproceso de trabajo se bloqueará cuando tenga que leer una página web del disco. Si se utilizan subprocesos de nivel de usuario, esta acción bloqueará todo el proceso, destruyendo el valor del multiproceso. Por lo tanto, es esencial que se utilicen subprocesos del núcleo para permitir que algunos subprocesos se bloqueen sin afectar a los demás.

9. Los subprocesos de un proceso cooperan. No son hostiles entre sí. Si es necesario ceder para el bien de la aplicación, un subproceso cederá. Después de todo, suele ser el mismo programador quien escribe el código para todos ellos.

10. Los subprocesos a nivel de usuario no pueden ser interrumpidos por el reloj a menos que se haya agotado todo el quantum del proceso. Los subprocesos a nivel de núcleo pueden ser interrumpidos individualmente. En este último caso, si un subproceso se ejecuta durante demasiado tiempo, el reloj interrumpirá el proceso actual y, por lo tanto, el subproceso actual. El núcleo es libre de elegir un subproceso diferente del mismo proceso para ejecutarlo a continuación si así lo desea.

11. En el caso de un solo subproceso, los aciertos de caché tardan 15 ms y los errores de caché, 90 ms. El promedio ponderado es 2./3×15 + 1/3×90. Por lo tanto, la solicitud promedio demora 40 ms y el servidor puede procesar 25 por segundo. En el caso de un servidor multiproceso, toda la espera del disco se superpone, por lo que cada solicitud demora 15 ms y el servidor puede procesar 66 2/3 solicitudes por segundo.

12. Sí. Si el servidor está completamente limitado por la CPU, no es necesario tener varios subprocesos. Solo agrega complejidad innecesaria. Como ejemplo, considere un número de asistencia telefónica (como 555-1212) para un área con 1 millón de personas. Si cada registro (nombre, número de teléfono) tiene, digamos, 64 caracteres, la base de datos completa ocupa 64 megabytes y se puede mantener fácilmente en la memoria del servidor para proporcionar una búsqueda rápida.

13. Los punteros son realmente necesarios porque se desconoce el tamaño de la variable global. Puede ser cualquier cosa, desde un carácter hasta una matriz de números de punto flotante. Si se almacenara el valor, habría que indicar el tamaño a *crear*3*global*, lo cual está bien, pero ¿de qué tipo debería ser el segundo parámetro?*colocar*3*global* ser, y qué tipo debe ser el valor de*leer*3*global*¿ser?

14. Podría suceder que el sistema de ejecución esté precisamente en el punto de bloquear o desbloquear un hilo, y esté ocupado manipulando las colas de programación. Este sería un momento muy inoportuno para que el controlador de interrupciones del reloj comience a inspeccionar esas colas para ver si es el momento de realizar el cambio de hilo, ya que podrían estar en un estado inconsistente. Una solución es establecer un indicador cuando se ingresa al sistema de ejecución. El controlador del reloj vería esto y establecería su propio indicador.

Luego, regresa. Cuando el sistema de ejecución termina, verifica el indicador de reloj, ve que se produjo una interrupción de reloj y ejecuta el controlador de reloj.

15. Sí, es posible, pero ineficiente. Un hilo que quiere hacer una llamada al sistema primero establece un temporizador de alarma y luego realiza la llamada. Si la llamada se bloquea, el temporizador devuelve el control al paquete de hilos. Por supuesto, la mayoría de las veces la llamada no se bloqueará y el temporizador debe ser borrado. Por lo tanto, cada llamada al sistema que pueda bloquearse debe ejecutarse como tres llamadas al sistema. Si los temporizadores se apagan prematuramente, pueden surgir todo tipo de problemas. Esta no es una forma atractiva de crear un paquete de hilos.

16. 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 se vuelve listo y está programado. Si utiliza la espera activa, se ejecutará indefinidamente. Con los subprocesos de nivel de usuario, no puede suceder que un subproceso de baja prioridad sea reemplazado repentinamente para permitir la ejecución de un subproceso de alta prioridad. No hay reemplazo. Con los subprocesos de nivel de núcleo, este problema puede surgir.

17. Cada subproceso invoca procedimientos por sí solo, por lo que debe tener su propia pila para las variables locales, direcciones de retorno, etc. Esto es igualmente válido para los subprocesos de nivel de usuario y de nivel de núcleo.

18. Una condición de carrera es una situación en la que dos (o más) procesos están a punto de realizar alguna acción. Dependiendo del momento exacto, uno u otro va primero. Si uno de los procesos va primero, todo funciona, pero si otro va primero, se produce un error fatal.

m

19. Sí. La computadora simulada podría ser

ultiprogramada. mientras se procesa *A* se está ejecutando, lee alguna variable compartida. El tictac del reloj tardío ocurre y el proceso *B* se ejecuta. También lee la misma variable. Luego, suma 1 a la variable. Cuando el proceso *A* se ejecuta, si también agrega uno a la variable, tenemos una condición de carrera.

Por ejemplo, Luego una simu-

20. Sí, todavía funciona, pero todavía hay mucho trabajo esperando, por supuesto.

21. Ciertamente funciona con la programación preventiva. De hecho, fue diseñado para ese caso. Cuando la programación no es preventiva, puede fallar. Considere el caso en el que *doblar* Inicialmente es 0, pero el proceso 1 se ejecuta primero. Simplemente se repetirá eternamente y nunca liberará la CPU.

22. Sí, se puede. La palabra de memoria se utiliza como un indicador, donde 0 significa que nadie está utilizando las variables críticas y 1 significa que alguien las está utilizando. Coloque un 1 en el registro e intercambie la palabra de memoria y el registro. Si el registro contiene un 0 después del intercambio, se ha concedido el acceso. Si contiene un 1, se ha denegado el acceso. Cuando un proceso finaliza, almacena un 0 en el indicador en la memoria.

23. Para realizar una operación de semáforo, el sistema operativo primero desactiva las interrupciones. Luego lee el valor del semáforo. Si está realizando una operación de semáforo,abajoy el semáforo es igual a cero, coloca el proceso que llama en una lista de procesos bloqueados asociados con el semáforo. Si está realizando unaarriba,Debe comprobar si hay algún proceso bloqueado en el semáforo. Si uno o más procesos están bloqueados, uno de ellos se elimina de la lista de procesos bloqueados y se vuelve ejecutable. Cuando se hayan completado todas estas operaciones, se pueden volver a habilitar las interrupciones.

24. Asociados a cada semáforo de conteo hay dos semáforos binarios,*METRO*, utilizado para la exclusión mutua, y*B*, utilizado para bloquear. También asociado con cada semáforo de conteo hay un contador que contiene el número dearribas menos el número deabajo s, y una lista de procesos bloqueados en ese semáforo. Para implementarabajo,Un proceso primero obtiene acceso exclusivo a los semáforos, al contador y a la lista haciendo unaabajoen*METRO*Luego, decrementa el contador. Si es cero o más, simplemente hace unarribaen*METRO*y sale. Si*METRO*es negativo, el proceso se pone en la lista de procesos bloqueados. Luego, unarribase hace en*METRO*y un
abajose hace en*B*para bloquear el proceso. Para implementararriba, fiprimero*METRO*esabajo Se realiza un intento de obtener la exclusión mutua y luego se incrementa el contador. Si es mayor que cero, nadie fue bloqueado, por lo que todo lo que se necesita hacer esarriba *METRO*. Sin embargo, si el contador ahora es negativo o cero, se debe eliminar algún proceso de la lista. Finalmente, unarribase hace en*B*y*METRO*En ese orden.

25. Si el programa opera en fases y ninguno de los procesos puede ingresar a la siguiente fase hasta que ambos hayan terminado con la fase actual, tiene todo el sentido utilizar una barrera.

26. Con la programación por turnos, funciona. Tarde o temprano*yo*se ejecutará y, finalmente, abandonará su región crítica. El punto es que, con la programación

prioritaria, *yo* nunca llega a ejecutarse en absoluto; con el modo round robin, obtiene un intervalo de tiempo normal periódicamente, por lo que tiene la oportunidad de salir de su región crítica.

27. Con los subprocesos del núcleo, un subproceso puede bloquearse en un semáforo y el núcleo puede ejecutar otro subproceso en el mismo proceso. En consecuencia, no hay ningún problema con el uso de semáforos. Con los subprocesos de nivel de usuario, cuando un subproceso se bloquea en un semáforo, el núcleo piensa que todo el proceso está bloqueado y no lo vuelve a ejecutar nunca más. En consecuencia, el proceso falla.

28. Su implementación es muy costosa. Cada vez que cambia cualquier variable que aparece en un predicado del que está esperando algún proceso, el sistema de ejecución debe volver a evaluar el predicado para ver si se puede desbloquear el proceso. Con los monitores Hoare y Brinch Hansen, los procesos solo se pueden reactivar en unseñal primitivo.

29. Los empleados se comunican entre sí mediante mensajes: pedidos, comida y bolsas en este caso. En términos UNIX, los cuatro procesos están conectados por tuberías.

30. No genera condiciones de carrera (nunca se pierde nada), pero en realidad es una espera ocupada.

31. Si un filósofo bloquea, los vecinos pueden ver más tarde que tiene hambre al comprobar su estado, en*prueba*, para que pueda despertarse cuando los tenedores estén disponibles.

32. El cambio significaría que después de que un filósofo dejara de comer, ninguno de sus vecinos podría ser elegido a continuación. De hecho, nunca serían elegidos. Supongamos que el filósofo 2 terminara de comer. Correría*prueba*para los filósofos 1 y 3, y ninguno de los dos se pondría en marcha, a pesar de que ambos tenían hambre y ambos tenedores estaban disponibles. De manera similar, si el filósofo 4 terminaba de comer, el filósofo 3 no se pondría en marcha. Nada lo haría empezar.

33. Variación 1: los lectores tienen prioridad. Ningún escritor puede empezar cuando un lector está activo. Cuando aparece un nuevo lector, puede empezar inmediatamente a menos que un escritor esté activo en ese momento. Cuando un escritor termina, si hay lectores esperando, se empiezan todos, independientemente de la presencia de escritores en espera. Variación 2: los escritores tienen prioridad. Ningún lector puede empezar cuando un escritor está esperando. Cuando termina el último proceso activo, se empieza un escritor, si hay uno; de lo contrario, se empiezan todos los lectores (si hay alguno). Variación 3: versión simétrica. Cuando un lector está activo, nuevos lectores pueden empezar inmediatamente. Cuando un escritor termina, un nuevo escritor tiene prioridad, si hay uno esperando. En otras palabras, una vez que hemos empezado a leer, seguimos leyendo hasta que no queden lectores. De forma similar, una vez que hemos empezado a escribir, todos los escritores pendientes pueden ejecutarse.

34. Será necesario *Nuevo Testamento* segundo.

35. Si un proceso aparece varias veces en la lista, obtendrá varios cuantos por ciclo. Este enfoque se podría utilizar para dar a los procesos más importantes una mayor proporción de la CPU. Pero cuando el proceso se bloquea, es mejor eliminar todas las entradas de la lista de procesos ejecutables.

36. En casos simples, puede ser posible determinar si la E/S será limitante observando el código fuente. Por ejemplo, un programa que lee todos sus archivos de entrada en buffers al inicio probablemente no estará limitado por la E/S, pero un problema que lee y escribe de manera incremental en una cantidad de archivos diferentes (como un compilador) probablemente sí lo estará. Si el sistema operativo proporciona una función como UNIX *PD* Comando que puede indicarle la cantidad de tiempo de CPU utilizado por un programa. Puede compararlo con el tiempo total necesario para completar la ejecución del programa. Por supuesto, esto tiene más sentido en un sistema en el que usted es el único usuario.

37. Para múltiples procesos en una tubería, el padre común podría pasar al sistema operativo información sobre el flujo de datos. Con esta información

El sistema operativo podría, por ejemplo, determinar qué proceso podría proporcionar salida a un proceso que bloquea una llamada de entrada.

38. La eficiencia de la CPU es el tiempo útil de la CPU dividido por el tiempo total de la CPU.*Q* ≥*yo*, el ciclo básico es que el proceso se ejecute durante*yo*y someterse a un cambio de proceso para*S*. Por lo tanto (a) y (b) tienen una eficiencia de*T/*(*S*+*yo*). Cuando el quantum es más corto que*yo*, cada ejecución de*yo*requerirá*T/Q*Cambios de proceso, pérdida de tiempo*ST/Q*La eficiencia aquí es entonces

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

Lo que se reduce a*Q* /(*Q*+*S*), que es la respuesta a (c). Para (d), simplemente sustituimos *Q*para*S*y se encuentra que la eficiencia es del 50 por ciento. Finalmente, para (e), como *Q*→0 la eficiencia pasa a 0.

39. El trabajo más corto primero es la forma de minimizar el tiempo promedio de respuesta.

0<*incógnita*≤3:*incógnita*, 3, 5, 6, 9. 3
<*incógnita*≤5:3,*incógnita*, 5, 6, 9, 5<
incógnita≤6:3, 5,*incógnita*, 6, 9. 6<
incógnita≤9:3, 5, 6,*incógnita*, 9. *X* >
9: 3, 5, 6, 9,*INCÓGNITA.*

40. En el caso del round robin, durante los primeros 10 minutos cada trabajo obtiene 1/5 de la CPU. Al final de los 10 minutos,*do*en finDurante los siguientes 8 minutos, cada trabajo obtiene 1/4 de la CPU, tiempo después del cual*D*en fintermina. Luego, cada uno de los tres trabajos restantes obtiene 1/3 de la CPU durante 6 minutos, hasta que*B*en finLos tiempos de finalización de los cinco trabajos son 10, 18, 24, 28 y 30, con un promedio de 22 minutos.

Para la programación prioritaria,*B*se ejecuta primero. Después de 6 minutos, finaliza. Los demás trabajos finalizan a los 14, 24, 26 y 30, durante un promedio de 18,8 minutos. Si los trabajos se ejecutan en el orden*A*a través de*mi*, terminan a las 10, 16, 18, 22 y 30, con un promedio de 19,2 minutos. Finalmente, el trabajo más corto primero arroja tiempos de finalización de 2, 6, 12, 20 y 30, con un promedio de 14 minutos.

41. La primera vez obtiene 1 quantum. En ejecuciones posteriores obtiene 2, 4, 8 y 15, por lo que debe intercambiarse 5 veces.

42. Se podría realizar una comprobación para ver si el programa estaba esperando una entrada y si hizo algo con ella. Un programa que no estaba esperando una entrada y no la procesó no obtendría ningún aumento de prioridad especial.

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

44. La fracción de la CPU utilizada es 35/50 + 20/100 + 10/200 +*incógnita*/250. Para que sea programable, debe ser menor que 1. Por lo tanto*incógnita*debe ser inferior a 12,5 ms.

45. Se necesita una programación de dos niveles cuando la memoria es demasiado pequeña para contener todos los procesos listos. Se coloca un conjunto de ellos en la memoria y se realiza una elección.

De ese conjunto. De vez en cuando, se ajusta el conjunto de procesos internos. Este algoritmo es fácil de implementar y razonablemente eficiente, ciertamente mucho mejor que, por ejemplo, el algoritmo round robin sin tener en cuenta si un proceso estaba en la memoria o no.

46. El núcleo puede programar los procesos de cualquier forma que desee, pero dentro de cada proceso ejecuta los subprocesos estrictamente en orden de prioridad. Al permitir que el proceso del usuario establezca la prioridad de sus propios subprocesos, el usuario controla la política, pero el núcleo maneja el mecanismo.

47. Un posible script de shell podría ser

```
si [ ! -f números ]; entonces
echo 0 > números; fi conteo=0
mientras (prueba $count != 200 )
hacer
 count='expr
 $count + 1 '
 n='tail -1
 números'
 expr $n + 1
 >>números
 hechos
```

Ejecute el script dos veces simultáneamente, iniciándolo una vez en segundo plano (usando &) y otra vez en primer plano. Luego examine el archivo*números* Probablemente comience pareciendo una lista ordenada de números, pero en algún momento perderá su orden debido a la condición de carrera creada al ejecutar dos copias del script. La carrera se puede evitar haciendo que cada copia del script pruebe y bloquee el archivo antes de ingresar al área crítica y lo desbloquee al salir de ella. Esto se puede hacer de la siguiente manera:

```
Si ln números
números.lock
entonces
 n='cola -1 números'
```

```
 expr $n + 1
 >>números rm
 números.lock
en fin
```

Esta versión simplemente omitirá un turno cuando el archivo sea inaccesible, las soluciones variantes podrían poner el proceso en suspensión, realizar una espera activa o contar solo los bucles en los que la operación sea exitosa.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 3

1. En Estados Unidos, pensemos en una elección presidencial en la que tres o más candidatos compiten por la nominación de algún partido. Después de todo, las elecciones primarias

Una vez que se han terminado los debates, cuando los delegados llegan a la convención del partido, puede ocurrir que ningún candidato tenga mayoría y que ningún delegado esté dispuesto a cambiar su voto. Esto es un punto muerto. Cada candidato tiene algunos recursos (votos) pero necesita más para hacer el trabajo. En países con múltiples partidos políticos en el parlamento, puede ocurrir que cada partido apoye una versión diferente del presupuesto anual y que sea imposible reunir una mayoría para aprobar el presupuesto.
Esto también es un punto muerto.

2. Si la impresora comienza a imprimir un archivo antes de que se haya recibido todo el archivo (esto se permite a menudo para acelerar la respuesta), el disco puede llenarse con otras solicitudes que no se pueden imprimir hasta que se haya terminado el primer archivo, pero que utilizan el espacio de disco necesario para recibir el archivo que se está imprimiendo en ese momento. Si el administrador de trabajos de impresión no comienza a imprimir un archivo hasta que se haya enviado todo el archivo a la cola, puede rechazar una solicitud que sea demasiado grande. Comenzar a imprimir un archivo es equivalente a reservar la impresora; si la reserva se pospone hasta que se sabe que se puede recibir todo el archivo, se puede evitar un bloqueo de todo el sistema. El usuario con el archivo que no cabe sigue bloqueado, por supuesto, y debe ir a otra función que permita imprimir archivos más grandes.

3. La impresora no es preferente; el sistema no puede comenzar a imprimir otro trabajo hasta que el anterior esté completo. El disco de cola de impresión es preferente; puede eliminar un archivo incompleto que está creciendo demasiado y hacer que el usuario lo envíe más tarde, suponiendo que el protocolo lo permita.

4. Sí, no hay ninguna diferencia.

5. Sí, existen grafos ilegales. Hemos establecido que un recurso solo puede estar en posesión de un único proceso. Un arco que va desde un cuadrado de recursos hasta un círculo de procesos indica que el proceso es el propietario del recurso. Por lo tanto, un cuadrado con arcos que van desde él hasta dos o más procesos significa que todos esos procesos poseen el recurso, lo que viola las reglas. En consecuencia, cualquier grafo en el que varios arcos salgan de un cuadrado y terminen en diferentes círculos viola las reglas. Los arcos que van de cuadrados a cuadrados o de círculos a círculos también violan las reglas.

6. Una parte de todos esos recursos podría reservarse para que los usen únicamente los procesos que sean propiedad del administrador, de modo que él o ella siempre pueda ejecutar un shell y los programas necesarios para evaluar un bloqueo y tomar decisiones sobre qué procesos eliminar para que el sistema vuelva a ser utilizable.

7. Ninguno de los dos cambios conduce a un punto muerto. No hay espera circular en ninguno de los dos casos.

8. La renuncia voluntaria a un recurso es muy similar a la recuperación mediante la prelación. La diferencia esencial es que no se espera que los procesos informáticos resuelvan esos problemas por sí solos. La prelación es análoga a que el operador o el sistema operativo actúen como un policía, anulando las reglas normales que obedecen los procesos individuales.

9. El proceso solicita más recursos de los que tiene el sistema. No hay forma concebible de que pueda obtenerlos, por lo que nunca puede finalizar, incluso si ningún otro proceso desea recursos.

10. Si el sistema tuviera dos o más CPU, dos o más procesos podrían ejecutarse en paralelo, lo que generaría trayectorias diagonales.

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

12. El método sólo puede utilizarse para orientar la programación si se conoce de antemano el momento exacto en el que se va a reclamar un recurso. En la práctica, esto rara vez sucede.

13. Una solicitud de*D*no es seguro, pero uno de*do*es seguro

14. Existen estados que no son seguros ni están bloqueados, sino que conducen a estados bloqueados. A modo de ejemplo, supongamos que tenemos cuatro recursos: cintas, trazadores, escáneres y CD-ROM, como en el texto, y tres procesos que compiten por ellos. Podríamos tener la siguiente situación:

Tiene	Necesidades	Disponible
A: 2 0 0 0	1 0 2 0	0 1 2 1
B: 1 0 0 0	0 1 3 1	
DO:0 1 2 1	1 0 1 0	

Este estado no está bloqueado porque aún pueden ocurrir muchas acciones, por ejemplo,*A* Todavía se pueden obtener dos impresoras. Sin embargo, si cada proceso solicita sus requisitos restantes, tenemos un punto muerto.

15. El sistema no presenta interbloqueos. Supongamos que cada proceso tiene un recurso. Hay un recurso libre. Cualquiera de los procesos puede solicitarlo y obtenerlo, en cuyo caso puede finalizar y liberar ambos recursos. En consecuencia, el interbloqueo es imposible.

16. Si un proceso tiene*metro*recursos que puede terminar y no puede verse involucrado en un punto muerto. Por lo tanto, el peor caso es cuando todos los procesos tienen*metro*− 1 recurso y necesita otro. Si queda un

recurso, un proceso puede terminar y liberar todos sus recursos, dejando que el resto también termine. Por lo tanto, la condición para evitar el interbloqueo es $a \geq pag(metro-1) + 1$.

17. No. *D* Aún puede terminar. Cuando termina, devuelve suficientes recursos para permitir *mi* (o *A*) para terminar, y así sucesivamente.

18. Con tres procesos, cada uno puede tener dos unidades. Con cuatro procesos, la distribución de unidades será (2, 2, 1, 1), lo que permitirá que los dos primeros procesos finalicen. Con cinco procesos, la distribución será (2, 1, 1, 1, 1), lo que permitirá que el primero finalice. Con seis procesos, cada uno con una unidad de cinta y queriendo otra, tenemos un punto muerto.
Por lo tanto, para $n < 6$ el sistema no tiene puntos muertos.

19. Comparar una fila de la matriz con el vector de recursos disponibles lleva*metro* operaciones. Este paso debe repetirse en el orden de*norte*veces para encontrar un proceso que pueda terminar y marcarse como finalizado. Por lo tanto, marcar un proceso como finalizado asume el orden o*Minnesota*Pasos. Repitiendo el algoritmo para todos*norte*procesos significa que el número de pasos es entonces*Minnesota*2.

20. La matriz de necesidades es la siguiente:

0 1 0 0 2
0 2 1 0 0
1 0 3 0 0
0 0 1 1 1

Si*incógnita*es 0, tenemos un punto muerto inmediatamente. Si*incógnita*es 1, proceso*D*puede ejecutarse hasta completarse. Cuando haya terminado, el vector disponible será 1 1 2 2 1.
Lamentablemente, ahora estamos en un punto muerto. Si*incógnita*son 2, despues*D*corre, el vector disponible es 1 1 3 2 1 y*do*puede ejecutarse. Después de que finalice y devuelva sus recursos, el vector disponible es 2 2 3 3 1, lo que permitirá*B*correr y completar, y luego*A*para ejecutar y completar. Por lo tanto, el valor más pequeño de*incógnita*que evita un punto muerto es 2.

21. Sí. Supongamos que todos los buzones están vacíos. Ahora*A*Envía a*B*y espera una respuesta,*B*Envía a*do*y espera una respuesta, y*do*Envía a*A*y espera una respuesta. Ahora se cumplen todas las condiciones para el bloqueo.

22. Supongamos que ese proceso*A*solicita los registros en el orden*a*,*b*,*do*. Si proceso*B* También pide*a*en finPrimero, uno de ellos lo conseguirá y el otro lo bloqueará. Esta situación siempre está libre de puntos muertos, ya que el ganador ahora puede correr hasta el final sin interferencias. De las otras cuatro combinaciones, algunas pueden llevar a puntos muertos y otras están libres de puntos muertos. Los seis casos son los siguientes:

abc	punto
vol	muerto
ver	libre punto
bc	muerto
a	libre
tax	posible
i	punto
cb	muerto
a	posible
	punto
	muerto
	posible
	punto
	muerto
	posible
	punto
	muerto

Dado que cuatro de los seis pueden llevar a un punto muerto, hay una probabilidad de 1/3 de evitar un punto muerto y una probabilidad de 2/3 de conseguirlo.

23. El bloqueo en dos fases elimina los bloqueos, pero genera una posible inanición. Un proceso debe seguir intentando y fracasando para adquirir todos sus registros. No hay un límite superior para el tiempo que puede llevar.

24. Para evitar la espera circular, numere los recursos (las cuentas) con sus números de cuenta. Después de leer una línea de entrada, un proceso bloquea la línea con el número más bajo.

cuenta primero, luego, cuando obtiene el bloqueo (lo que puede implicar una espera), bloquea la otra. Dado que ningún proceso espera nunca una cuenta inferior a la que ya tiene, nunca hay una espera circular, por lo tanto, nunca hay un bloqueo.

25. Cambie la semántica de la solicitud de un nuevo recurso de la siguiente manera: si un proceso solicita un nuevo recurso y está disponible, obtiene el recurso y conserva lo que ya tiene. Si el nuevo recurso no está disponible, se liberan todos los recursos existentes. Con este escenario, el bloqueo es imposible y no hay peligro de que se adquiera el nuevo recurso pero se pierdan los existentes. Por supuesto, el proceso solo funciona si es posible liberar un recurso (puede liberar un escáner entre páginas o una grabadora de CD entre CD).

26. Le daría una calificación de F (suspendido). ¿Qué hace el proceso? Como claramente necesita el recurso, simplemente lo vuelve a solicitar y se bloquea nuevamente. 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 en competencia y asignar un recurso recién liberado al proceso que ha estado esperando más tiempo. Al agotar periódicamente el tiempo de espera y volver a intentarlo, un proceso pierde su antigüedad.

27. Si ambos programas piden primero a Woofer, los ordenadores se quedarán sin nada con la secuencia interminable: solicitar Woofer, cancelar solicitud, solicitar Woofer, cancelar solicitud, etc. 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 ordenadores han sido programados para ir primero a por el perro o la caseta del perro, se produce un punto muerto o una inanición. En realidad, no hay mucha diferencia entre los dos aquí.

En la mayoría de los problemas de interbloqueo, la inanición no parece grave porque la introducción de

demoras aleatorias suele hacer que sea muy improbable. Ese enfoque no funciona en este caso.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 4

1. La probabilidad de que los cuatro procesos estén inactivos es 1/16, por lo que el tiempo de inactividad de la CPU es 1/16.

2. Si cada trabajo tiene un 50 % de espera de E/S, tardará 20 minutos en completarse en ausencia de competencia. Si se ejecuta de forma secuencial, el segundo terminará 40 minutos después de que comience el primero. Con dos trabajos, la utilización aproximada de la CPU es de 1 – 0,52. De esta forma, cada uno obtiene 0,375 minutos de CPU por minuto de tiempo real. 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, si se ejecutan secuencialmente, los trabajos finalizan después de 40 minutos, pero si se ejecutan en paralelo, finalizan después de 26,67 minutos.

3. Es necesario copiar casi toda la memoria, lo que requiere leer cada palabra y reescribirla en una ubicación diferente. La lectura de 4 bytes lleva 10 nseg.

Por lo tanto, leer 1 byte lleva 2,5 ns y escribirlo lleva otros 2,5 ns, lo que da un total de 5 ns por byte compactado. Esta es una velocidad de 200 000 000 bytes/s. Para copiar 128 MB (227 bytes, que son aproximadamente 1,34×108bytes), la computadora necesita 227/200.000.000 segundos, lo que equivale aproximadamente a 671 ms. Este número es ligeramente pesimista porque si el agujero inicial en la parte inferior de la memoria es*a*bytes, esos*a*No es necesario copiar bytes. Sin embargo, si hay muchos huecos y muchos segmentos de datos, los huecos serán pequeños, por lo que*a*será pequeño y el error en el cálculo también será pequeño.

4. El mapa de bits necesita 1 bit por unidad de asignación. Con 227/*norte*Unidades de asignación, esto es 224/*norte*bytes. La lista enlazada tiene 227/216o 211nodos, cada uno de 8 bytes para un total de 2 14bytes. Para pequeños*norte*, la lista enlazada es mejor. Para grandes*norte*, el mapa de bits es mejor. El punto de cruce se puede calcular igualando estas dos fórmulas y resolviendo*norte*El resultado es 1 KB. Para*norte*Si el tamaño es menor a 1 KB, es mejor una lista enlazada.*norte*Si el tamaño es mayor a 1 KB, es mejor un mapa de bits. Por supuesto, la suposición de que los segmentos y los agujeros se alternan cada 64 KB es muy poco realista. Además, necesitamos*norte* <=64 KB si los segmentos y agujeros son de 64 KB.

5. El primer ajuste ocupa 20 KB, 10 KB y 18 KB. El mejor ajuste ocupa 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.

6. La memoria real utiliza direcciones físicas, que son los números a los que reaccionan los chips de memoria en el bus. Las direcciones virtuales son las direcciones lógicas que hacen referencia al espacio de direcciones de un proceso. Por tanto, una máquina con una palabra de 16 bits puede generar direcciones virtuales de hasta 64 KB, independientemente de si la máquina tiene más o menos memoria que 64 KB.

7. 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), (7, 2656).

8. (a) 8212 b) 4100 c) 24684

9. Construyeron una MMU y la insertaron entre el 8086 y el bus. De esta manera, todas las direcciones físicas del 8086 pasaron a la MMU como direcciones virtuales. La MMU luego las asignó a direcciones físicas, que pasaron al bus.

10. El espacio total de direcciones virtuales para todos los procesos combinados es$Nevada$Por lo tanto, se necesita esta cantidad de almacenamiento para las páginas. Sin embargo, una cantidad$a$puede estar en RAM, por lo que la cantidad de almacenamiento en disco requerida es solo$Nevada-a$Esta cantidad es mucho más de lo que se necesita en la práctica porque rara vez habrá$norte$procesos que realmente se están ejecutando y aún más raramente todos ellos necesitarán la memoria virtual máxima permitida.

11. Un error de página cada$a$Las instrucciones agregan una sobrecarga adicional den/kµsegundo al promedio, por lo que la instrucción promedio demora 10 +n/knseg.

12. La tabla de páginas contiene 232/213entradas, que son 524.288. La carga de la tabla de páginas lleva 52 ms. Si un proceso tarda 100 ms, esto consta de 52 ms para cargar la tabla de páginas y 48 ms para ejecutarse. Por lo tanto, el 52 por ciento del tiempo se dedica a cargar tablas de páginas.

13. Se utilizan veinte bits para los números de página virtuales, dejando 12 para el desplazamiento. Esto produce una página de 4 KB. Veinte bits para la página virtual implica 220
páginas.

14. El número de páginas depende del número total de bits en*a*,*b*,y*do* combinados. No importa cómo se dividan entre los campos.

15. Para una tabla de páginas de un nivel, hay 232/212o se necesitan 1 millón de páginas. Por lo tanto, la tabla de páginas debe tener 1 millón de entradas. Para la paginación de dos niveles, la tabla de páginas principal tiene 1000 entradas, cada una de las cuales apunta a una segunda tabla de páginas. Solo se utilizan dos de ellas. Por lo tanto, en total, solo se necesitan tres entradas de la tabla de páginas, una en la tabla de nivel superior y una en cada una de las tablas de nivel inferior.

16. El código y la cadena de referencia son los siguientes

CARGA 6144,R0	1(yo), 12(d)
EMPUJE R0	2(I), 15(D)
Llama al 5120	2(I), 15(D)
JEQ 5152	10(yo)

El código (I) indica una referencia de instrucción, mientras que (D) indica una referencia de datos.

17. El tiempo efectivo de instrucción es 1*yo*+5(1 −*yo*), dónde*yo*es la tasa de aciertos. Si igualamos esta fórmula con 2 y calculamos*yo*, encontramos que*yo*Debe ser al menos 0,75.

18. ElÆEl bit nunca es necesario en la TLB. La mera presencia de una página allí significa que se ha hecho referencia a la página; de lo contrario, no estaría

allí. Por lo tanto, el bit es completamente redundante. Sin embargo, cuando la entrada se vuelve a escribir en la memoria, el bitÆSe establece el bit en la tabla de páginas de memoria.

19. Una memoria asociativa compara básicamente una clave con el contenido de varios registros simultáneamente. Para cada registro debe haber un conjunto de comparadores que comparen cada bit del contenido del registro con la clave que se busca.
La cantidad de puertas (o transistores) necesarias para implementar dicho dispositivo es una función lineal de la cantidad de registros, por lo que ampliar el diseño resulta costoso linealmente.

20. Con páginas de 8 KB y un espacio de dirección virtual de 48 bits, la cantidad de páginas virtuales es 248⁄213, que es 235(unos 34 mil millones).

21. La memoria principal tiene 228⁄213= 32.768 páginas. Una tabla hash de 32 K tendrá una longitud de cadena media de 1. Para llegar a menos de 1, tenemos que pasar al siguiente tamaño,

65.536 entradas. Al distribuir 32.768 entradas en 65.536 espacios de tabla, se obtendrá una longitud de cadena media de 0,5, lo que garantiza una búsqueda rápida.

22. Probablemente esto no sea posible, excepto en el caso inusual y no muy útil de un programa cuyo curso de ejecución sea completamente predecible en el momento de la compilación. Si un compilador recopila información sobre las ubicaciones en el código de las llamadas a los procedimientos, esta información podría usarse en el momento del enlace para reorganizar el código objeto de modo que los procedimientos se ubiquen cerca del código que los llama. Esto haría más probable que un procedimiento esté en la misma página que el código que lo llama. Por supuesto, esto no ayudaría mucho en el caso de procedimientos llamados desde muchos lugares en el programa.

23. 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 6 errores de página; LRU produce 7.

24. Se elegirá la primera página con un bit 0, en este caso*D*.

25. Los contadores son Página 0: 0110110
Página 1: 01001001
Página 2: 00110111
Página 3: 10001011

26. La primera página con*R*=Se elegirá el valor 0 y la edad > τ. Como el escaneo comienza desde abajo, se descarta la primera página (1620).

27. La edad de la página es 2204 - 1213 = 991. Si τ = 400, definitivamente está fuera del conjunto de trabajo y no se hizo referencia a ella recientemente, por lo que se la expulsará. τ =1000 La situación es diferente. Ahora la página se encuentra dentro del conjunto de trabajo (por poco), por lo que no se elimina.

28. La latencia de búsqueda más rotación es de 20 mseg. El tiempo es de 1,25 mseg, para un total de 21,25 mseg. Para páginas de 2 KB, la transferencia Cargando 32 de estas páginas se tarda 680 ms. 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 ms. Cargar 16 de estas páginas lleva 360 ms.

29. NRU elimina la página 2. FIFO elimina la página 3. LRU elimina la página 1. Second Chance elimina la página 2.

30. El tambor de paginación PDP-1 tenía la ventaja de no tener latencia rotacional, lo que ahorraba media rotación cada vez que se escribía memoria en el tambor.

31. El texto ocupa ocho páginas, los datos cinco páginas y la pila cuatro páginas. El programa no cabe porque necesita 17 páginas de 4096 bytes. Con una página de 512 bytes, la situación es diferente. Aquí el texto ocupa 64 páginas, los datos 33 páginas y la pila 31 páginas, lo que hace un total de 128 páginas de 512 bytes, que caben. 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 al mismo tiempo.

33. Es posible. Suponiendo que no exista segmentación, la información de protección debe estar en la tabla de páginas. Si cada proceso tiene su propia tabla de páginas, cada uno también tiene sus propios bits de protección. Pueden ser diferentes.

34. El programa recibe 15.000 fallos de página, cada uno de los cuales utiliza 2 ms de tiempo de procesamiento adicional. En conjunto, la sobrecarga por fallos de página es de 30 segundos. Esto significa que de los 60 segundos utilizados, la mitad se gastó en la sobrecarga por fallos de página y la otra mitad en ejecutar el programa. Si ejecutamos el programa con el doble de memoria, obtenemos la mitad de fallos de página de memoria y solo 15 segundos de sobrecarga por fallos de página, por lo que el tiempo de ejecución total será de 45 segundos.

35. Funciona para el programa si no se puede modificar. Funciona para los datos si no se pueden modificar. Sin embargo, es común que no se pueda modificar el programa y extremadamente raro que no se puedan modificar los datos. Si el área de datos del archivo binario se sobrescribiera con páginas actualizadas, la próxima vez que se iniciara el programa, no tendría los datos originales.

36. La instrucción podría encontrarse a horcajadas sobre un límite de página, lo que provocaría dos fallos de

página solo para obtener la instrucción. La palabra obtenida también podría abarcar un límite de página, lo que generaría dos fallos más, para un total de cuatro. Si las palabras deben alinearse en la memoria, la palabra de datos puede provocar solo un fallo, pero una instrucción para cargar una palabra de 32 bits en la dirección 4094 en una máquina con una página de 4 KB es legal en algunas máquinas (incluido el Pentium).

37. La fragmentación interna se produce cuando la última unidad de asignación no está llena. La fragmentación externa se produce cuando se desperdicia espacio entre dos unidades de asignación.
En un sistema de paginación, el espacio desperdiciado en la última página se pierde debido a la fragmentación interna. En un sistema de segmentación pura, invariablemente se pierde algo de espacio entre los segmentos. Esto se debe a la fragmentación externa.

38. No. La clave de búsqueda utiliza tanto el número de segmento como el número de página virtual, por lo que se puede encontrar la página exacta en una sola coincidencia.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 5

1. En la figura, vemos un controlador con dos dispositivos. La razón por la que se espera que un solo controlador maneje varios dispositivos es para eliminar la necesidad de tener un controlador por dispositivo. Si los controladores se vuelven casi gratuitos, será más sencillo simplemente incorporar el controlador en el propio dispositivo. Este diseño también permitirá múltiples transferencias en paralelo y, por lo tanto, brindará un mejor rendimiento.
2. Fácil. El escáner genera un máximo de 400 KB/s. Tanto el bus como el disco funcionan a 16,7 MB/s, por lo que ni el disco ni el bus se acercan a la saturación.
3. No es una buena idea. El bus de memoria es seguramente más rápido que el bus de E/S, de lo contrario, ¿para qué molestarse con él? Considere lo que sucede con una solicitud de memoria normal. El bus de memoria termina primero, pero el bus de E/S sigue ocupado. Si la CPU espera hasta que el bus de E/S termine, ha reducido el rendimiento de la memoria al del bus de E/S. Si solo prueba el bus de memoria para la segunda referencia, fallará si esta es una referencia de dispositivo de E/S. Si hubiera alguna forma de abortar instantáneamente la referencia del bus de E/S anterior para probar la segunda, la mejora podría funcionar, pero nunca existe esa opción. En general, es una mala idea.
4. Cada transacción de bus tiene una solicitud y una respuesta, cada una de las cuales demora 100 ns, o 200 ns por transacción de bus. Esto da 5 millones de transacciones de bus por segundo. Si cada una es válida para 4 bytes, el bus tiene que manejar 20 MB/s. El hecho de que estas transacciones puedan distribuirse entre cuatro dispositivos de E/S en forma de round-robin es irrelevante. Una transacción de bus demora 200 ns, independientemente de si las solicitudes consecutivas se dirigen al mismo dispositivo o a dispositivos diferentes, por lo que no importa la cantidad de canales que tenga el

controlador DMA. El bus no lo sabe ni le importa.

5. Una interrupción requiere colocar 34 palabras en la pila. Para regresar de la interrupción es necesario recuperar 34 palabras de la pila. Esta sobrecarga por sí sola es de 680 nseg. Por lo tanto, la cantidad máxima de interrupciones por segundo no es más que aproximadamente 1,47 millones, suponiendo que no hay trabajo para cada interrupción.

6. Podría haberse hecho al principio. Una razón para hacerlo al final es que el código del procedimiento de servicio de interrupción es muy corto. Al generar primero otro carácter y luego reconocer la interrupción, si ocurre otra interrupción inmediatamente, la impresora funcionará durante la interrupción, lo que hará que imprima un poco más rápido. Una desventaja de este enfoque es un tiempo muerto ligeramente más largo cuando se pueden deshabilitar otras interrupciones.

7. Sí. La PC apilada apunta a la primera instrucción no recuperada. Se han ejecutado todas las instrucciones anteriores y la instrucción a la que apunta y sus sucesoras no se han ejecutado. Esta es la condición para una ejecución precisa.

Interrupciones. No es difícil lograr interrupciones precisas en una máquina con una sola secuencia de comandos. El problema surge cuando las instrucciones se ejecutan fuera de orden, lo que no sucede aquí.

8. La impresora imprime 50×80×6 = 24.000 caracteres/min, es decir, 400 caracteres/seg. Cada carácter utiliza 50µ980 ms de tiempo de CPU para la interrupción, por lo que colectivamente en cada segundo la sobrecarga de interrupción es de 20 ms. Al utilizar E/S controlada por interrupciones, los 980 ms restantes están disponibles para otro trabajo. En otras palabras, la sobrecarga de interrupción cuesta solo el 2% de la CPU, lo que prácticamente no afectará al programa en ejecución.

9. La independencia de dispositivos significa que se accede a los archivos y dispositivos de la misma manera, independientemente de su naturaleza física. Los sistemas que tienen un conjunto de llamadas para escribir en un archivo, pero un conjunto diferente de llamadas para escribir en la consola (terminal) no exhiben independencia de dispositivos.

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

11. Según los datos de la figura 5-17, para el ejemplo del disquete, hay 9×512×8 = 36864 bits por pista. A 200 mseg por rotación, la tasa de bits es de 184.320 bits/seg. El disco duro tiene un promedio de 281 sectores por pista, por lo que hay 281×512×8 = 1.150.976 bits/pista en promedio. Un tiempo de rotación de 8,33 ms corresponde a 120 rotaciones/seg (7200 rpm), por lo que en un segundo el disco puede transferir 120× 1.150.976 bits, es decir, unos 138 millones de bits por segundo. La velocidad de datos de un disquete es aproximadamente tres veces mayor que la de un módem de 56 Kbps. La velocidad de datos de un disco duro es aproximadamente un 38% más rápida que la de Fast Ethernet. Sin embargo, estos cálculos subestiman las velocidades de datos máximas reales, ya que por cada

512 bytes de datos de un disco hay también una cantidad de bytes de información de formato para identificar la pista y el sector, así como un espacio entre sectores, necesario para evitar que los sectores se superpongan si hay ligeras variaciones de velocidad.

12. Durante este proceso, que dura 4,1 ms, es necesario copiar cuatro veces un paquete. Además, hay dos interrupciones que suponen 2 ms. Por último, el tiempo de transmisión es de 0,83 ms, lo que supone un total de 6,93 ms por cada 1024 bytes. La velocidad máxima de datos es, por tanto, de 147.763 bytes/s, o aproximadamente el 12 por ciento de la capacidad nominal de la red de 10 megabits/s (si incluimos la sobrecarga del protocolo, las cifras empeoran aún más).

13. Si la impresora se asignara tan pronto como apareciera la salida, un proceso podría saturar la impresora imprimiendo algunos caracteres y luego entrar en suspensión durante una semana.

14. El disco gira a 120 rpm, por lo que una rotación demora 1000/120 ms. Con 200 sectores por rotación, el tiempo de sector es 1/200 de este número o 5/120 = 1/24 ms. Durante la búsqueda de 1 ms, 24 sectores pasan por debajo del cabezal. Por lo tanto, la desviación del cilindro debería ser 24.

15. Como vimos en el problema anterior, el tiempo de sector es de 1/24 mseg. Esto significa que el disco puede leer 24.000 sectores/seg. Como cada sector contiene 512 bytes, la velocidad de datos es de 12.288.000 bytes/seg. Esta velocidad es de 11,7 MB/seg.

16. El nivel RAID 2 no solo puede recuperarse de unidades dañadas, sino también de errores transitorios no detectados. Si una unidad presenta un solo bit defectuoso, el nivel RAID 2 lo corregirá, pero el nivel RAID 3 no.

17. La probabilidad de 0 fallos, $PAG0$, es $(1 - pag)a$. La probabilidad de 1 fallo, $PAG1$, es $kp(1 - pag)a-1$ La probabilidad de una falla RAID es entonces $1 - PAG0 - PAG1$ Esto es $1 - (1 - pag)a - kp(1 - pag)a-1$.

18. Se genera un campo magnético entre dos polos. No solo es difícil hacer que la fuente de un campo magnético sea pequeña, sino que además el campo se propaga rápidamente, lo que genera problemas mecánicos al intentar mantener la superficie de un medio magnético cerca de una fuente o sensor magnético. Un láser semiconductor genera 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.

19. Es posible. Si la mayoría de los archivos se almacenan en sectores lógicamente consecutivos, podría valer la pena intercalar los sectores para darle tiempo a los programas de procesar los datos recién recibidos, de modo que cuando se emita la siguiente solicitud, el disco esté en el lugar correcto.
Que esto valga la pena depende en gran medida del tipo de programas que se ejecuten y de lo uniforme que sea su comportamiento.

20. El tiempo de rotación es de 200 ms. Para leer todos los sectores en orden se necesita 1/2 rotación para obtener el sector 0 y 2,75 rotaciones para obtener los datos (después de leer el sector 7, la transferencia finaliza). Por lo tanto, se necesitan 3,25 rotaciones para 650 ms.
Leer 4K en 650 ms equivale a 6302 bytes/s. Para un disco no intercalado, se necesitan 300 ms para leer 4K, lo que equivale a 13 653 bytes/s. El intercalado reduce la capacidad a 6302/13653 o 0,46 de su capacidad anterior.

21. Tal vez sí y tal vez no. El intercalado doble es, en efecto, una desviación cilíndrica de dos sectores. Si el cabezal puede realizar una búsqueda de pista a pista en menos de dos sectores, no se necesita una desviación cilíndrica adicional. Si no puede, se necesita una desviación cilíndrica adicional para evitar perder un sector después de una búsqueda.

22. La capacidad de la unidad y las velocidades de transferencia se duplican. El tiempo de búsqueda y el retraso rotacional promedio son los mismos.

23. Una consecuencia bastante obvia es que ningún sistema operativo existente funcionará porque todos buscan allí dónde están las particiones del disco. Cambiar el formato de la tabla de particiones hará que todos los sistemas operativos fallen. La única forma de cambiar la tabla de particiones es cambiar simultáneamente todos los sistemas operativos para que utilicen 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. No necesariamente. Un programa UNIX que lee 10.000 bloques emite las solicitudes una a la vez, bloqueando cada una de ellas hasta que se completa. Por lo tanto, el controlador del disco ve solo una solicitud a la vez; no tiene la oportunidad de hacer nada más que procesarlas en el orden de llegada. Harry debería haber iniciado muchos procesos al mismo tiempo para ver si el algoritmo del elevador funcionaba.

26. Hay una carrera, pero no importa. Como la escritura estable ya se ha completado, el hecho de que la RAM no volátil no se haya actualizado simplemente significa que el programa de recuperación sabrá qué bloque se estaba escribiendo.
Leerá ambas copias. Si las encuentra idénticas, no cambiará ninguna, lo que es la acción correcta. El efecto del bloqueo justo antes de que se actualizara la RAM no volátil simplemente significa que el programa de recuperación tendrá que realizar dos lecturas de disco más de las que debería.

27. Dos mseg 60 veces por segundo son 120 mseg/seg, o el 12 por ciento de la CPU

28. El número de segundos en un año medio es 365,25×24×3600. Este número es
31.557.600. El contador vuelve a dar la vuelta después de 232segundos a partir del 1 de enero de 1970. El valor de 232/31.557.600 son 136,1 años, por lo que el cambio de fecha se producirá en 2106,1, es decir, a principios de febrero de 2106. Por supuesto, para entonces, todas las computadoras tendrán al menos 64 bits, por lo que no ocurrirá en absoluto.

29. Cada línea requiere 3200×8 = 25.600 muestras/seg. A 1µCada línea consume 25,6 ms del tiempo del procesador cada segundo. Con 39 líneas, el procesador está ocupado durante 39 segundos.×25,6 = 998,4 ms cada segundo, lo que da una capacidad de la tarjeta de 39 líneas.

30. Después de escribir un carácter en un terminal RS232, transcurre un tiempo (relativamente) largo antes de que se imprima. Esperar sería un desperdicio, por lo que se utilizan interrupciones. Con terminales mapeados en memoria, el carácter se acepta instantáneamente, por lo que las interrupciones no tienen sentido.

31. A 56 Kbps, tenemos 5600 interrupciones/seg, lo que equivale a 560 mseg. Esto representa el 56 % de la CPU.

32. Para desplazarse por la ventana se necesitan copiar 59 líneas de 80 caracteres o 4720 caracteres. Copiar 1 carácter (16 bytes) lleva 800 ns, por lo que toda la ventana lleva 3,776 ms. Escribir 80 caracteres en la pantalla lleva 400 ns, por lo que desplazarse y mostrar una nueva línea lleva 4,176 ms. Esto da unas 239,5 líneas/s.

33. Supongamos que el usuario, sin darse cuenta, le pidió al editor que imprimiera miles de líneas. Luego, presiona DEL para detenerlo. Si el controlador no descartara la salida, la salida podría continuar durante varios segundos después de DEL, lo que haría que el usuario presionara DEL una y otra vez y se frustrara cuando no sucediera nada.

34. Debe mover el cursor a la línea 5 posición 7 y luego eliminar 6 caracteres. La secuencia es ESC [5 ; 7 H ESC [6 P

35. El procesador integrado en el terminal tiene que mover todos los caracteres una línea hacia arriba simplemente copiándolos. Visto desde dentro, el terminal está mapeado en memoria. No hay una manera fácil de evitar esta organización a menos que se disponga de hardware especial.

36. Las 25 líneas de caracteres, cada una de 8 píxeles de alto, requieren 200 escaneos para dibujarse. Hay 60 pantallas por segundo, o 12.000 escaneos/seg. A 63,6µseg/scan, el haz se mueve horizontalmente a 763 ms por segundo, lo que deja 237 ms para escribir en la RAM de video. Por lo tanto, la RAM de video está disponible el 23,7 % del tiempo.

37. La velocidad máxima a la que se puede mover el ratón es de 200 mm/s, es decir, 2000 mickeys/s. Si cada informe tiene 3 bytes, la velocidad de salida es de 6000 bytes/s.

38. Con un sistema de color de 24 bits, solo 224Se pueden representar los colores. No todos. Por ejemplo, supongamos que un fotógrafo toma fotografías de 300 latas de pintura azul puro, cada una con una cantidad ligeramente diferente de pigmento. La primera lata podría estar representada por el valor (R, G, B) (0, 0, 1). La siguiente podría estar representada por (0, 0, 2), etc. Como la coordenada B tiene solo 8 bits, no hay

forma de representar 300 valores diferentes de azul puro. Algunas de las fotografías tendrán que ser renderizadas con el color incorrecto. Otro ejemplo es el color (120,24, 150,47, 135,89). No se puede representar, solo se puede aproximar por (120, 150, 136).

39. (a) Cada píxel ocupa 3 bytes en RGB, por lo que el espacio de tabla es 16×24×3 bytes, que son 1152 bytes. (b) A 100 nseg por byte, cada carácter ocupa 115,2µseg. Esto da una velocidad de salida de aproximadamente 8681 caracteres/seg.

40. Para reescribir la pantalla de texto es necesario copiar 2000 bytes, lo que se puede hacer en 20 µsegundos. Para reescribir la pantalla de gráficos es necesario copiar 1024×768×3 = 2.359.296 bytes, o aproximadamente 23,6 ms.

41. En Windows, el sistema operativo llama a los procedimientos del controlador por sí mismo. En X Windows, no sucede nada parecido. X simplemente recibe un mensaje y lo procesa internamente.

42. El primer parámetro es esencial. En primer lugar, las coordenadas son relativas a alguna ventana, por lo que*HDC-1*es necesario especificar la ventana y, por lo tanto, el origen. En segundo lugar, el rectángulo se recortará si queda fuera de la ventana, por lo que se necesitan las coordenadas de la ventana. En tercer lugar, el color y otras propiedades del rectángulo se toman del contexto especificado por*HDC-1*Es bastante esencial.

43. El tamaño de la pantalla es 400×160×3 bytes, es decir, 192 000 bytes. A 10 fps, esto equivale a 1 920 000 bytes/s o 15 360 000 bits/s. Esto consume el 15 % de Fast Ethernet.

44. El ancho de banda de un segmento de red se comparte, por lo que 100 usuarios que solicitan distintos datos simultáneamente en una red de 1 Mbps verán cada uno una velocidad efectiva de 10 Kbps. Con una red compartida, un programa de TV puede ser multicast, 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.

45. Si*norte*=10. La CPU puede seguir realizando su trabajo a tiempo, pero la energía utilizada disminuye considerablemente. Si la energía consumida en 1 segundo a máxima velocidad es*mi*, luego funciona a máxima velocidad durante 100 ms y luego permanece inactivo durante 900 ms.*MI/*10. Correr a una velocidad de 1/10 durante un segundo entero utiliza*MI/*100, un ahorro del 9*MI/*100. El porcentaje de ahorro al cortar el voltaje es del 90%.

46. El sistema de ventanas utiliza mucha más memoria para su visualización y utiliza más memoria virtual que el modo de texto. Esto hace que sea menos probable que el

disco duro esté inactivo durante un período lo suficientemente largo como para provocar que se apague automáticamente.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 6

1. Puedes subir y bajar del árbol tantas veces como quieras usando ''..''. Algunos de los muchos caminos son

```
/etc/contraseña
/./etc/contraseña
/./../etc/contraseña
/./../../etc/contraseña
/etc/../etc/contraseña
/etc/../etc/../etc/
contraseña
/etc/../etc/../
etc/../etc/contraseña
/etc/../etc/../etc/../
etc /../etc/contraseña
```

2. La forma de hacerlo en Windows es usar la extensión del archivo. Cada extensión corresponde a un tipo de archivo y a algún programa que maneja ese tipo. Otra forma es recordar qué programa creó el archivo y ejecutarlo. En Macintosh funciona de esta manera.

3. Estos sistemas cargaban el programa directamente en la memoria y comenzaban a ejecutarse en la palabra 0, que era el número mágico. Para evitar intentar ejecutar el encabezado como código, el número mágico era unRAMAInstrucción con una dirección de destino justo encima del encabezado. De esta manera, era posible leer el archivo binario directamente en el espacio de direcciones del nuevo proceso y ejecutarlo en 0, sin siquiera saber qué tan grande era el encabezado.

4. El sistema operativo se preocupa por la longitud de los registros cuando los archivos se pueden estructurar como registros con claves en una posición específica dentro de cada registro y es posible solicitar un registro con una clave determinada. En ese caso, el sistema debe saber qué tan grandes son los registros para poder buscar la clave en cada uno de ellos.

5. Para empezar, si no hubieraabierto,En cadaleerSería necesario especificar el nombre del archivo que se va a abrir. El sistema tendría que buscar el i-nodo correspondiente, aunque podría estar almacenado en caché. Un problema que surge rápidamente es cuándo volver a vaciar el i-nodo al disco. Sin embargo, podría agotarse el tiempo de espera. Sería un poco complicado, pero podría funcionar.

6. No. Si desea volver a leer el archivo, simplemente acceda aleatoriamente al byte 0.

7. Sí. ElrebautizarLa llamada no cambia la hora de creación ni la hora de la última modificación, pero al crear un nuevo archivo se obtiene la hora actual como hora de creación y hora de la última modificación. Además, si el disco está lleno, la copia puede fallar.

8. La parte asignada del archivo debe comenzar en un límite de página y tener una longitud de un número entero de páginas. Cada página asignada utiliza el

archivo en sí como memoria de respaldo. La memoria no asignada utiliza un archivo de borrador o una partición como memoria de respaldo.

9. Utilice nombres de archivo como */usr/ast/archivo* Aunque parezca un nombre de ruta jerárquico, en realidad es solo un nombre único que contiene barras diagonales incrustadas.

10. Una forma es agregar un parámetro adicional alleerllamada del sistema que indica desde qué dirección leer. En efecto, cadaleerEntonces tiene la posibilidad de hacer una búsqueda dentro del archivo. Las desventajas de este esquema son (1) un parámetro adicional en cadaleer llamar y (2) requerir que el usuario realice un seguimiento de dónde se encuentra el puntero del archivo.

11. El componente dotdot mueve la búsqueda a */usuario*, entonces *../astuto* lo pone en */usuario/ast*. De este modo *../ast/x* es lo mismo que */usuario/ast/x*.

12. Dado que el almacenamiento desperdiciado es *entre* las unidades de asignación (archivos), no dentro de ellas, esto es fragmentación externa. Es exactamente análoga a la fragmentación externa de la memoria principal que ocurre con un sistema de intercambio o un sistema que utiliza segmentación pura.

13. La transferencia tarda 9 ms en iniciarse. Para leer 213bytes a una velocidad de transferencia de 223 bytes/seg requiere 2-10seg (977 ms), para un total de 9,977 ms. Volver a escribirlo lleva otros 9,977 ms. Por lo tanto, copiar un archivo lleva 19,954 ms. Compactar la mitad de un disco de 16 GB implicaría copiar 8 GB de almacenamiento, lo que equivale a 220archivos. A 19,954 ms por archivo, esto lleva 20 923 segundos, es decir, 5,8 horas. Claramente, compactar el disco después de cada eliminación de archivo no es una gran idea.

14. Si se hace correctamente, sí. Al compactar, cada archivo debe organizarse de modo que todos sus bloques sean consecutivos, para un acceso rápido. Windows tiene un programa que desfragmenta y reorganiza el disco. Se recomienda a los usuarios que lo ejecuten periódicamente para mejorar el rendimiento del sistema. Pero dado el tiempo que lleva, ejecutarlo una vez al mes puede ser una buena frecuencia.

15. Una cámara digital fija graba una cierta cantidad de fotografías en secuencia en un medio de almacenamiento no volátil (por ejemplo, una memoria flash). Cuando se reinicia la cámara, el medio se vacía. A partir de entonces, las fotografías se graban una a la vez en secuencia hasta que el medio se llena, momento en el que se cargan en un disco duro. Para esta aplicación, lo ideal es un sistema de archivos contiguo dentro de la cámara (por ejemplo, en el medio de almacenamiento de fotografías).

16. Encuentra la dirección del primer bloque en la entrada del directorio. Luego sigue la cadena de punteros de bloque en la FAT hasta que encuentra el bloque que necesita. Luego recuerda este número de bloque para el siguiente bloque.leerllamada al sistema.

17. El bloque indirecto puede contener 256 direcciones de disco. Junto con las 10 direcciones de disco directas, el archivo máximo tiene 266 bloques. Como cada bloque tiene 1 KB, el archivo más grande tiene 266 KB.

18. Debe haber una manera de indicar que los punteros del bloque de direcciones contienen datos, en lugar de punteros. Si queda un bit sobrante en algún lugar entre los atributos, se puede utilizar. Esto deja los nueve punteros para datos. Si los punteros son a bytes cada uno, el archivo almacenado podría tener hasta $9a$ bytes de longitud. Si no queda ningún bit entre los atributos, la primera dirección del disco puede contener una dirección no válida para marcar los bytes siguientes como datos en lugar de punteros. En ese caso, el archivo máximo es $8a$ bytes.

19. Elinor tiene razón. Tener dos copias del i-nodo en la tabla al mismo tiempo es un desastre, a menos que ambas sean de solo lectura. El peor caso es cuando ambas se actualizan simultáneamente. Cuando los i-nodos se vuelven a escribir en el disco, el último en escribirse borrará los cambios realizados por el otro y se perderán los bloques del disco.

20. Los enlaces duros no requieren espacio adicional en el disco, solo un contador en el i-nodo para llevar un registro de cuántos hay. Los enlaces simbólicos necesitan espacio para almacenar el nombre del archivo al que apuntan. Los enlaces simbólicos pueden apuntar a archivos en otras máquinas, incluso a través de Internet. Los enlaces duros se limitan a apuntar a archivos dentro de su propia partición.

21. El mapa de bits requiere B bits. La lista gratuita requiere DF bits. La lista libre requiere menos bits si $DF < B$ Alternativamente, la lista gratuita es más corta si $F/B < 1/D$, dónde *PENSIÓN COMPLETA* es la fracción de bloques libres. Para direcciones de disco de 16 bits, la lista libre es más corta si el 6 por ciento o menos del disco está libre.

22. 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 do: 1111 1111 1111 1100
(d) Después de eliminar el archivo B: 1111 1110 0000 1100

23. No es un problema grave en absoluto. La reparación es sencilla, sólo lleva tiempo. El algoritmo de recuperación consiste en hacer una lista de todos los bloques de todos los archivos y tomar el complemento como la nueva lista libre. En UNIX esto se puede hacer escaneando todos los i-nodos. En el sistema de archivos FAT, el problema no puede ocurrir porque no hay una lista libre. Pero incluso si la hubiera, todo lo que habría que hacer para recuperarlo es escanear el FAT en busca de entradas libres.

24. Es posible que la tesis de Ollie no esté respaldada con la fiabilidad que desearía. Un programa de respaldo puede ignorar un archivo que esté abierto para escritura, ya que el estado de los datos en dicho archivo puede ser indeterminado.

25. Deben llevar un registro de la hora del último volcado en un archivo en el disco. En cada volcado, se añade una entrada a este archivo. En el momento del volcado, se lee el archivo y se anota la hora de la última

entrada. Cualquier archivo modificado desde ese momento se volca.

26. En (a) y (b), no se marcaría el 21. En (c), no habría cambio. En (d), 21 no se marcaría.

27. Muchos archivos UNIX son cortos. Si el archivo completo cabe en el mismo bloque que el inodo, solo se necesitaría un acceso al disco para leer el archivo, en lugar de dos, como sucede actualmente. Incluso para archivos más largos habría una ventaja, ya que se necesitaría un acceso al disco menos.

28. No debería ocurrir, pero debido a un error en algún lugar podría ocurrir. Significa que un bloque aparece en dos archivos y también dos veces en la lista libre. El primer paso para reparar el error es eliminar ambas copias de la lista libre. A continuación, se debe adquirir un bloque libre y copiar allí el contenido del bloque enfermo. Finalmente, se debe cambiar la aparición del bloque en uno de los archivos para que haga referencia a la copia recién adquirida del bloque. En este punto, el sistema vuelve a ser coherente.

29. El tiempo necesario es $yo+40\times(1-yo)$. La trama es simplemente una línea recta.

30. El tiempo por bloque se compone de tres componentes: tiempo de búsqueda, latencia rotacional y tiempo de transferencia. En todos los casos, la latencia rotacional más el tiempo de transferencia es el mismo, 125 ms. Solo difiere el tiempo de búsqueda. Para 13 cilindros es 78 ms; para 2 cilindros es 12 ms. Por lo tanto, para archivos colocados aleatoriamente el total es 203 ms, y para archivos agrupados es 137 ms.

31. A 15.000 rpm, el disco tarda 4 ms en dar una vuelta. El tiempo de acceso medio (en ms) para leer a bytes es entonces $8 + 2 + (a1 / 262144)\times 4$. Para bloques de 1 KB, 2 KB y 4 KB, los tiempos de acceso son 10,015625 ms, 10,03125 ms y 10,0625 ms, respectivamente (sin apenas diferencias). Estos tiempos dan como resultado velocidades de aproximadamente 102 240 KB/s, 204 162 KB/s y 407 056 KB/s, respectivamente.

32. Si todos los archivos tuvieran un tamaño de 1 KB, cada bloque de 2 KB contendría un archivo y 1 KB de espacio desperdiciado. Intentar poner dos archivos en un bloque no está permitido porque la unidad que se utiliza para llevar un registro de los datos es el bloque, no el semibloque. Esto da como resultado un desperdicio de espacio del 50 por ciento. En la práctica, cada sistema de archivos tiene archivos grandes y muchos pequeños, y estos archivos utilizan el disco de forma mucho más eficiente. Por ejemplo, un archivo de 32.769 bytes utilizaría 17 bloques de disco para el almacenamiento, dada una eficiencia de espacio de 32.768/34.816, que es aproximadamente el 94 por ciento.

33. El formato de directorio permite hasta 255 bloques en el disco (debido al número de 8 bits en la entrada del directorio). Esto claramente no es suficiente, por lo que el primer cambio debe ser pasar a más bloques de disco. En lugar de 16 direcciones de 8 bits, deberíamos usar 8 direcciones de 16 bits. Esto permite 65.536 bloques de disco por disco. Si hacemos que los

bloques sean de 32.768 bytes, el mismo tamaño que usa MS-DOS en discos grandes, el tamaño máximo del disco (en realidad, el tamaño máximo de la partición) es ahora de 2 GB. Sin embargo, los archivos individuales están limitados a ocho bloques de disco por extensión y 255 extensiones para un máximo de 2040 bloques o 63,75 MB. Un diseño con un disco máximo de 2 GB y un archivo máximo de 63,75 MB es plausible. Si esto no es suficiente, se podría usar uno de los bytes no utilizados en la entrada del directorio para aumentar el campo de extensión a 16 bits. Esto eleva la cantidad de bloques por archivo a 524.280 y el tamaño del archivo a casi 16 GB. Los tamaños de bloques más grandes alcanzan incluso más.

34. El bloque más grande es 32.768. Con 32.768 de estos bloques, el archivo más grande sería de 1 GB.

35. Limita la suma de las longitudes de todos los archivos a un tamaño que no supere el del disco.
No se trata de una restricción muy grave. Si los archivos fueran en conjunto más grandes que el disco, no habría lugar para almacenarlos todos en el disco.

36. El nodo i contiene 10 punteros. El bloque indirecto simple contiene 256 punteros. El bloque indirecto doble contiene 256 punteros.2punteros. El bloque triple indirecto es bueno para 2563punteros. Sumando todo esto, obtenemos un tamaño de archivo máximo de 16.843.018 bloques, lo que equivale aproximadamente a 16,06 GB.

37. Se necesitan las siguientes lecturas de disco:

directorio para /
i-nodo para*/usuario* directorio
para*/usuario* i- nodo
para*/usuario/ast* directorio
para*/usuario/ ast* i-nodo
para*/usr/ast/cursos* directorio
para*/ usr/ast/cursos* i-nodo
para*/usr/ast/cursos/os*
directorio
para*/usr/ast/cursos/os* i-nodo
para*/*
usr/ast/cursos/os/folleto.t

En total, se requieren 10 lecturas de disco.

38. Algunas ventajas son las siguientes. En primer lugar, no se desperdicia espacio en disco en i-nodos sin usar. En segundo lugar, no es posible quedarse sin i-nodos. En tercer lugar, se necesita menos movimiento de disco ya que el i-nodo y los datos iniciales se pueden leer en una sola operación. Algunas desventajas son las siguientes. En primer lugar, las entradas de directorio ahora necesitarán una dirección de disco de 32 bits en lugar de un número de i-nodo de 16 bits. En segundo lugar, se utilizará un disco entero incluso para archivos que no contienen datos (archivos vacíos, archivos de dispositivo). En tercer lugar, las comprobaciones de integridad del sistema de archivos serán más lentas debido a la necesidad de leer un bloque entero para cada i-nodo y porque los i-nodos estarán dispersos por todo el disco. En cuarto lugar, los archivos cuyo tamaño se ha diseñado cuidadosamente para adaptarse al tamaño del bloque ya no se ajustarán al tamaño del bloque debido al i-nodo, lo que arruina el rendimiento.

1. XGA es 1024×768. Con 24 bits/píxel y 25 fotogramas/seg obtenemos 471.859.200 bits/seg. Esta velocidad es demasiado alta para UltraWide SCSI, que solo puede alcanzar 320 Mbps.
2. La televisión NTSC estándar tiene una resolución de aproximadamente 640×480 píxeles. A 8 bits por píxel y 30 fotogramas por segundo, obtenemos un ancho de banda de 73 Mbps. Apenas lo alcanza con un canal. Dos canales serían demasiado.
3. De la tabla, HDTV es 1280×720 contra 640×480 para la televisión convencional. Tiene tres veces más píxeles y, por lo tanto, necesita tres veces más ancho de banda. La razón por la que no necesita cuatro veces más ancho de banda es que la relación de aspecto de la televisión de alta definición es diferente a la de la televisión convencional para adaptarse mejor a la de la película de 35 mm.
4. Para avanzar a cámara lenta, es suficiente que cada fotograma se muestre dos o más veces seguidas. No se necesita ningún archivo adicional. Retroceder lentamente es tan malo como retroceder rápidamente, por lo que se necesita un archivo adicional.

5. El audio se muestrea a 16 bits por muestra, 44.100 veces/seg con dos canales. Esto da una tasa de audio sin comprimir de 1.411.200 bits/seg o 176.400 bytes/seg. En 74 minutos, esto suma 747 MB. Esta es la capacidad total del CD. No está comprimido en absoluto. La razón por la que los datos se limitan a 650 MB es que se utiliza una mejor corrección de errores para los datos, ya que un error es más grave que para la música. Si se hubiera utilizado incluso un factor de compresión de dos en los CD de audio, los datos habrían sido menos de 374 MB y se podrían almacenar más de 74 minutos en un CD.

6. Hay 32.768 magnitudes posibles. Por ejemplo, supongamos que la señal varía de −32,768 voltios a +32.767 voltios y que el valor almacenado para cada muestra es la señal redondeada al número de milivoltios más cercano, como un entero de 16 bits con signo. 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, supongamos que la señal es de 0,0005 voltios. Esto se registra en 0 o 1. En el último caso, el error es del 50%. Por lo tanto, el ruido de cuantificación afecta a las amplitudes bajas más que a las altas. Los conciertos para flauta serán golpeados más fuerte que el rock and roll debido a sus amplitudes más bajas.

7. Un esquema de compresión/expansión de volumen podría implementarse de la siguiente manera. Un bit de la salida se reserva para indicar que la señal grabada se expande. Los 15 bits restantes se utilizan para la señal. Cuando los 5 bits de orden superior de la señal de 20 bits no son 00000, el bit de expansión es 0 y los otros 15 bits contienen los 15 bits de orden superior de los datos muestreados. Cuando los 5 bits de orden superior de la señal son 00000, el bit de expansión se activa y la señal de amplitud de 20 bits se desplaza 5 bits a la izquierda. En el extremo del oyente tiene lugar el proceso inverso. Este esquema aumenta ligeramente el ruido de cuantificación para señales fuertes (debido a una señal de 15 bits en lugar de una señal de 16 bits), pero lo disminuye para señales silenciosas, cuando el efecto de la cuantificación es más notable. Una desventaja importante es que esto no

es un estándar y no funcionaría con los reproductores de CD existentes, pero podría funcionar para música en línea reproducida con un complemento especial que usara este esquema en ambos extremos. Una versión más sofisticada podría utilizar 2 bits para denotar cuatro regímenes de expansión diferentes para diferentes niveles de señal.

8. El sistema PAL tiene más líneas de escaneo y más resolución espacial que el NTSC. Tiene 625 líneas verticales frente a las 525 del NTSC. También tiene más píxeles por línea, lo que da como resultado una imagen más nítida y utiliza el ancho de banda adicional. Por otro lado, el NTSC tiene más cuadros por segundo, por lo que es mejor para captar la acción rápida. Ninguno es "mejor" que el otro en este sentido. Se han hecho diferentes concesiones: mejor resolución en el tiempo frente a mejor resolución en el espacio. Todo esto es completamente independiente de los esquemas de codificación de color utilizados.
9. La diferencia no causa ningún problema. El algoritmo DCT se utiliza para codificar fotogramas I en un esquema similar al JPEG. Los macrobloques se utilizan en P-

Marcos para localizar macrobloques que aparecieron en marcos anteriores. Las dos cosas no tienen nada que ver entre sí y no entran en conflicto.

10. No, no lo hacen. El algoritmo de compensación de movimiento encontrará cada macrobloque en el cuadro anterior en algún desplazamiento desde su ubicación actual. Al codificar el hecho de que el macrobloque actual debe tomarse del cuadro anterior en una posición (Δ*incógnita*,Δ*y*) a partir del actual, no es necesario volver a transmitir el bloque en sí.

11. Los procesos que dan soporte a las tres transmisiones de video ya utilizan el 0,808 del tiempo de CPU, por lo que quedan 192 ms por segundo para el audio. Proceso de audio*A*Se ejecuta 33,333 veces por segundo, proceso de audio.*B*Se ejecuta 25 veces por segundo y procesa audio.*do*Se ejecuta 20 veces por segundo, lo que da un total de 78,333 ejecuciones por segundo. Estas 78,333 ejecuciones pueden utilizar 192 ms, por lo que cada ejecución puede utilizar 192/78,333 o 2,45 ms.

12. El primer proceso utiliza el 0,400 % de la CPU. El segundo utiliza el 0,375 % de la CPU. Juntos utilizan el 0,775 %. El límite de RMS para dos procesos es 2.× (20,5- 1), que es 0,828, por lo que se garantiza que RMS funciona.

13. Como 0,65 < ln 2, RMS siempre puede programar las películas, sin importar cuántas haya. Por lo tanto, RMS no limita la cantidad de películas.

14. La secuencia que comienza en*a*=150 es*A*6,*B*5,*do*4,*A*7,*B*6,*A*5, y*do*5. Cuando *do*5 termina en*a*=235 no hay trabajo que hacer hasta*a*=240 cuando*A*y*B*estar listo, por lo que el sistema queda inactivo durante 5 ms. La elección de ejecutar*B*5 antes *do*El número 4 es arbitrario. También se permite la otra opción.

15. Un lector de DVD es adecuado para ver películas en casa, pero el elevado tiempo de búsqueda de los sistemas de grabación óptica actuales limita su utilidad a proporcionar un único flujo de datos. Las unidades de DVD no admiten múltiples flujos con diferentes horas de inicio ni funciones de control similares a las de un VCR, como pausa, rebobinado y avance rápido para distintos usuarios. Con la tecnología actual, los datos tendrían que almacenarse

en una memoria extremadamente grande. Los discos duros son simplemente mejores.

16. Si el tiempo de espera en el peor de los casos es de 6 minutos, se debe iniciar una nueva transmisión cada 6 minutos. Para una película de 180 minutos, se necesitan 30 transmisiones.

17. La velocidad de datos es de 0,5 MB/seg. Un minuto de vídeo utiliza 30 MB. Para avanzar o retroceder 1 minuto cada vez se necesitan 60 MB.

18. La HDTV no supone ninguna diferencia. La película sigue teniendo 216.000 fotogramas. La pérdida por cada fotograma es de aproximadamente medio bloque de disco, o 0,5 KB. Para toda la película, esta pérdida es de 108 KB.

19. Hay cierta pérdida por cada fotograma. Cuantos más fotogramas tenga, mayor será la pérdida. NTSC tiene una mayor velocidad de fotogramas, por lo que tiene una pérdida ligeramente mayor. Pero, dadas las cifras involucradas, esta pérdida no es una fracción significativa del espacio total del disco.

20. El efecto principal de la HDTV son los fotogramas más grandes. Los fotogramas grandes tienden a hacer que la desventaja de los bloques pequeños sea menos grave porque se pueden leer de manera eficiente.

De ahí el argumento del rendimiento del disco a favor de los bloques
disminuye. grandes. Además, si los marcos no se dividen en bloques (ya que no lo son),

En este caso, tener fotogramas I que son una fracción sustancial de un bloque es un problema grave. A menudo puede ocurrir que un bloque esté parcialmente lleno y aparezca un fotograma I grande a continuación, desperdiciando una gran cantidad de espacio en el bloque actual. En general, pasar a HDTV favorece el modelo de bloques pequeños.

21. El buffer es lo suficientemente grande si el número de fotogramas I es 4 o menos. La probabilidad de obtener exactamente*a*Los I-frames son*do*(24,*a*)*IaB*24−*a*,dónde*I*es 0,1 y*B*es 0,9. Las probabilidades de obtener exactamente 0, 1, 2, 3 y 4 fotogramas I son 0,0798, 0,213, 0,272, 0,221 y 0,129, respectivamente. La suma de estos es 0,915. Esto significa que hay una probabilidad de falla de 0,085 o 8,5 %. Esto es demasiado grande para aceptarlo.

22. Para obtener el punto de reproducción en el medio del búfer, necesitamos poder leer y almacenar tres transmisiones a la vez. Cuando la película se reanuda a los 12 minutos, comenzamos a almacenar las transmisiones que están actualmente en 15 minutos y 20 minutos. Después de 3 minutos, hemos almacenado 15-18 minutos y 20-23 minutos. En ese punto, descartamos la transmisión privada y comenzamos a mostrar desde el búfer. Después de 2 minutos adicionales, tenemos 15-25 minutos almacenados y el punto de reproducción es 17 minutos. En este punto, solo cargamos el búfer de la transmisión ahora en 25 minutos. En 3 minutos, tenemos 15-28 minutos en el búfer y el punto de reproducción es 20 minutos. Hemos logrado nuestro objetivo. Debido a que no estamos sincronizados con las transmisiones de video a pedido cercanas, esto es lo mejor que

podemos hacer.

23. Una alternativa es tener un archivo independiente para cada idioma. Esta alternativa minimiza el uso de RAM pero desperdicia grandes cantidades de espacio en disco. Si el espacio en disco es barato y el objetivo es admitir la mayor cantidad posible de transmisiones a la vez, este enfoque es atractivo. Otra alternativa es almacenar la pista de audio para cada idioma por separado y realizar una búsqueda adicional por cuadro para obtener el audio. Este esquema hace un uso eficiente del espacio en disco, pero introduce búsquedas adicionales y, por lo tanto, reduce el rendimiento.

24. La constante de normalización, *do*, 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.

25. Un disco de 14 GB contiene 14×230o 15.032.385.536 bytes. Si se dividen uniformemente en 1000 cilindros, cada cilindro contiene 15.032.385, lo que es suficiente para un videoclip de 30 segundos. Por lo tanto, cada clip ocupa un cilindro. La pregunta es entonces qué fracción del peso total está representada por los 10 clips superiores de 1000. Sumando 1, 1/2, ... 1/10, obtenemos 2,92895. Multiplicando esto por 0,134 obtenemos 0,392, por lo que el brazo pasa casi el 40% de su tiempo dentro de los 10 cilindros del medio.

26. Para cuatro elementos, la ley de Zipf arroja 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 sistemas de distribución, todas las unidades se utilizarán por igual, suponiendo que todos los que pagan por una película la vean hasta el final. Sin embargo, el resultado en un momento determinado podría ser diferente.
Si todo el mundo en el pueblo quiere empezar a ver una película a las 8SOYLa disposición de la Fig. 7-0(b) golpearía inicialmente con más fuerza el primer disco, luego el siguiente disco 15 minutos después, etc. Las disposiciones de la Fig. 7-0(c) o (d) no se verían afectadas de esta manera.

27. PAL funciona a 25 fotogramas por segundo, por lo que los dos usuarios tienen una diferencia de 150 fotogramas. Fusionarlos en 3 minutos significa cerrar la brecha en 50 fotogramas por minuto. Uno va 25 fotogramas por minuto más rápido y el otro va 25 fotogramas por minuto más lento. La velocidad de fotogramas normal es de 1500 fotogramas por minuto, por lo que la velocidad de subida o bajada es de 25/1500 o 1/60, lo que equivale aproximadamente al 1,67 %.

28. Para NTSC, con 30 cuadros por segundo, una ronda es de 33,3 ms. El disco gira 180 veces por segundo, por lo que la latencia rotacional promedio es de media rotación o 2,8 ms. MPEG-2 funciona a unos 500.000 bytes por segundo o aproximadamente 16.667 bytes por cuadro. A 320 MB por segundo, el tiempo de transferencia de un cuadro es de aproximadamente 51µPor lo tanto, los tiempos de búsqueda, latencia rotacional y transferencia suman aproximadamente 5,8 ms. Por lo tanto, cinco flujos consumen 29 ms de los 33,3 ms, que es el máximo.

29. El tiempo de búsqueda promedio pasa de 3,0 ms a 2,4 ms, por lo que el tiempo por operación se reduce a 5,2 ms. Esto agrega un flujo más, lo que hace un

total de seis.

30. Seis flujos. La división en bandas es inútil. Cada operación de disco aún demora 5,2 ms para que el brazo pase por encima de los datos. Si el tiempo de transferencia es 51µsegundo o 13 µsec no hace mucha diferencia

31. Para el primer lote de cinco solicitudes, la crítica es la del cilindro 676, cuarto en la lista, pero con fecha límite de a=712 mseg. Por lo tanto, cada solicitud debe atenderse en 3 mseg o menos para que la cuarta se realice en a=712 mseg.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 8

1. Tanto USENET como SETI@home podrían describirse como sistemas distribuidos de área extensa. Sin embargo, USENET es en realidad más primitivo que el esquema de la Figura 8-1c, ya que no requiere ninguna infraestructura de red más allá de las conexiones punto a punto entre pares de máquinas. Además, dado que no realiza ningún trabajo de procesamiento más allá del necesario para garantizar la difusión adecuada de los artículos de noticias, podría debatirse si es realmente un sistema distribuido del tipo que nos ocupa en este capítulo. SETI@home es un ejemplo más típico de un sistema distribuido de área extensa; los datos se distribuyen a nodos remotos que luego devuelven los resultados de los cálculos al nodo coordinador.

2. Dependiendo de los detalles de cómo se conectan las CPU a la memoria, una de ellas pasa primero, por ejemplo, se apodera del bus primero. Completa su operación de memoria y luego sucede la otra. No es predecible cuál pasa primero, pero si el sistema ha sido diseñado para consistencia secuencial, no debería importar.

3. Una máquina de 200 MIPS emitirá 200 millones de referencias de memoria por segundo, 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 la cantidad de solicitudes de memoria por segundo a 20 millones, lo que permite que 20 CPU compartan el bus. Para obtener 32 CPU en el bus, cada una podría solicitar no más de
12.500.000 solicitudes por segundo. Si solo 12,5 millones de los 200 millones de referencias de memoria salen por el bus, la tasa de errores de caché debe ser de 12,5/200 o 6,25 %. Esto significa que la tasa de aciertos es del 93,75 %.

4. Las CPU 000, 010, 100 y 110 se desconectan de las memorias 010 y 011.

5. Cada CPU gestiona completamente sus propias señales. Si se genera una señal desde el teclado y este no está asignado a ninguna CPU en particular (el caso habitual), de alguna manera la señal debe enviarse a la CPU correcta para que la gestione.

6. Para emitir una llamada al sistema, un proceso genera una trampa. La trampa interrumpe su propia CPU. De algún modo, la información de que una CPU esclava ha tenido una trampa debe transmitirse a la CPU maestra. Esto no sucede en el primer modelo. Si hay instrucciones de trampa entre procesadores, esto se puede utilizar para enviar una señal a la CPU maestra. Si no existen tales instrucciones, el esclavo puede recopilar los parámetros de la llamada al sistema y colocarlos en una estructura de datos en la memoria que el maestro sondea continuamente cuando está inactivo.

7. He aquí una posible solución:

```
ingresar3región:
      Bloqueo TST              | Probar el valor del bloqueo
      JNE                      | Si no es cero, vuelve a
ENTRAR3REGISTRO                intentarlo
      TSL REGIÓN,              | Copiar bloqueo al registro y
BLOQUEO                        establecer el bloqueo en 1
      REGISTRO CMP,#0          | ¿El bloqueo era cero?
      JNE                      | Si no era cero, se
ENTRAR3REGIÓN                  estableció el bloqueo, por
      RET                      lo que se repite el bucle |
                               Regresa al llamador; se
                               ingresó a la región crítica
```

8. Probablemente los bloqueos en las estructuras de datos sean suficientes. Es difícil imaginar algo que un fragmento de código pueda hacer que sea crítico y que no involucre alguna estructura de datos del núcleo. Por ejemplo, toda adquisición y liberación de recursos utiliza estructuras de datos. Si bien no se puede demostrar, es muy probable que los bloqueos en las estructuras de datos sean suficientes.

9. Se necesitan 16 ciclos de autobús para mover el bloque y va en ambos sentidos para cada uno.TSL. Por lo tanto, cada 50 ciclos de bus, 32 de ellos se desperdician en mover el bloque de caché. En consecuencia, el 64% del ancho de banda del bus se desperdicia en transferencias de caché.

10. Sí, lo haría, pero el tiempo entre sondeos podría terminar siendo muy largo, lo que degradaría el rendimiento. Pero sería correcto, incluso sin un máximo.

11. Es tan bueno comoTSLSe utiliza precargando un 1 en el registro que se va a utilizar. Luego, ese registro y la palabra de memoria se intercambian atómicamente. Después de la instrucción, la palabra de memoria se bloquea (es decir, tiene un valor de 1). Su valor anterior ahora está contenido en el registro. Si anteriormente estaba bloqueada, la palabra no se ha cambiado y el llamador debe realizar un bucle. Si anteriormente estaba desbloqueada, ahora está bloqueada.

12. El bucle consta de unTSLinstrucción (5 nseg), un ciclo de bus (10 nseg) y una JMPDe vuelta a laTSLinstrucción (5 nseg). Por lo tanto, en 20 nseg, se solicita 1 ciclo de bus que ocupa 10 nseg. El bucle consume el 50% del bus.

13. *A*Es el proceso que se acaba de seleccionar. Es posible que haya otros en la misma CPU.

14. La programación por afinidad tiene que ver con colocar el subproceso correcto en la CPU correcta.
Hacerlo podría reducir los errores de TLB, ya que estos se mantienen dentro de cada CPU. Por otro lado, no tiene efecto sobre los errores de página, ya que si una página está en la memoria de una CPU, está en la memoria de 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 es que los nodos en las esquinas opuestas intenten comunicarse. Sin embargo, con un toro, las esquinas opuestas están a solo dos saltos de distancia. El peor de los casos es que una esquina intente comunicarse con un nodo en el medio.a, se necesita (a−1)/2 saltos para ir de una esquina al medio horizontalmente y otro (a −1)/2 saltos para ir al centro verticalmente, para un total de$a$−1. Para incluso$a$, el medio es un cuadrado de cuatro puntos en el medio, por lo que el peor caso es desde una esquina hasta el punto más distante en ese cuadrado de cuatro puntos.$a1$ /2 saltos para llegar horizontalmente y también$a1$ /2 verticalmente, por lo que el diámetro es$a$.

17. La red se puede dividir en dos mediante un plano que

pase por el medio, lo que da dos sistemas, cada uno con una geometría de 8×8×4. Hay 64 enlaces funcionando entre las dos mitades, para un ancho de banda de bisección de 64 Gbps.

18. Si solo consideramos el tiempo de red, obtenemos 1 nseg por bit o 512 nseg de retraso por paquete. Para copiar 64 bytes de a 4 bytes a la vez, se necesitan 320 nseg en cada lado, o 640 nseg en total. Si sumamos el tiempo de transmisión de 512 nseg, obtenemos un total de 1132 nseg. Si se necesitan dos copias adicionales, obtenemos 1792 nseg.

19. Si consideramos solo el tiempo de transmisión, una red de 1 Gbps entrega 125 MB/seg. Mover 64 bytes en 1152 nseg es 55,6 MB/seg. Mover 64 bytes en 1792 nseg es 35,7 MB/ seg.

20. En una máquina de memoria compartida, basta con pasar el puntero al mensaje desde la CPU que ejecuta elenviara la CPU que ejecuta elrecibir,con posibles traducciones entre direcciones de memoria virtual y física. En un

Multiordenador sin memoria compartida: una dirección en el espacio de direcciones de una CPU no tiene ningún significado para otra CPU, por lo que el contenido real del búfer de envío debe transmitirse como paquetes y luego reensamblarse en el búfer del proceso receptor. Para el programador, los procesos parecen idénticos, pero el tiempo requerido será mucho mayor en el multiordenador.

21. Es hora de moverse*a*Los bytes por E/S programada son 20*a*nsec. El tiempo para DMA es 2000 + 5*a*nsec. Igualando estos y resolviendo para*a*Obtenemos el punto de equilibrio en 133 bytes.

22. Es evidente que ocurre algo incorrecto si se ejecuta una llamada al sistema de forma remota.
Intentar leer 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 de vuelta a la máquina que realiza la llamada. Una forma de gestionar las llamadas al sistema remoto es atraparlas y enviarlas de vuelta al sitio de origen para su ejecución.

23. En primer lugar, en una red de difusión, se podría realizar una solicitud de difusión. En segundo lugar, se podría mantener una base de datos centralizada de quién tiene qué página. En tercer lugar, cada página podría tener una base de origen, indicada por la parte superior*a*bits de su dirección virtual; 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*,*mi*, y*GRAMO*, el nodo 2 tiene*B*y*F*, y el nodo 3 tiene*do*,*D*, *yo*, y*I*El corte entre los nodos 1 y 2 ahora contiene*De*y*EB*para un peso de 5. El corte entre los nodos 2 y 3 ahora contiene*CD*,*CI*,*ES*, y *FH*para un peso de 14. El corte entre los nodos 1 y 3 ahora contiene*Eh*y *G.H.*para un peso de 8. La suma es 27.

25. La tabla de archivos abiertos se guarda en el núcleo, por lo que si un proceso tiene archivos abiertos, cuando se descongela e intenta utilizar uno de sus archivos, el nuevo núcleo no lo sabe. Un segundo problema es la máscara de señal, que también se almacena en el núcleo original. Un tercer problema es

que si hay una alarma pendiente, se activará en la máquina equivocada. En general, el núcleo está lleno de fragmentos de información sobre el proceso, y también deben migrarse correctamente.

26. Los nodos Ethernet deben ser capaces de detectar colisiones entre paquetes, por lo que el retardo de propagación entre los dos nodos más separados debe ser menor que la duración del paquete más corto que se va a enviar. De lo contrario, el remitente puede transmitir un paquete en su totalidad y no detectar una colisión, aunque el paquete sufra una colisión cerca del otro extremo del cable.

27. El middleware se ejecuta en diferentes sistemas operativos, por lo que el código es claramente diferente porque las llamadas del sistema integrado son diferentes. Lo que tienen en común es producir una interfaz común con la capa de aplicación que está por encima de ellas. Si la capa de aplicación solo realiza llamadas a la capa de middleware y

Si no se hacen llamadas al sistema, todas las versiones pueden tener el mismo código fuente. Si también se hacen llamadas al sistema, estas serán diferentes.

28. Los servicios más adecuados son

(a) Conexión no confiable.
(b) Flujo de bytes confiable.

29. Se mantiene de forma jerárquica. Hay un servidor mundial para *.educación* que sabe de todas las universidades y una *.com* servidor que conoce todos los nombres que terminan en *.con*. Así que mirar hacia arriba *cs.uni.edu*, una máquina primero buscaría *uni* al *. educación* servidor, luego vaya allí para preguntar sobre *cs*, etcétera.

30. 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. Se necesita alguna manera de hacer posible dirigir una conexión entrante a un proceso en particular. Esto se hace haciendo que cada proceso escuche un puerto específico. Se ha acordado que los servidores web escucharán el puerto 80, por lo que las conexiones entrantes dirigidas al servidor web se envían al puerto 80. El número en sí fue una elección arbitraria, pero se tuvo que elegir algún número.

31. Pueden hacerlo. Por ejemplo, *www.intel.com* No dice nada sobre dónde está el servidor.

32. Una forma de hacerlo sería que el servidor web empaquetara la página completa, incluidas todas las imágenes, en un archivo zip grande y enviara todo el documento la primera vez, de modo que solo se necesitara una conexión. Una segunda forma de hacerlo sería utilizar un protocolo sin conexión, como UDP. Esto eliminaría la sobrecarga de conexión, pero requeriría que los servidores y los navegadores realicen su propio control de errores.

33. Tener el valor de unaleerDepender de si un proceso está en la misma máquina que el último escritor no es en absoluto transparente. Esto justifica que los cambios solo sean visibles para el proceso que los realiza. Por otro lado, tener un único administrador de caché por máquina es más fácil y más barato de

implementar. Un administrador de este tipo se vuelve mucho más complicado si tiene que mantener múltiples copias de cada archivo modificado, y el valor devuelto depende de quién esté realizando la lectura.

34. Algunos archivos almacenados en caché deben enviarse de vuelta al servidor. Se pueden utilizar todos los algoritmos de paginación estándar, como LRU o segunda oportunidad. Sin embargo, a diferencia de la memoria virtual, es posible utilizar LRU exacto porque las referencias a archivos son poco frecuentes (escala de tiempo de milisegundos, no de nanosegundos).

35. La memoria compartida funciona con páginas completas. Esto puede dar lugar a una compartición falsa, en la que el acceso a variables no relacionadas que se encuentran en la misma página provoca una pérdida de memoria. Colocar cada variable en una página separada es un desperdicio. El acceso basado en objetos elimina estos problemas y permite una compartición más fina.

36. El hash en cualquiera de los campos de la tupla cuando se inserta en el espacio de la tupla no ayuda porque*en*Puede tener principalmente parámetros formales. Una optimización que siempre funciona es observar que todos los campos de ambos*afuera*y*en* están tipificados. Por lo tanto, se conoce la firma de tipo de todas las tuplas en el espacio de tuplas y se necesita el tipo de tupla en una*en*También se conoce como "subespacio de tuplas". Esto sugiere crear un subespacio de tuplas para cada firma de tipo. Por ejemplo, todas las tuplas (int, int, int) van en un espacio y todas las tuplas (string, int, float) van en un espacio diferente. Cuando una*en*se ejecuta, solo se debe buscar el subespacio coincidente.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 9

1. La restricción es que no puede haber dos celdas que contengan las mismas dos letras, de lo contrario el descifrado sería ambiguo. Por lo tanto, cada uno de los 676 elementos de la matriz contiene uno diferente de los 676 digramas. El número de combinaciones diferentes es, por lo tanto, 676. Se trata de un número muy grande.

2. ha llegado el momento dijo la morsa de hablar 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 no hay prisa dijo el carpintero le agradecieron mucho por eso

De*A través del espejo*((Tweedledum y Tweedledee).

3. El número de permutaciones es*norte*!, por lo que este es el tamaño del espacio de claves. Una ventaja es que el ataque estadístico basado en propiedades de

lenguajes naturales no funciona porque una E realmente representa una E, etc.

4. El remitente elige una clave aleatoria y la envía al tercero de confianza cifrada con la clave secreta que comparte con él. A continuación, el tercero de confianza descifra la clave aleatoria y la vuelve a cifrar con la clave secreta que comparte con el receptor. A continuación, este mensaje se envía al receptor.

5. Una función como $y = incógnita^a$ es fácil de calcular pero tomando el a-ésima raíz de y Es mucho más difícil.

6. Depende de la longitud de la contraseña. El alfabeto que compone las contraseñas tiene 62 símboElol se.spacio total de búsqueda es $625 + 626 + 627 + 628$, que es aproximadamente 2×1014. Si se sabe que la contraseña es a personajes, los El espacio de búsqueda se reduce a solo $62a$.La relación entre estos es entonces $2 \times 1014 / 62a$. Para a de 5 a 8, estos valores son 242,235, 3907, 63 y 1. En otras palabras, saber que la contraseña tiene solo 5 caracteres reduce el espacio de búsqueda en un

Factor de 242.235, ya que no es necesario probar todas las contraseñas largas. Esto es una gran ventaja. Sin embargo, saber que tiene ocho caracteres no ayuda mucho, ya que significa que se pueden omitir todas las contraseñas cortas (fáciles).

7. Intente calmar al asistente. El algoritmo de cifrado de contraseñas es público. Las contraseñas están cifradas por el*acceso*programa tan pronto como se escriben, y la contraseña cifrada se compara con la entrada en el archivo de contraseñas.

8. No, no es así. El estudiante puede averiguar fácilmente cuál es el número aleatorio de su superusuario. Esta información se encuentra en el archivo de contraseñas sin cifrar. Si es 0003, por ejemplo, entonces simplemente intenta cifrar las posibles contraseñas como *Susan0003*,*Boston0003*,*IBMPC0003*, etc. Si otro usuario tiene contraseña *Boston0004*Pero no lo descubrirá.

9. Existen muchos criterios. A continuación se indican algunos de ellos:

Debe ser fácil e indoloro de medir (no muestras de sangre) Debe haber muchos valores disponibles (no el color de los ojos)
La característica no debe cambiar con el tiempo (no el color del cabello). Debe ser difícil falsificar la característica (no el peso).

10. No, no es factible. El problema es que no se comprueban los límites de las matrices. Las matrices no se alinean con los límites de las páginas, por lo que la MMU no resulta de ninguna ayuda. Además, realizar una llamada al núcleo para cambiar la MMU en cada llamada a un procedimiento sería prohibitivamente costoso.

11. Si se puede confiar en todas las máquinas, funciona bien. Si no se puede confiar en algunas, el sistema se desmorona, porque una máquina no confiable podría enviar un mensaje a una 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 en el superusuario o en un estudiante.

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

13. No funcionaría utilizarlas hacia adelante. Si un intruso capturara una, sabría cuál utilizar la próxima vez. Usarlas hacia atrás evita este peligro.

14. Una forma de firmar un documento sería que la tarjeta inteligente leyera el documento, creara un hash del mismo y luego encriptara el hash con la clave privada del usuario, almacenada en la tarjeta. El hash encriptado se enviaría a la computadora del cibercafé, pero la clave secreta nunca saldría de la tarjeta inteligente, por lo que el sistema es seguro.

15. Si se utilizan las capacidades para hacer posible tener pequeños dominios de protección, no; en caso contrario, sí. Si se inicia un editor, por ejemplo, con solo

Si el editor tiene acceso a todos los objetos del usuario, los troyanos pueden hacer su trabajo sucio, independientemente de si tienen o no capacidad para editar el archivo o no.

16. El compilador podría insertar código en todas las referencias de matriz para realizar una comprobación de límites. Esta característica evitaría ataques de desbordamiento de búfer. No se hace porque ralentizaría significativamente todos los programas. Además, en C no es ilegal declarar una matriz de tamaño 1 como parámetro de procedimiento y luego hacer referencia al elemento 20, pero claramente la matriz real cuya dirección se ha pasado debería tener al menos 20 elementos.

17. Desde el punto de vista de la seguridad, sería ideal. A veces, los bloques usados quedan expuestos y se filtra información valiosa. Desde el punto de vista del rendimiento, poner a cero los bloques desperdicia tiempo de CPU, lo que degrada el rendimiento.

18. Debe leer la contraseña completa todo el tiempo, incluso si detecta con anticipación que la contraseña es incorrecta. De esa manera, siempre se producirá un error de página cuando la contraseña se encuentre parcialmente en una página que no tenga memoria.

19. Para cualquier sistema operativo, todos los programas deben comenzar su ejecución en una dirección conocida o tener una dirección de inicio almacenada en una posición conocida en el encabezado del archivo del programa. (a) El virus primero copia las instrucciones en la dirección de inicio normal o la dirección en el encabezado a un lugar seguro, y luego inserta un salto a sí mismo en el código o su propia dirección de inicio en el encabezado. (b) Cuando termina con su propio trabajo, el virus ejecuta las instrucciones que tomó prestadas seguidas de un salto a la siguiente instrucción que se habría ejecutado, o transfiere el control a la dirección que encontró en el encabezado original.

20. Un registro de arranque maestro requiere sólo un sector y, si el resto de la primera pista está libre,

proporciona espacio donde un virus puede ocultar el sector de arranque original, así como una parte sustancial de su propio código. Los controladores de disco modernos leen y almacenan en búfer pistas enteras a la vez, por lo que no habrá demoras perceptibles ni sonidos de búsquedas adicionales a medida que se leen los datos adicionales.

21. Los programas en C tienen extensión *.do*. En lugar de utilizar elaccesoLlamada del sistema para probar el permiso de ejecución, examine el nombre del archivo para ver si termina en *.do*Este código lo hará.

```
carbonizar
se*archivo
3 nombre;
int len;
archivo3nombre = dp->d3nombre; len
= strlen(archivo3nombre);
si (strcmp(&archivo3nombre[len - 2], ".c") == 0)
infectar(es);
```

22. Probablemente no lo sepan, pero pueden suponer que al combinar una palabra del virus con el resto mediante la técnica XOR se obtendrá un código de máquina válido. Sus ordenadores pueden probar cada palabra del virus por turno y ver si alguna de ellas produce un código de máquina válido. Para ralentizar este proceso, Virgil puede utilizar un algoritmo de cifrado mejor, como utilizar claves diferentes para las palabras pares e impares, y luego rotar la primera palabra que quede por una cantidad de bits determinada por una función hash en las claves, rotar la segunda palabra por esa cantidad de bits más uno, etc.

23. El compresor es necesario para comprimir otros programas ejecutables como parte del proceso de infección.

24. La mayoría de los virus no quieren infectar un archivo dos veces. Es posible que ni siquiera funcione. Por lo tanto, es importante poder detectar el virus en un archivo para ver si ya está infectado. Todas las técnicas que se utilizan para dificultar que el software antivirus detecte los virus también dificultan que el propio virus pueda identificar qué archivos han sido infectados.

25. Primero, ejecutar el*disco duro*El programa del disco duro es un error. Puede estar infectado y puede infectar el sector de arranque. Debe ejecutarse desde el CD-ROM original o desde un disquete protegido contra escritura. En segundo lugar, los archivos restaurados pueden estar infectados. Si los vuelve a colocar sin limpiarlos, es posible que se vuelva a instalar el virus.

26. Sí, pero el mecanismo es ligeramente diferente al de Windows. En UNIX, un virus acompañante puede instalarse en un directorio de la ruta de búsqueda anterior a aquel en el que se encuentra el programa real. El ejemplo más común es insertar un programa*es*en un directorio de usuarios, lo que efectivamente anula*/contenedor/ls*Porque se encuentra primero.

27. Un gusano es un programa independiente que funciona por sí solo. Un virus es un fragmento de código que se adjunta a otro programa. El gusano se reproduce

haciendo más copias del programa gusano. El virus se reproduce infectando otros programas.

28. Obviamente, ejecutar cualquier programa de una fuente desconocida es peligroso. Los archivos autoextraíbles pueden ser especialmente peligrosos, ya que pueden liberar múltiples archivos en múltiples directorios y el propio programa de extracción podría ser un troyano. Si tiene la opción, es mucho mejor obtener los archivos en forma de un archivo normal, que luego puede extraer con herramientas en las que confíe.

29. No importa. Si se utiliza el relleno de ceros, entoncesS2debe contener el prefijo verdadero como un entero sin signo en el orden inferior$a$bits. Si se utiliza la extensión de signo, entoncesS2
También debe ser un signo extendido. MientrasS2contiene los resultados correctos de cambiar una
dirección verdadera, no importa lo que haya en los bits superiores no utilizados de
S2.

30. Aquí está la matriz de protección:

	GMV	asw	usuarios	desarrollar	Notas del PPP	programa	proyecto
GMV	Leer	Leer	Leer	Leer	Escribir		
asw							
usuarios		Leer	Leer	Leer			
desarrollar				Leer			
Notas del PPP							
programa							
proyecto							

31. Las ACL son las siguientes:

Notas del PPP	gmw:RW; *:R
programa	asw:RWX; desarrollo:RX; *:R
proyecto	asw:RW; usuarios:RW

Supongamos que*significa todo.

32. Si*asw*quiere permitir*GMV*pero ningún otro miembro de los usuarios puede ver splash.gif y modificar la ACL paraasw:RW; desarrollo:R; gmw:R.

33. Los navegadores existentes vienen precargados con las claves públicas de varios terceros de confianza, como Verisign Corporation. Su negocio consiste en verificar las claves públicas de otras empresas y crear certificados para ellas. Estos certificados están firmados, por ejemplo, con la clave privada de Verisign. Dado que la clave pública de Verisign está integrada en el navegador, los certificados firmados con su clave privada se pueden verificar.

34. Simplemente se introduce en la matriz dos veces. En

el ejemplo que se da en el texto, printer1 está en dos dominios simultáneamente. No hay problema aquí.

35. Para que un archivo sea legible para todos *excepto* Para una sola persona, las listas de control de acceso son la única posibilidad. Para compartir archivos privados, se pueden utilizar listas de control de acceso o capacidades. Para hacer públicos los archivos, las listas de control de acceso son las más fáciles, pero también es posible colocar una capacidad para el archivo o los archivos en un lugar conocido en un sistema de capacidades.

36. El servidor verificará que la capacidad sea válida y luego generará una capacidad más débil.
Esto es legal. Después de todo, el amigo puede simplemente regalar la capacidad.

capacidad que ya posee. Darle el poder de entregar algo aún más débil no es una amenaza para la seguridad. Si tiene la capacidad de entregar, por ejemplo, poder de lectura y escritura, entregar poder de solo lectura no es un problema.

37. No. Eso sería escribir, lo cual viola la ley.*propiedad.

38. No. Eso sería leer hacia arriba, lo cual viola la propiedad de seguridad simple.

39. Un proceso que escribe en otro proceso es similar a un proceso que escribe en un archivo. En consecuencia,*La propiedad debería ser válida. Un proceso podría escribir hacia arriba, pero no hacia abajo. Proceso*B*Podría enviar a*do,D*, y*mi*, pero no a*A*.

40. En la fotografía original, los ejes R, G y B permiten valores integrales discretos de 0 a 255, inclusive. Esto significa que hay 224puntos válidos en el espacio de color que un píxel puede ocupar. Cuando se quita 1 bit para el canal encubierto, solo se permiten los valores pares (suponiendo 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 buena. En total, solo se pueden representar 1/8 de los colores. Los colores no permitidos se asignan al color adyacente cuyos valores son todos números pares, por ejemplo, los colores (201, 43, 97), (201, 42, 97), (200, 43, 96) y (200, 42, 97) ahora se asignan todos al punto (200, 42, 96) y ya no se pueden distinguir.

41. La imagen contiene 1.920.000 píxeles. Cada píxel tiene 3 bits que se pueden utilizar, dada una capacidad bruta de 720.000 bytes. Si se duplica efectivamente debido a la compresión del texto antes de almacenarlo, la imagen puede contener texto ASCII que ocupa aproximadamente 1.440.000 bytes antes de la compresión. Por lo tanto, una sola imagen puede contener un disquete entero de datos ASCII. No hay expansión debido a la esteganografía. La imagen con los datos ocultos tiene el mismo tamaño que la imagen original. La eficiencia es del 25%. Esto se puede ver 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 lo tanto, por cada píxel de 24 bits,

se están codificando efectivamente 6 bits de texto ASCII.

42. Los disidentes podrían firmar los mensajes utilizando una clave privada y luego intentar difundir ampliamente su clave pública. Esto podría ser posible si alguien la sacara de contrabando del país y luego la publicara en Internet desde un país libre.

43.A continuación se muestran dos programas en C que realizan el trabajo. Se ejecutan en el mismo directorio pero en diferentes ventanas. Primero se inicia la decodificación.

```
/* codificador encubierto - Escrito por Albert S. Woodhull
el 3 de diciembre de 2000
  Versión C, utilizando permisos de archivo

  Este programa espera leer una cadena de caracteres
  ASCII '0' y '1' desde su entrada estándar. Genera
  una salida encubierta alternando
```

```
 hacer que el propietario de un archivo sea legible o
 no, controlando el tiempo que el archivo está en
 cada estado.

 Cada "tiempo de bit" consta de tres intervalos de
 tiempo. El intervalo intermedio determina el valor
 del bit. Para indicar un "0", el archivo es legible
 durante los dos primeros intervalos y no lo es
 durante el último intervalo.
 Para indicar un "1", el archivo es legible durante
 el primer intervalo y no lo es durante el segundo y
 el tercer intervalo.

 Un colaborador puede determinar la legibilidad de
 un archivo para su propietario incluso si el propio
 colaborador no tiene acceso, siempre que el archivo
 esté en un directorio que pueda leerse.

 La llamada al sistema de suspensión se utiliza para
 controlar el tiempo. Esto hace que
 Todo el proceso es bastante lento, ya que no puedes dormir
 menos de 1 segundo.
* /

# definir MAX 80

# incluir
<stdio.h> #
incluir
<fcntl.h>

int principal(vacío)
{
 entero c;
 int i
 = 0;
 int n
 = 0;
 entero;
 carácter s[MAX];

 /* obtiene la cadena de entrada, guarda solo
 '0' y '1', cuenta los caracteres */ while (((c
 = getchar()) != EOF) && (n < MAX))
  si (( c == '0') || (c == '1')) s[n++] = c;
 s[n] = ' ';

 /* crea el archivo de señal */
 fd = creat("/tmp/tmp000", 0600);
```

```
/* para cada '0' o '1' ejecuta la secuencia
correspondiente */ while (i != n)
{
 c = s[i++];
 chmod("/tmp/tmp000", 0);
```

```
    interruptor(c)
    {
     caso '0':
      dormir(2);
      chmod("/tmp/tmp000",
      0400); dormir(1);
      romper;
     caso '1':
      dormir(1);
      chmod("/tmp/tmp000",
      0400); dormir(2);
      romper;
    }
   }

   /* deshacerse de la evidencia
   */ unlink("/tmp/tmp000");
  }

  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  - - - - - - - - - - - - - - - - - - -


  /* decodificador encubierto - Escrito por Albert S. Woodhull
  el 3 de diciembre de 2000
   Versión C, utilizando permisos de archivo.

   Este programa comprueba repetidamente los permisos de
   un archivo. El codificador complementario cambia
   alternativamente el bit legible del propietario de 0 a
   1 de forma temporizada. Una transición de legible a
   ilegible señala el comienzo de un bit, el momento de
   la transición de regreso a legible señala el valor del
   bit. Si el tiempo ilegible es mayor que
   el tiempo legible el bit es un cero, si es más corto el bit
   es un 1.
  * /

  # incluir <stdio.h>
  # incluir <sys/stat.h>


  # define VERDADERO  1
  # define FALSO 0

  int principal(vacío)
  {
   estructura stat statbuf;
   int decodificación, c0, c1, modo, p0, p1;
```

°

```
/* Iniciar el decodificador antes de iniciar el
codificador. Este bucle espera
 el archivo a crear. */
mientras (stat("/tmp/tmp000", &statbuf) < 0) /* no hacer nada
*/ ;

modo = statbuf.st3modo;

/* Este bucle detecta el comienzo del
primer bit. */ while ((mode & S3(IRUSR)
!= 0 ) {

 stat("/tmp/tmp000",
 &statbuf); modo =
 statbuf.st3modo;
}

decodificación = VERDADERO;
mientras (decodificación == VERDADERO) {

 c0 = c1 = 0;

 /* fase 0, usa c0 para
 contar */ p0 = TRUE;
 mientras (p0 == VERDADERO) {

  /* refrescar statbuf y verificar el
  final de la señal */ if
  (stat("/tmp/tmp000", &statbuf) < 0)
   p0 = FALSO;
  modo = statbuf.st3modo;
  si ((modo & S3IRUSR) == 0) c0++; de lo contrario p0 = FALSO;
 }

 /* fase 1, usa c1 para
 contar */ p1 = TRUE;
 mientras (p1 == VERDADERO) {

  /* refrescar statbuf y verificar el
  final de la señal */ if
  (stat("/tmp/tmp000", &statbuf) <0) p1 =
  FALSE; mode = statbuf.st3modo;
  si ((modo & S3IRUSR) != 0) c1++; de lo contrario p1 = FALSO;
 }

 /* decide el valor del bit y lo
 muestra */ if (c0 > c1)
 printf("0"); else printf("1");

 /* hacer visible la salida ahora */
```

```
    flush(NULO);

    /* ver si la señal sigue ahí */
    if (stat("/tmp/tmp000", &statbuf)
    < 0) {

     decodificación = FALSO;
     putchar('0);
    }
   }
  }
```

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 10

1. El proceso de llamada debe colocar el número de llamada del sistema en un registro o en la pila.

2. Los archivos que se listarán son:*pez macabí, pez cuacker, caballito de mar*, y*comadreja*.

3. Imprime el número de líneas del archivo.*XYZ*que contienen la cadena "nd" en ellos.

4. El pipeline es el siguiente:

```
cara -8 z | cola -1
```

La primera parte selecciona las primeras ocho líneas de*el*y se los pasa a*cola*, que simplemente escribe el último en la pantalla.

5. Están separados, por lo que la salida estándar se puede redirigir sin afectar el error estándar. En una canalización, la salida estándar puede ir a otro proceso, pero el error estándar sigue escribiendo en la terminal.

6. Cada programa se ejecuta en su propio proceso, por lo que se inician seis procesos nuevos.

7. Sí. La memoria del niño es una copia exacta de la del padre, incluida la pila. Por lo tanto, si las variables de entorno estaban en la pila del padre, también estarán en la pila del niño.

8. Como los segmentos de texto se comparten, solo se deben copiar 36 KB. puede copiar 80 bytes por microsegundo, por lo que 36 KB tardan 0,46 ms.

La máquina
Añade otro 1

Se necesitan aproximadamente 1,46 ms para entrar y salir del kernel, y todo el proceso demora aproximadamente 1,46 ms.

9. El niño puede cambiar algunas variables y luego salir. Contenedor,Se garantiza a los padres que nada de lo que haga el niño afectará la dirección de los padres.

espacio. Contenedor,Esta garantía ya no es válida, lo que introduce la posibilidad de errores difíciles de encontrar.

10. CadaΔIEl uso de la CPU se divide a la mitad, por lo que después de un intervalo es 10, luego 5, 2, 1 y 0. Se necesitan 5ΔIpara golpear 0.

11. Sí. Ya no puede ejecutarse, por lo que cuanto antes su memoria vuelva a la lista libre, mejor.

12. Las señales son como las interrupciones de hardware. Un ejemplo es la señal de alarma, que indica que el proceso se realizará en un número específico de segundos en el futuro. Otro es la señal de excepción de punto flotante, que indica una división por cero o algún otro error. También existen muchas otras señales.

13. 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 consistiera en un bucle que enviara una señal al proceso con PID.*i*a pesar de *i*Desde 1 hasta el PID máximo. Muchos de estos procesos no estarían preparados para la señal y serían eliminados por ella. Si desea eliminar sus propios procesos, está bien, pero eliminar los procesos de su vecino no es aceptable.

14. Sería imposible con UNIX o Windows 2000, pero el hardware Pentium lo hace posible. Lo que se necesita es utilizar las características de segmentación del hardware, que no son compatibles con UNIX ni Windows 2000. 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.

15. Generalmente, los daemons se ejecutan en segundo plano y realizan tareas como imprimir y enviar correos electrónicos. Como las personas no suelen estar sentadas en el borde de sus sillas esperando a que terminen, se les da una prioridad baja, lo que consume tiempo de CPU que no necesitan los procesos

interactivos.

16. Un PID debe ser único. Tarde o temprano, el contador volverá a cero y luego ascenderá hasta, por ejemplo, 15. Si sucede que el proceso 15 se inició hace meses, pero aún está en ejecución, 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.

17. Cuando el proceso finaliza, se le otorgará al padre el estado de salida de su hijo. El PID es necesario para poder identificar al padre y poder transferir el estado de salida al proceso correcto.

18. Si todos los*intercambio*3*banderas*Los bits están configurados, elclonLa llamada inicia un hilo convencional. Si se borran todos los bits, la llamada es esencialmente unatenedor.

19. El valor de 1000 es completamente arbitrario. El único requisito es que cada subproceso en tiempo real obtenga una bondad mayor que cada subproceso de tiempo compartido.

20. Para cargar el sistema operativo es necesario comprender el formato del sistema de archivos, poder buscar en el directorio raíz y poder interpretar el formato binario ejecutable en el que se almacena el sistema operativo en el disco. Eso es pedir demasiado a un programa de arranque de 512 bytes. Apenas puede cargar el programa de arranque desde una ubicación fija, pero puede ser un programa largo, con un conocimiento detallado del sistema de archivos, el directorio y los formatos binarios.

21. 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 necesaria es de 370 KB. Sin texto compartido, cada programa necesita 190 KB, por lo que tres de ellos necesitan un total de 570 KB.

22. Sí. Con un campo de 16 bits, puede haber un máximo de 64 000 entradas de mapa de núcleo, por lo tanto, un máximo de 64 000 marcos de página. Por lo tanto, no hay forma de manejar memorias mayores a 64 MB. Cuando salió el VAX original, 2 MB se consideraba una memoria muy grande y 64 MB era, en efecto, infinito. Ahora, este límite de 64 MB se ha vuelto bastante evidente.

23. El segmento de texto no puede cambiar, por lo que nunca debe paginarse. Si se necesitan sus marcos, simplemente se pueden abandonar. Las páginas siempre se pueden recuperar del sistema de archivos. El segmento de datos no se debe paginar de nuevo al archivo ejecutable, porque es probable que haya cambiado desde que se incorporó. Paginarlo de nuevo arruinaría el archivo ejecutable. El segmento de pila ni siquiera está presente en el archivo ejecutable.

24. Dos procesos podrían asignar el mismo archivo a sus espacios de direcciones al mismo tiempo. Esto les brinda una forma de compartir la memoria física. La mitad de la memoria compartida podría usarse como

búfer desde *A* a *B* y la otra mitad como amortiguador *B* a *A* Para comunicarse, un proceso escribe un mensaje en su parte de la memoria compartida y luego envía una señal al otro para indicar que hay un mensaje esperándolo. La respuesta podría utilizar el otro búfer.

25. 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.

26. Originalmente, se asignaron cuatro páginas del archivo: 0, 1, 2 y 3. La llamada tiene éxito y, una vez realizada, solo quedan asignadas las páginas 2 y 3, es decir, los bytes 16 384 a 32 767.

27. Es posible. Por ejemplo, cuando la pila crece más allá de la página inferior, se produce un error de página y el sistema operativo normalmente le asigna la página inmediatamente inferior. Sin embargo, si la pila ha topado con el segmento de datos, no se puede asignar la siguiente página a la pila, por lo que el proceso debe finalizar. Incluso si hay otra página disponible en la memoria virtual, el área de paginación de

El disco podría estar lleno, lo que haría imposible asignar almacenamiento de respaldo para la nueva página, lo que también finalizaría el proceso.

28. Es posible si los dos bloques no son compañeros. Considere la situación de la figura 10-0(e).

Dos nuevas solicitudes llegan para 8 páginas cada una. En este punto, las 32 páginas inferiores de la memoria son propiedad de 4 usuarios diferentes, cada uno con 8 páginas. Ahora los usuarios 1 y 2 liberan sus páginas, pero los usuarios 0 y 3 retienen las suyas. Esto produce una situación con 8 páginas utilizadas, 8 páginas libres, 8 páginas libres y 8 páginas utilizadas. Tenemos dos bloques adyacentes de igual tamaño que no se pueden fusionar porque no son compañeros.

29. La paginación a una partición permite el uso de un dispositivo sin formato, sin la sobrecarga que supone utilizar estructuras de datos del sistema de archivos. Para acceder a un bloque*norte*El sistema operativo puede calcular la posición del disco simplemente agregándolo al bloque inicial de la partición. No es necesario recorrer todos los bloques indirectos que de otra manera serían necesarios.

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

31. Los resultados son los siguientes:

(a) Se concede el bloqueo.
(b) Se concede el bloqueo.
(do)*do*está bloqueado porque los bytes 20 a 30 no están disponibles.
(d)*A*está bloqueado porque los bytes 20 a 25 no están disponibles.
(mi)*B*está bloqueado porque el byte 8 no está disponible para bloqueo exclusivo.

En este punto, nos encontramos en un punto muerto. Ninguno de los procesos podrá volver a ejecutarse.

32. Se plantea el problema de qué proceso obtiene el

bloqueo cuando está disponible. La solución más sencilla es dejarlo sin definir. Esto es lo que hace POSIX porque es el más fácil de implementar. Otra es exigir que los bloqueos se concedan en el orden en que se solicitaron. Este enfoque implica más trabajo de implementación, pero evita la inanición. Otra posibilidad es dejar que los procesos proporcionen una prioridad al solicitar un bloqueo y utilizar estas prioridades para tomar una decisión.

33. Un enfoque es dar un error y negarse a llevar a cabo laBusco.Otra forma es hacer que el desplazamiento sea negativo. Mientras no se utilice, no se produce ningún daño. Solo si se intenta leer o escribir el archivo, debería aparecer un mensaje de error. Si elbusco es seguido por otrobuscoEsto hace que el desplazamiento sea positivo, no se da ningún error.

34. El propietario puede leerlo, escribirlo y ejecutarlo, y todos los demás (incluido el grupo del propietario) pueden leerlo y ejecutarlo, pero no escribirlo.

35. Sí. Cualquier dispositivo de bloques capaz de leer y escribir un bloque arbitrario puede utilizarse para albergar un sistema de archivos. Incluso si no hubiera forma de buscar un bloque específico, siempre es posible rebobinar la cinta y luego avanzar hasta el bloque solicitado. Un sistema de archivos de este tipo no sería un sistema de archivos de alto rendimiento, pero funcionaría. El autor ha hecho esto en un PDP-11 utilizando DECtapes y funciona.

36. No. El archivo sigue teniendo un único propietario. Si, por ejemplo, solo el propietario puede escribir en el archivo, la otra parte no puede hacerlo. Vincular un archivo a su directorio no le otorga de repente derechos que no tenía antes. Simplemente crea una nueva ruta para acceder al archivo.

37. Cuando se cambia el directorio de trabajo, se utiliza elchdirEn la llamada al sistema, se obtiene el inodo para el nuevo directorio de trabajo y se guarda en la memoria, en la tabla de inodos. El inodo para el directorio raíz también está allí. En la estructura de usuario, se mantienen punteros a ambos. Cuando se debe analizar un nombre de ruta, se inspecciona el primer carácter. Si es un ''/'', se utiliza el puntero al inodo raíz como punto de partida; de lo contrario, se utiliza el puntero al inodo del directorio de trabajo.

38. El acceso al i-nodo del directorio raíz no requiere acceso a disco, por lo que tenemos lo siguiente:

1. Lectura del*/*directorio para buscar "usr".
2. Lectura en el i-nodo para*/usuario*.
3. Lectura del*/usuario*directorio para buscar "ast".
4. Lectura en el i-nodo para*/usuario/ast*.
5. Lectura del*/usuario/ast*directorio para buscar "trabajo".
6. Lectura en el i-nodo para*/usuario/ast/trabajo*.
7. Lectura del*/usuario/ast/trabajo*directorio para buscar ''f''.
8. Lectura en el i-nodo para*/usuario/ast/trabajo/f*.

En total, se necesitan ocho accesos al disco antes de que el i-nodo necesario esté en la memoria.

39. El i-nodo contiene 10 direcciones. El bloque indirecto simple contiene 256. El bloque indirecto doble lleva a 65.536 y el indirecto triple lleva a 16.777.216, lo que da un total de 16.843.018 bloques. Esto limita el

tamaño máximo de archivo a 10 + 256 + 65.536 + 16.777.216 bloques, lo que equivale a unos 16 gigabytes.

40. Cuando se cierra un archivo, el contador de su i-nodo en la memoria se reduce. Si es mayor que cero, el i-nodo no se puede eliminar de la tabla porque el archivo aún está abierto en algún proceso. Solo cuando el contador llega a cero se puede eliminar el i-nodo. Sin el recuento de referencia, el sistema no sabría cuándo eliminar el i-nodo de la tabla. Hacer una copia separada del i-nodo cada vez que se abre el archivo no funcionaría porque los cambios realizados en una copia no serían visibles en las otras.

41. Los accesos a la memoria caché del búfer son tan poco frecuentes en comparación con los accesos a la memoria que el costo de administrar la cola LRU en el software es aceptable. Con las páginas, normalmente hay una o dos referencias por instrucción, lo que hace que mantener una cola LRU sea poco práctico.

42. Al forzar el contenido del caché del búfer a salir al disco cada 30 segundos, el daño causado por un bloqueo se limita a 30 segundos.*actualizar*Si no se ejecutaba, un proceso podría escribir un archivo y luego salir con todo el contenido del archivo todavía en la memoria caché del búfer. De hecho, el usuario podría cerrar la sesión y volver a casa con el archivo todavía en la memoria caché del búfer. Una hora después, el sistema podría bloquearse y perder el archivo, que todavía estaría solo en la memoria caché del búfer y no en el disco. Al día siguiente no tendríamos un usuario satisfecho.

43. Todo lo que tiene que hacer es establecer el número de enlaces en 1, ya que solo una entrada de directorio hace referencia al i-nodo.

44. Generalmente esobtenerpid, obteneruid, obtenergid,o algo así. Todo lo que hacen es buscar un entero de un lugar conocido y devolverlo. Cada llamada hace más.

45. El archivo se elimina simplemente. Esta es la forma normal (en realidad, la única forma) de eliminar un archivo.

46. Un disquete de 1,44 MB puede contener 1440 bloques de datos sin procesar. El bloque de arranque, el superbloque, el bloque de descripción de grupo, el mapa de bits de bloque y el mapa de bits de i-nodo de un sistema de archivos ext2 utilizan cada uno 1 bloque. Si se crean 8192 i-nodos de 128 bytes, estos i- nodos ocuparían otros 1024 bloques, lo que dejaría solo 411 bloques sin utilizar.

Se necesita al menos un bloque para el directorio raíz, lo que deja espacio para 410 bloques de datos de archivos. En realidad, Linux*mkfs*El programa es lo

suficientemente inteligente como para no crear más inodos de los que se pueden utilizar, por lo que la ineficiencia no es tan grave. De forma predeterminada, se crearán 184 inodos que ocupan 23 bloques. Sin embargo, debido a la sobrecarga del sistema de archivos ext2, Linux normalmente utiliza el sistema de archivos MINIX en disquetes y otros dispositivos pequeños.

47. A menudo es esencial contar con alguien que pueda hacer cosas que normalmente están prohibidas. Por ejemplo, un usuario inicia un trabajo que genera una cantidad infinita de resultados. Luego, el usuario cierra la sesión y se va de vacaciones durante tres semanas a Londres. Tarde o temprano, el disco se llenará y el superusuario tendrá que finalizar manualmente el proceso y eliminar el archivo de salida. Existen muchos otros ejemplos similares.

48. Probablemente alguien tenía el archivo abierto cuando el profesor cambió los permisos. El profesor debería haber eliminado el archivo y luego haber colocado otra copia de su archivo maestro en el directorio público. Además, debería haber utilizado un método mejor para distribuir archivos, como una página web, pero eso está fuera del alcance de este ejercicio.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 11

1. Una ventaja es que todo está en un solo lugar, lo que facilita su búsqueda. Una desventaja es que un bloque de disco defectuoso en el índice de nivel superior de una colmena puede causar un desastre en todo el sistema.

2. El HAL es simple y directo. Incluir el ratón, el disco y todos los demás controladores de dispositivos lo volvería difícil de manejar y destruiría su función como una capa delgada que oculta ciertas diferencias básicas del hardware del propio ordenador, pero no los dispositivos de E/S.

3. A una base de datos genealógica le puede resultar conveniente registrar las fechas de nacimiento y muerte de los antepasados utilizando el formato de hora estándar del sistema. De hecho, cualquier base de datos histórica podría utilizarlo.

4. Los DPC se ejecutan en un contexto arbitrario. Los APC se ejecutan en el contexto de un subproceso específico. El acto de señalar el proceso implica escribir un marco en la pila del usuario para que se pueda devolver la señal. Esto requiere acceso al espacio de direcciones del usuario. En consecuencia, se necesita un APC.

5. (a) El administrador de procesos utiliza el administrador de objetos para crear subprocesos.

(b) El administrador de memoria utiliza el administrador de seguridad para ver si se puede asignar un archivo.

(c) El administrador plug-and-play utiliza el administrador de configuración para registrar un nuevo dispositivo.

6. Una señal es manejada por un nuevo hilo en el contexto de algún proceso. Por ejemplo, cuando se presiona la tecla Salir o incluso cuando falla un hilo. Realmente no tiene sentido capturar una señal en el contexto de un hilo. Realmente tiene que ser por proceso. Por lo tanto, el manejo de señales es

realmente una actividad por proceso.

7. Tendría más sentido en servidores. Las máquinas cliente tienen menos procesos simultáneos. Las bibliotecas compartidas solo tienen sentido si hay varios procesos que las comparten. De lo contrario, es más eficiente vincular estáticamente las bibliotecas y aceptar la duplicación. La ventaja de la vinculación estática es que solo se cargan los procedimientos que realmente se necesitan. Con las DLL puede haber procedimientos en la memoria que nadie esté usando.

8. Las funciones en *ntdll.dll* son las que necesitan los subsistemas, es decir, las llamadas al sistema. Las funciones en *ntoskrnl.exe* Son funciones exportadas que utilizan los controladores y otras partes del sistema operativo. No existe ninguna relación entre ellas. El hecho de que sean tan parecidas es solo una coincidencia. Fácilmente podrían haber diferido en un factor de cinco.

9. No. Los 3 bits de orden inferior del puntero de objeto en el controlador se utilizan para indicadores, como se indica en el texto. Estos deben enmascararse antes de que se pueda utilizar el puntero.

seguido. En consecuencia, cada puntero tiene 3 bits cero como bits de orden inferior. Esto significa que los encabezados de objetos deben comenzar en direcciones que sean múltiplos de 8 bytes.

10. Hay un límite de 32 operaciones porque solo hay 32 bits de derechos en el identificador del objeto.

11. 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 de usuario (pero tienen un semáforo de respaldo cuando se necesita bloquear). El administrador de objetos no sabe sobre ellos y no tienen controladores, como se indicó en el texto.Esperar varios objetoses una llamada al sistema, el sistema no puede realizar una operación booleana OR de varias cosas, una de las cuales desconoce. La llamada debe ser una llamada al sistema porque los semáforos y los mutex son objetos del núcleo. En resumen, no es posible tener ninguna llamada al sistema que mezcle objetos del núcleo y objetos del usuario de esta manera. Tiene que ser uno o el otro.

12. (a) El último hilo sale.
(b) Se ejecuta un hiloProceso de salida.
(c) Otro proceso con un identificador para este lo mata.

13. El quantum de Windows 2000 Professional es de 20 ms y hay 12 subprocesos con prioridades superiores a 3. Por lo tanto, el primer subproceso con prioridad 3 debe esperar 240 ms hasta que pueda intentarlo.

14. Como máximo, unos pocos microsegundos. Interrumpe el hilo actual inmediatamente. Es solo una cuestión de cuánto tiempo lleva ejecutar el código del despachador para realizar el cambio de hilo.

15. Reducir su prioridad por debajo de la prioridad base podría usarse como castigo por usar un tiempo de CPU u otros recursos excesivo.

16. El procesador no permitirá que se ejecuten dichas instrucciones en modo usuario y las detectará como errores. En principio, un filtro que lee un programa binario y reemplaza todos losENyAFUERASe podrían escribir instrucciones con llamadas a rutinas que

utilicen servicios legales del sistema operativo. Alternativamente, se podría interpretar el programa y manejar estas instrucciones haciendo llamadas al sistema Windows 2000 para realizar la E/ S.

17. Una forma de hacerlo es aumentar la prioridad de los procesos importantes. Una segunda forma de hacerlo es otorgarles cuantos más largos.

18. El problema no se puede resolver utilizando tablas de páginas. Las tablas de páginas se asignan entre direcciones virtuales y físicas. El problema aquí es que el procedimiento compartido se coloca en diferentes direcciones virtuales. Si una instrucción en la dirección 65,536 leeJMP300, saltará a una memoria no válida (por debajo de 64 KB) y generará una trampa. Si está parcheado para leerJMP65836que avanzará 300 bytes, lo cual es

correcto para ese proceso. Pero si un proceso diferente tiene esta instrucción en, digamos, la dirección 131.072, saltar a 65836 es incorrecto. No importa en qué parte de la memoria física se encuentre la página. Para el segundo proceso, la CPU está generando una dirección virtual incorrecta. Ninguna configuración de las tablas de páginas puede asignar correctamente una dirección virtual incorrecta.

19. El Pentium tiene varios segmentos, cada uno de los cuales comienza en la dirección virtual 0. Cada uno *.dll*en finSe podría asignar un segmento a un segmento independiente que comience en la dirección virtual 0 de ese segmento. Para ello, Windows 2000 debería soportar la segmentación.

20. Sí. Los VAD son la forma en que el administrador de memoria realiza un seguimiento de las direcciones que están en uso y las que están libres. Se necesita un VAD para una región reservada a fin de evitar que un intento posterior de reservarla o asignarla tenga éxito.

21. (1) es una decisión de política sobre cuándo y cómo recortar un conjunto de trabajo. (2) y (3) son obligatorios. (4) es una decisión de política sobre cuán agresivamente escribir páginas sucias en el disco. (5) y (6) son obligatorios. (7) no es realmente una cuestión de política ni obligatorio; el sistema nunca tiene que poner páginas en cero, pero si el sistema está inactivo, poner páginas en cero siempre es mejor que simplemente ejecutar el bucle inactivo.

22. No se mueve en absoluto. Una página solo va a una de las listas cuando no está presente en ningún conjunto de trabajo. Si todavía está en un conjunto de trabajo, no va a ninguna de las listas libres.

23. No puede ir a la lista modificada, ya que contiene páginas que aún están asignadas y podrían ser rechazadas. Una página no asignada no está en esa categoría. Ciertamente no puede ir directamente a la lista libre porque esas páginas pueden abandonarse a voluntad. Una página sucia no puede abandonarse a voluntad. En consecuencia, primero debe volver a

escribirse en el disco, luego puede ir a la lista libre.

24. Copiar una palabra requiere una lectura y una escritura, lo que supone un total de 20 ns para mover 4 bytes. Esto supone 5 ns por byte copiado. Una pantalla XGA tiene 1024×768×3 bytes, que tardan aproximadamente 11,8 mseg en copiarse en el mejor de los casos (suponiendo que el programa se ejecuta completamente fuera del caché L1).

25. 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)

26. El hecho de que el bloque 66 sea contiguo a una ejecución existente no ayuda, ya que los bloques no están en un orden lógico de archivos. En otras palabras, utilizar el bloque 66 como nuevo bloque no es mejor que utilizar el bloque 90. Las entradas en la MFT son:

0, 8: (4, 20), (2, 64), (3, 80), (1, 66)

27. Es un accidente. Los 16 bloques aparentemente se comprimieron en 8 bloques. Podrían haber sido 9 u 11 con la misma facilidad.

28. Se pueden eliminar todos, excepto el SID del usuario, sin afectar la solidez de la seguridad.

SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 12

1. Las mejoras en el hardware de las computadoras se han debido en gran medida a los transistores más pequeños. Algunos factores que pueden limitar esto son: (a) las propiedades ondulatorias de la luz pueden limitar las técnicas fotolitográficas convencionales para producir circuitos integrados,
(b) la movilidad de los átomos individuales en los sólidos puede conducir a la degradación de las propiedades de capas muy delgadas de semiconductores, aislantes y conductores, y (c) la radiactividad de fondo puede alterar los enlaces moleculares o afectar a cargas almacenadas muy pequeñas. Sin duda, hay otros factores.

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

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

4. No. La diferencia se relaciona más con el hecho de que los servidores DNS almacenan en caché y están organizados jerárquicamente. Las rutas podrían haberse dado fácilmente en orden descendente, pero ahora está bien establecida la convención de hacerlo al revés.

5. Probablementeestadísticaes redundante. Podría lograrse mediante una combinación deabierto, fstat,ycerca.Sería muy difícil simular cualquiera de los otros.

6. Si los controladores se colocan debajo de los

subprocesos, no pueden ser subprocesos independientes al estilo de MINIX. Deben ejecutarse como parte de algún otro subproceso, más al estilo de UNIX.

7. Es posible. Lo que se necesita es un proceso a nivel de usuario, el servidor de semáforos. Para crear un semáforo, un usuario le envía un mensaje solicitando un nuevo semáforo.
Para utilizarlo, el proceso de usuario pasa la identidad del semáforo a otros procesos. Estos pueden enviar mensajes al servidor de semáforos solicitando una operación. Si la operación se bloquea, no se envía ninguna respuesta, bloqueando así al que realiza la llamada.

8. El patrón es de 8 ms de código de usuario, luego 2 ms de código de sistema. Con la optimización, cada ciclo ahora es de 8 ms de código de usuario y 1 ms de código de sistema. Por lo tanto, el ciclo se reduce de 10 ms a 9 ms. Al multiplicar por 1000 estos ciclos, un programa de 10 segundos ahora toma 9 segundos.

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

10. Los nombres externos pueden ser tan largos como se necesite y de longitud variable. Los nombres internos suelen tener 32 o 64 bits y siempre una longitud fija. Los nombres externos no necesitan ser únicos. 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 jerárquicos. Los nombres internos son generalmente índices en tablas y, por lo tanto, forman un espacio de nombres plano.

11. Si la nueva tabla es 2×Si es tan grande como la anterior, no se llenará rápidamente, lo que reducirá la cantidad de veces que se necesitará una tabla actualizada. Por otro lado, es posible que no se necesite tanto espacio, por lo que puede desperdiciar memoria. Este es un clásico dilema entre tiempo y espacio.

12. Sería arriesgado hacer eso. Supongamos que el PID estuviera en la última entrada. En ese caso, al salir del bucle se quedaría*pag*apuntando a la última entrada. Sin embargo, si no se encontró el PID,*pag*Puede terminar apuntando a la última entrada o a una posterior, dependiendo de los detalles del código compilado, qué optimizaciones están activadas, etc. Lo que podría funcionar con un compilador podría fallar con otro. Es mejor establecer una bandera.

13. Se podría hacer, pero no sería una buena idea. Un controlador IDE o SCSI tiene muchas páginas. Tener un código condicional tan largo hace que el código fuente sea difícil de seguir. Sería mejor colocar cada uno en un archivo separado y luego usar el*Archivo Make*para determinar cuál incluir. O, al menos, se podría utilizar la compilación condicional para incluir un archivo de controlador u otro.

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

15. No es fácil. Varias invocaciones al mismo tiempo podrían interferir entre sí. Podría ser posible si los datos estáticos estuvieran protegidos por un mutex, pero eso significaría que una llamada a un procedimiento simple podría quedar bloqueada inesperadamente.

16. Sí. El código se replica cada vez que se llama a la macro. Si se llama muchas veces, el programa será mucho más grande. Se trata de un equilibrio típico entre tiempo y espacio: un programa más grande y más rápido en lugar de un programa más pequeño y más lento. Sin embargo, en un caso extremo, el programa más grande podría no caber en la TLB, lo que provocaría que se tambaleara y, por lo tanto, se ejecutara más lento.

17. Comience por realizar una operación OR EXCLUSIVA entre los 16 bits inferiores y superiores de la palabra para formar un entero de 16 bits.*s*Para cada bit, hay cuatro casos: 00 (resulta en un 0), 01 (resulta en un 1), 10 (resulta en un 1) y 11 (resulta en un 0). Por lo tanto, si el número de 1 en*s*es impar, la paridad es impar; de lo contrario es par.

Crea una tabla con 65.536 entradas, cada una de las cuales contiene un byte con el bit de paridad. La macro se ve así:

```
# definir bits de paridad(w)[(w & 0xFFFF) ^ ((w>>16) & 0xFFFF)]
```

18. No, no, no, el valor del color "comprimido" sería tan grande como el original y además, se necesitaría una paleta de colores enorme. No tiene ningún sentido.

19. 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. Por lo tanto, con más de 384 píxeles, gana GIF. Una paleta de colores de 16 bits de ancho contiene 65.536 entradas de 3 bytes cada una, para un total de 196.608 bytes. El ahorro aquí es de 1 byte por píxel. Por lo tanto, con más de 196.608 píxeles, gana la compresión de 16 bits. Suponiendo una relación de aspecto de 4:3, el punto de equilibrio es una imagen de 512×384 píxeles. Para VGA (640×480), el color de 16 bits requiere menos datos que el color real de 24 bits.

20. En el caso de una ruta que se encuentra en la caché de nombres de ruta, no tiene ningún efecto porque el i-nodo se omite de todos modos. Si no se lee, no importa si ya está en la memoria. En el caso de una ruta que no se encuentra en la caché de nombres pero que implica un i-nodo anclado, la anclación sí ayuda, ya que elimina una lectura de disco.

21. Registrar la fecha de la última modificación, el tamaño y, posiblemente, una firma calculada, como una suma de comprobación o un CRC, puede ayudar a determinar si ha cambiado desde la última referencia. Una advertencia: un servidor remoto podría proporcionar información falsa sobre un archivo y podría ser necesaria la regeneración local de una firma calculada.

22. Se podría asignar un número de versión o una suma de comprobación al archivo y almacenar esta información junto con la pista. Antes de acceder a un archivo remoto, se realizaría una comprobación para asegurarse de que el número de versión o la suma de

comprobación sigan coincidiendo con el archivo actual.

23. Un sistema de archivos normalmente intentará escribir datos nuevos en el bloque de disco disponible más cercano después del último utilizado. Si se escriben dos archivos simultáneamente, esto puede provocar que se intercalen los bloques de datos en el disco, lo que hace que ambos archivos se fragmenten y, por lo tanto, sean más difíciles de leer. Este efecto se puede mejorar almacenando los datos en la memoria intermedia para maximizar el tamaño de las escrituras o escribiendo en archivos temporales y luego copiando cada salida a un archivo permanente cuando finaliza el programa.

24. Brooks hablaba de proyectos grandes en los que la comunicación entre los programadores hace que todo sea más lento. Ese problema no ocurre con un proyecto de una sola persona y, por lo tanto, la productividad puede ser mayor.

25. Si un programador puede producir 1000 líneas de código por un costo de $100,000, una línea de código cuesta $100. En el capítulo 11, dijimos que Windows 2000 consistía en

29 millones de líneas de código, lo que equivale a 2.900 millones de dólares. Parece una cantidad enorme. 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 al año.

26. Supongamos que la memoria cuesta 100 dólares por 64 MB (compruébelo con los precios actuales). Entonces, una máquina de gama baja necesita 1600 dólares en discos. Si el resto de la PC cuesta 500 dólares, el costo total asciende a 2100 dólares. Esto es demasiado caro para el mercado de gama baja.

27. Un sistema integrado puede ejecutar un solo programa o una pequeña cantidad de ellos. Si todos los programas se pueden mantener cargados en la memoria en todo momento, tal vez no sea necesario un administrador de memoria ni un sistema de archivos. Además, los controladores solo serían necesarios para unos pocos dispositivos de E/S y podría tener más sentido escribir los controladores de E/S como rutinas de biblioteca. Las rutinas de biblioteca también podrían compilarse mejor en programas individuales, en lugar de en bibliotecas compartidas, eliminando así la necesidad de bibliotecas compartidas. Probablemente se podrían eliminar muchas otras características en casos específicos.