

Julio 2022

Preguntas (V/F):

1. Algunos sistemas de E/S deshabilitan la cache selectivamente (En DMA)

Por E/S puede ser que modifiques datos directamente en memoria sin pasar por cache, caso que se puede dar en la E/S por asignación a memoria. En ese caso lo de cache es distinto a lo que hay en memoria, que es lo que realmente necesitas.

2. Sobre para qué sirve signal

Es una función que se encarga de registrar una señal en un manejador de señales. Se le pasa la función manejadora y la señal.

3. Sobre qué comparten los hilos

Los hilos de un mismo proceso comparten el mismo espacio de memoria, por tanto, dos hilos del mismo proceso pueden compartir estructuras de datos, variables, código, archivos abiertos...

4. La tabla de páginas invertida es siempre más pequeña que la de paginas normal

La tpi tiene una entrada por cada marco de página de memoria física. La tp normal suele tener más, pero no SIEMPRE. Normalmente son más pequeñas, cuando el tamaño de las direcciones virtuales es mayor que el de las físicas.

5. Lo de los fallo con lw

6. Una interrupcion siempre provoca que un proceso se bloquee

Falso, nunca se bloquea. Pasa a estado listo

7. El vector de interrupciones de los procesos (la de otros años)

Falso. Los procesos no tienen vector de interrupciones

8. Usuario real y efectivo

9. El Working Set de un proceso es mayor cuanto más se cumple el principio de localidad en su ejecución

10. Si espacio virtual = espacio físico hace falta traducción?

11. La MMU se encarga de traducir direcciones virtuales a físicas

12. Otra de espacio virtual = espacio físico que no recuerdo

13. El tamaño de los nodos i es mayor que el de la FAT equivalente.

Ejercicio 2. Funcionamiento de las interrupciones mediante DMA. Explicar el proceso de todo lo que hace comentando explícitamente el papel que juega el vector de interrupciones, el manejador (handler) de interrupciones, el despachador (dispatch) de llamadas al sistema y el manejador (handler) de llamadas al sistema.

Ejercicio 3 el mismo que Junio 2021

Había otro apartado que pedía indicar cómo quedaban las tablas de páginas tras escribir el contenido en una dirección a cuya página se accedía por primera vez.

Sistemas operativos I- Enero 2022

Preguntas (V/F):

14. Algunos sistemas de E/S deshabilitan la cache selectivamente (En DMA)

Por E/S puede ser que modifiques datos directamente en memoria sin pasar por cache, caso que se puede dar en la E/S por asignación a memoria. En ese caso lo de cache es distinto a lo que hay en memoria, que es lo que realmente necesitas.

15. Sobre para qué sirve signal

Es una función que se encarga de registrar una señal en un manejador de señales. Se le pasa la función manejadora y la señal.

16. Sobre qué comparten los hilos

Los hilos de un mismo proceso comparten el mismo espacio de memoria, por tanto, dos hilos del mismo proceso pueden compartir estructuras de datos, variables, código, archivos abiertos...

17. La tabla de páginas invertida es siempre más pequeña que la de paginas normal

La tpi tiene una entrada por cada marco de página de memoria física. La tp normal suele tener más, pero no SIEMPRE. Normalmente son más pequeñas, cuando el tamaño de las direcciones virtuales es mayor que el de las físicas.

18. La resolución de un fallo de paginas provoca una interrupción

Es obligatorio que se produzca ya que cuando hay un fallo de página se tiene que interrumpir el proceso para cargar los elementos necesarios.

19. Lo de los fallo con lw

20. Una interrupcion siempre provoca que un proceso se bloquee

Falso, nunca se bloquea. Pasa a estado listo

21. El vector de interrupciones de los procesos (la de otros años)

Falso. Los procesos no tienen vector de interrupciones

22. (Una más que no me acuerdo que era) Algo sobre usuario real y efectivo

Ejercicio 2:

Explicar el código de las interrupciones acompañado de explicar vector de interrupcion, manejador de interrupciones, dispatcher de llamadas al sistema y manejador de llamadas al sistema.

Ejercicio 3:

3. 2 puntos Contesta razonadamente a las siguientes cuestiones:

- a) Sea un sistema con palabras de 64 bits. Sus direcciones virtuales son de 48 bits y sus direcciones físicas de 32 bits. Las páginas son de 8KB y la unidad direccionable es la media palabra (es decir, 32 bits). Haz una estimación razonada del tamaño de la tabla de páginas.
- b) Si para este sistema se quiere usar un sistema páginado en dos niveles de modo que las tablas del segundo nivel quepan en una página (supón ahora que cada entrada de dicha tabla ocupa 64 bits). ¿Cuántos bits se deben asignar para direccionar la tabla del primer nivel? Supón que el sistema tiene 10 procesos activos, ¿Cuántas tablas de primer y segundo nivel hay en total como máximo?

Pero había que estimar también el tamaño de la RAM y el tamaño de la TLB. En el apartado b) solo preguntaba los bits para la tabla de primer nivel

Ejercicio 4:

Mapa de memoria (ponía una foto del mapa de memoria y la instrucción con la que se pasaba al hilo el parametro):

1. Decir donde estaba una variable local al hilo principal
2. Decir donde estaba la función main
3. Decir donde estaba el parámetro enviado al hilo

Instrucción creación del hilo: `pthread_create(&hilo1,NULL,funcion_hilo,(void *)(intptr_t)local);`

No había visto intptr_t en mi puta vida: lo que hace es que si local=3; si imprimo el %p del parametro enviado al hilo, este sea 0x03 creo.

Sistemas Operativos I - Junio 2021

NOS HAN FOLLAO...el texto no es así exactamente pero bueno..

Eran dos ejercicios. El primero 7 puntos, responder 14 de las 20 preguntas dependiendo del último número del dni:

1. Qué es el TRAP

Instrucción para hacer un "salto" al modo kernel del SO, por ejemplo, un syscall.

2. ¿Conviene usar un DMA en procesos donde se deseé que las operaciones de E/S sean rápidas?

No, si quieras hacer que las operaciones de E/S sean rápidas, usa la CPU que es más rápida. El DMA está para que los demás procesos se vean beneficiados al no sobrecargar a la CPU.

3. Algunos sistemas de E/S deshabilitan la cache selectivamente. Indica cual es y el motivo de hacerlo.

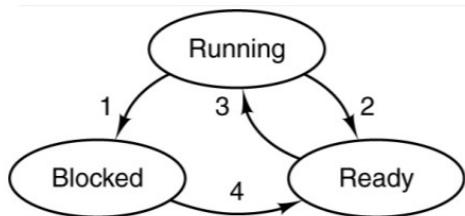
Por E/S puede ser que modifiques datos directamente en memoria sin pasar por cache, caso que se puede dar en la E/S por asignación a memoria. En ese caso lo de cache es distinto a lo que hay en memoria, que es lo que realmente necesitas.

4. puedo un proceso ser zombie y huérfano

No. Un proceso entra en estado zombie porque ha muerto, pero su padre aún está vivo. Tan pronto como muera el padre, el zombie, en lugar de quedar huérfano, pasará casi inmediatamente a ser "adoptado" por el proceso de PID 1, con lo que el zombie dejará de ser zombie y desaparecerá.

5. el diagrama de procesos añadir huérfano y zombie. (no daba el diagrama)

Un proceso se convierte en huérfano al morir su padre, independientemente de lo que esté haciendo. No tiene sentido dibujar un estado huérfano porque es independiente del estado del proceso. Zombie?? Igual.



6. que es el usuario efectivo

Es el usuario que cuenta con los permisos de acceso a los ficheros y directorios a los que se hacen referencia en el fichero ejecutable.

7. ¿Qué hace la función setUid?

setUid sirve para establecer el usuario efectivo. **real**

8. que es pause

Es una llamada al sistema que fuerza a un proceso a detenerse hasta que recibe una señal (bien sea de kill o de un temporizador).

9. que es signal, argumentos, que hace

Es una función que se encarga de registrar una señal en un manejador de señales. Se le pasa la función manejadora y la señal.

10. que es setuid, argumentos

Función que fija el usuario efectivo del proceso y su argumento es el entero que identifica a este usuario que se le quiere hacer efectivo.

11. si tienes 4MB de memoria y 4 hilos, cuanta memoria comparten

Los hilos comparten la memoria

12. ¿Por qué cada hilo debe tener su propia pila?

Porque se usa para almacenar las direcciones de retorno de llamadas a funciones, entonces como cada hilo tiene una secuencia de llamadas a funciones que no tiene que ver con las de los demás, pues cada uno necesita su propio stack.

13. ¿que es el modo núcleo

Modo en el que el sistema operativo tiene acceso completo a todo el hardware y puede ejecutar cualquier instrucción que la máquina sea capaz de ejecutar.

14. ¿Cómo funcionan los planificadores por sorteo?

Se dan boletos a cada proceso y se sortea a quien le toca. Cada proceso recibe un numero diferente de boletos.

15. ¿Es cierto que un planificador eficiente debe primar a los procesos con muchas operaciones E/S?

Sí, ¿por qué? Porque las operaciones que consumen la mayor parte del tiempo son operaciones de E/S. Creo que es más porque los procesos con mucho E/S aprovechan menos el quantum y liberan antes la CPU, entonces es bueno darles prioridad.

16. ¿El Working Set de un proceso es mayor cuanto más se cumple el principio de localidad en su ejecución?

Mientras mejor sea la localidad menor será el tamaño del working-set y por lo tanto menos memoria real necesitará el proceso para correr eficientemente.

17. ¿La cantidad de memoria que ocupa tanto una memoria FAT como los i-nodos dependen del tamaño del disco duro?

En FAT sí, depende del tamaño del disco duro. En los nodos-i no depende del tamaño del disco duro, salvo que a mayor tamaño de los nodos, menor espacio habrá en disco (como es de esperar).

18. ¿Qué hace la función clone?

La función clone realiza una tarea similar a fork, crear procesos hijos al proceso que lo ejecuta; pero además permite que el proceso hijo comparta partes de su contexto con el padre, como el espacio de memoria, la tabla de descriptores de fichero y la tabla de manejadores de señales. El uso que habitualmente se le da a clone es para implementar hilos de ejecución (threads).

19. ¿Tiene sentido que el tamaño del espacio físico y el virtual sean del mismo tamaño?

El tamaño de la memoria virtual y física son independientes, puede ser normal que sean iguales, aunque usualmente no lo sean. En multiprogramación se usa el espacio virtual para tener mismos espacios virtuales entre procesos y que compitan por la memoria principal.

20. Las instrucciones IN/OUT son necesarias para leer y escribir los puertos cuando la gestión de E/S es con espacios separados de E/S en memoria. ¿Cuál es el motivo?

Hacen falta porque no puedes usar lw y sw al ser espacios separados. Una posición cualquiera es necesario distinguir si es del mismo espacio de direcciones o de otro, y se usan para eso.

21. ¿Qué tiene que hacer la CPU para que un DMA haga una operación de lectura de un fichero en disco?

La CPU escribe en los registros especiales de la controladora de disco aquellos valores que necesita leer. Es decir, copias desde los registros del procesador a los de la controladora

. La CPU sigue a sus cosas, mientras la controladora de disco realiza todas las operaciones por hardware necesarias en el disco para copiar en sus sectores la información y almacenarla en el buffer. Cuando tiene todo almacenado, lanza una señal de interrupción a través del bus a la CPU, en donde la CPU la resuelve con el resolutor de interrupciones.

Ejercicio 2, 3 puntos:

2.1 Parecido a este de 2017 pero te pedía estimacion RAM, tabla de paginas, rango de comienzo y final TLB.

2.2 Te pedia TP1 , TP2

2.3 La pregunta de lw, cuantas veces accede minimo y maximo a memoria fisica, y cuantas veces accede minimo y maximo a disco duro.

3. 2 puntos Contesta razonadamente a las siguientes cuestiones:

- a) Sea un sistema con palabras de 64 bits. Sus direcciones virtuales son de 48 bits y sus direcciones físicas de 32 bits. Las páginas son de 8KB y la unidad direccionable es la media palabra (es decir, 32 bits). Haz una estimación razonada del tamaño de la tabla de páginas.
- b) Si para este sistema se quiere usar un sistema páginado en dos niveles de modo que las tablas del segundo nivel quepan en una página (supón ahora que cada entrada de dicha tabla ocupa 64 bits). ¿Cuántos bits se deben asignar para direccionar la tabla del primer nivel? Supón que el sistema tiene 10 procesos activos, ¿Cuántas tablas de primer y segundo nivel hay en total como máximo?

Sistemas Operativos I - Enero 2021

primera tanda: ej1: impresora; ej2: este; ej3: mismo que ej3 enero 2018.

segunda tanda: ej1: read en el kernel y tal; ej2: este; ej3: mismo que ej3 enero 2018.

8.

Preguntas de respuesta breve

- a. **Execv("/bin/ls") y un fork(). ¿Puede variar el número de procesos que se crean si el código asociado al execv no existe?**

Si ese código no existe, el execv da un error. Si da un error no hay cambio de imagen y el fork se ejecuta. Pero si ese código existe, sí hay un cambio de imagen y el fork no se ejecuta.

- b. **Un fallo de TLB significa que hay que ir copiar la página desde la memoria secundaria a un marco de la principal, lo que implica un gran retardo**

Falso, ya que al producirse un fallo de TLB significa que no se resuelve la traducción, por lo que se va a la tabla de páginas a buscar.

- c. **Cada proceso almacena su vector de interrupciones en la zona de kernel de la memoria, concretamente en su entrada de la tabla de procesos**

Falso, ya que los procesos no tienen vector de interrupciones.

- d. **Sleep y pause son lo mismo**

Falso.

Sleep: vuelve inactivo a un proceso por un determinado tiempo.

Pause: fuerza a un proceso a detenerse hasta que recibe una señal.

- e. **¿La tabla de páginas invertidas, son más pequeñas o grandes que las ordinarias?**

Normalmente son más pequeñas, cuando el tamaño de las direcciones virtuales es mayor que el de las físicas.

- f. **Los threads comparten el espacio de memoria virtual, el stack, los datos de cache y los registros.**

Falso, el stack y los registros no se comparten.

- g. (no recuerdo el comienzo de la pregunta pero tenía que ver con ficheros y usuarios :))... **Los usuarios real y efectivo habitualmente coinciden.**

CIERTO, no coinciden sólo cuando el archivo tiene activado el sticky bit.

El usuario real es el encargado de la ejecución del proceso mientras que el efectivo es el que determina el propietario del fichero, el acceso a este y la comprobación de permisos para el envío de señales. EUID y UID sólo coinciden siempre cuando el STICKY BIT no está activado, puesto que este bit precisamente sirve para cambiar los permisos del usuario que ejecuta, que pasa a tener los mismos que el usuario propietario.

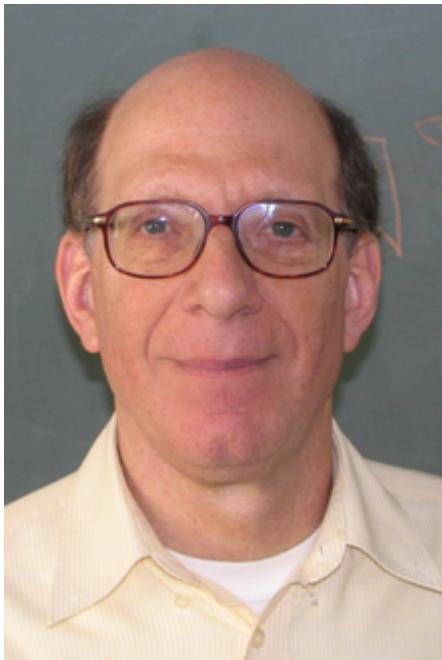
~~False, realmente no existe un usuario real y uno efectivo por fichero. Existe un usuario real y uno efectivo por cada fichero EJECUTABLE y solamente se usan cuando ese fichero ejecutable se convierte en proceso. Por eso, los usuarios real y efectivo están relacionados a los procesos no a los ficheros.~~

- g. **Las tablas de página invertidas son siempre más pequeñas que las tablas de página ordinarias.**

Falso, eventualmente son más pequeñas pero podrían ser más grandes. (no entiendo esta respuesta *verdadero* pero fue la que dijo rivera). Las tablas de página invertida tienen un tamaño concreto: tantas entradas como frames haya en la memoria física (tiene tantas filas como páginas físicas haya y es tan ancha como los bits necesarios para almacenar las parejas nroPagVirtual-nroPagFisico, identificativo de proceso y los bits de control). La tabla de páginas ordinaria tiene tantas entradas como páginas virtuales haya y el ancho es mucho más pequeño (tiene que guardar mucha menos información).

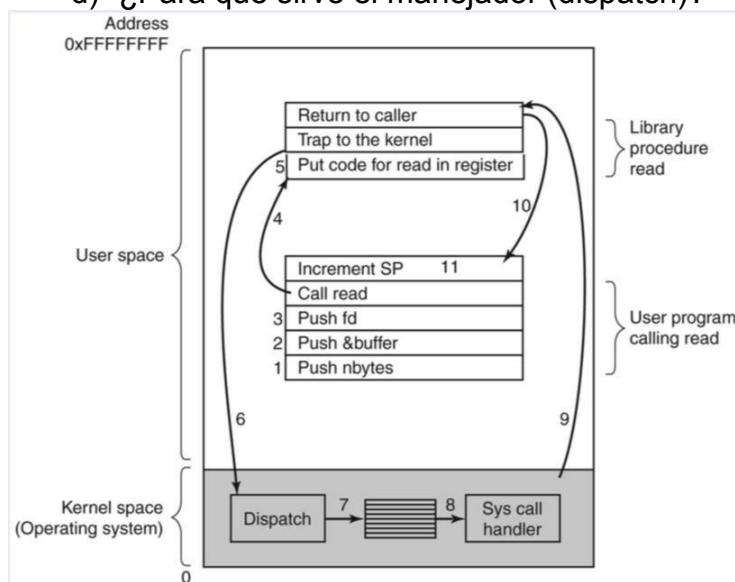
Atentamente, tus héroes anónimos.

Sistemas Operativos I - Enero 2018



<3

- 1) Indicar qué hace...
 - a) Paso 4.
 - b) Paso 5.
 - c) Paso 7.
 - d) ¿Para qué sirve el manejador (dispatch)?



Pasos para realizar la llamada al sistema *read()*: (páginas 50-52 Tanenbaum)

- **Pasos 1-2-3:** El programa llamador mete los parámetros en la pila.
- **Paso 4:** Llamada al procedimiento de biblioteca.
- **Paso 5:** El procedimiento de biblioteca coloca el número de la llamada al sistema en un lugar en el que el SO lo espera, como un registro.
- **Paso 6:** Se ejecuta una instrucción TRAP para cambiar del modo usuario al modo kernel y empezar la ejecución en una dirección fija dentro del núcleo.
- **Dispatch:** Examina el número de llamada al sistema.
- **Paso 7:** Se pasa la llamada al manejador, mediante una tabla de apuntadores a manejadores de

llamadas al sistema, indexados en base al número de llamada al sistema.

- **Paso 8:** Se ejecuta el manejador de llamadas al sistema
- **Paso 9:** El manejador termina su trabajo y el control se le regresa al procedimiento de biblioteca que está en espacio de usuario, en la instrucción que va después del TRAP.
- **Paso 10:** El procedimiento regresa el programa de usuario.
- **Paso 11:** El programa de usuario limpia la pila.
-

2) Preguntas de respuesta breve:

- a) ¿Puede producirse una interrupción debido a un fallo de página?

Es obligatorio que se produzca. Cuando hay un fallo de página se tiene que interrumpir el proceso para ir a buscar la página que se necesita.

- b) Uso de sigaction: ¿qué es gestion.sa_handler=SIG_INT? ¿Qué hace la instrucción pause y sigpending?

El manejador va a tomar todas las señales que lleguen como un Ctrl-C de teclado (debería ser SIGINT y no SIG_INT). ← No tiene que ver con el valor del sa_handler, es paja y además inventada, que la función se llame SIG_INT no implica que tenga que atender a la señal SIGINT, eso se indica al usar 'sigaction([señal], &gestion, NULL);' por ejemplo, y definiendo en la función como funcione. Es decir, puede ser, pero no lo es a la fuerza.

En el caso de que "SIG_INT" sea lo que se pida, entonces puede estar haciendo referencia a una función de la cual desconocemos su contenido. `void SIG_INT()`

Pause: Fuerza a un proceso a detenerse hasta que reciba una señal (bien sea de kill o de un temporizador).

Sigpending: devuelve una máscara de señales pendientes de envío.

- c) ¿Qué método de gestión de E/S no utiliza IN y OUT? ¿Por qué?

Entrada y salida por asignación de memoria, ya que en vez de ejecutar código en ensamblador los registros de control están almacenados en variables en memoria, por lo que se pueden controlar mediante C o C++.

Al estar en el mismo espacio de direcciones (algunas son reservadas para los puertos) puede especificar la dirección directamente sin necesidad de usar métodos/funciones añadidos.

- d) Dada la instrucción `lw $t0, 0($s3)`, número máximo y mínimo de fallos de TLB y fallos de páginas, dada una memoria paginada en 2 niveles. (*Si no recuerdo mal, creo que decía que \$s3 leía una posición de memoria y almacenaba su valor en el registro \$t0*).

Mínimo de fallos es 0, puede que esté en la TLB de primeras.

Máximo de fallos: 3 fallos: 1 de TLB y 2 de página.

Busca \$s3 en TLB y no está (fallo TLB), va a TP1 y no está (fallo pagina), va a disco y carga en el marco 'x' de memoria principal y actualiza esa entrada de la TP1. Luego vuelve a la entrada de la TP1, marco 'x' y busca la entrada de la TP2 y no está (fallo pagina), va a disco y la carga en el marco 'y' de memoria principal y actualiza esa entrada de la TP2. **¿\$t0 no produce FP porque es un registro ?** ← También puede no estar la instrucción lw en memoria ¿no? Entonces serían otros 3 fallos, 1 de TLB y 2 de página a mayores, es decir el máximo sería 6 fallos.

Máximo de fallos: 6

- e) ¿Qué hace `pthread_yield()`? ¿En qué ejemplo sería útil?

Obliga a un hilo a renunciar a la CPU. Es útil en el trabajo conjunto de hilos creados por el mismo programador, cuando dada una condición, el hilo que usa la CPU ve que es mejor dejar a otro hilo usarla. Es útil cuando es predecible que un hilo vaya a bloquearse, de tal forma que cede la CPU voluntariamente a otro hilo.

- 3) **fallo de página**, si: dados un proceso A y otro B, el proceso A accede a (*ponía 6 direcciones hexadecimales*) y el proceso B accede a (*ponía otras 6 direcciones*) Memoria paginada de **2 niveles**. Direcciones virtuales de **40** bits. Direcciones físicas de **32** bits. Páginas de **16KB** (kibibytes). Unidad de direccionamiento **4 Bytes**. Entradas de **4 Bytes**. En la de 2º nivel cada tabla cabe exactamente en una página. Indicar la **secuencia de fallos hexadecimales***). Si se da un fallo de página, se conmuta el proceso por el otro, y se resuelve por orden de secuencia. 1082

Proceso A: 0x0F1001A0BB, 0x00047C3001, 0x0F1001A0BC, 0x02017C3FFF, 0x00047C30FF, 0x0F1001A0BD

Proceso B: 0x0C100130B0 , 0x00001C2000 , 0X0C100130B5 , 0x02017C3FF1 , 0x111101FFF00 , 0x0C100130B6

- Primero calculamos el formato de dirección:
40 bits = 1erNivel + 2ºNivel + OFFSET

2ºNivel:

$$\text{Tam.Tabla 2ºNivel} = \text{Tam.Página} = 16 * 2^{10} \text{ byte}$$

$$(\text{Tam.Tabla}/\text{Tam.Entrada}) = (16 * 2^{10} / 4) = 4 * 2^{10} = 2^{12} \gg \log_2 2^{12} = 12 \text{ bits}$$

OFFSET: se hace de la misma manera con (Tam.Página/U.Direccionable) >> 12bits

1erNivel: 1erNivel = 40 – (12+12) = 16bits

- Ahora los tamaños de las memorias:

Mem.Virtual = nºDir.Virtuales * tam.palabra = $2^{40} * 4 \text{ bytes} = 2^{42} \text{ bytes}$ de memoria virtual

nºPag.Virtuales = Mem.Virtual / Tam.Página = $2^{42} \text{ bytes} / 2^{14} \text{ bytes} = 2^{28} \text{ páginas}$ >> $\log_2 2^{28} = 28 \text{ bits}$

Esto se puede sacar directamente de 1erNivel (16bits) + 2ºNivel (12bits) = 28 bits

Mem.Física = $2^{32} * 4 \text{ bytes} = 2^{34} \text{ bytes}$ de memoria física

nºMarcos = Mem.Física / Tam.Página = $2^{34} / 2^{14} = 2^{20}$ marcos >> 20 bits

Para cada entrada utilizaremos 20 bits para referenciar a los marcos y 1 bit de A/P (ausente/presente).

- Secuencia (suponer que la pila está inicialmente vacía): **TP1** – **TP2** - **OFFSET**

A: 0x0F1001A0BB ---- 0000 1111 0001 0000 0000 0001 1010 0000 1011 1011

TP1 = 3856, TP2 = 26, OFFSET = 187

La entrada 3856 de la TP1 del proceso A tiene bit A/P = 0, por lo que hay un fallo de página. Vamos al disco a buscarla y la cargamos en el marco 0 de memoria principal y actualizamos la entrada 3856 de la TP1 del proceso A.

CAMBIO DE CONTEXTO

B: 0x0C100130B0 ---- 0000 1100 0001 0000 0000 0001 0011 0000 1011 0000

TP1 = 3088, TP2 = 19, OFFSET = 176

La entrada 3088 de la TP1 del proceso B tiene bit A/P = 0, por lo que hay un fallo de página. Vamos al disco a buscarla y la cargamos en el marco 1 de memoria principal y actualizamos la entrada 3088 de la TP1 del proceso B.

CAMBIO DE CONTEXTO

A: Ahora que hemos cargado la página en la entrada 3856 de la TP1 en el marco 0, accedemos a ella y buscamos la entrada 0x01A (26) en la TP2. Tiene bit de A/P = 0, por lo que hay un fallo de página. Vamos al disco a buscarla y la cargamos en el marco 2 de memoria principal y actualizamos la entrada 26 de la TP2 (entrada 3856 de la TP1) del proceso A.

CAMBIO DE CONTEXTO

B: exactamente lo mismo para el proceso A.

CAMBIO DE CONTEXTO

A: ya puede acceder a la página solicitada (que está en marco 2) a través de la entrada 3856 de TP1, entrada 26 de la TP2 con bit de A/P = 1. Dentro de la página accedemos a los datos desplazados 0x0BB = 187 direcciones respecto a la dirección base de la página.

Se completó la solicitud de A con 2 fallos de página.

Para hacerlo más corto vamos a ir a diferentes casuísticas del problema:

Cuando A solicita 0x0F1001A0BC va a la entrada 3856 de la TB1 con bit A/P = 1, accede a la entrada 26 de la TB2 con bit A/P = 1. Carga los datos desplazados 188 direcciones respecto a la dirección base de la página.

Se completó la solicitud de A con 0 fallos de página.

Más adelante vamos a ver que ambos procesos solicitan A: 0x02017C3FFF y B: 0x02017C3FF1

Los bits de las entradas tanto para la TP1 como para la TP2 son los mismos, pero al tratarse de direcciones virtuales son completamente independientes. Habrá una entrada TP1 = 0x0201 del proceso A y otra igual para B, al igual que la 0x7C3 para la TP2. Pero todas ellas cargan la información en marcos de memoria física diferentes por lo que es imposible que A y B accedan a la información del otro.

4) Diapositiva DMA. Puso el dibujo y mandaba cambiarlo a interrupciones de E/S y explicarlo. Información para resolverlo:

5)

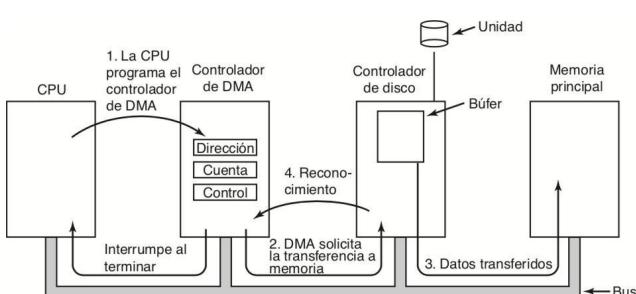


Figura 5-4. Operación de una transferencia de DMA.

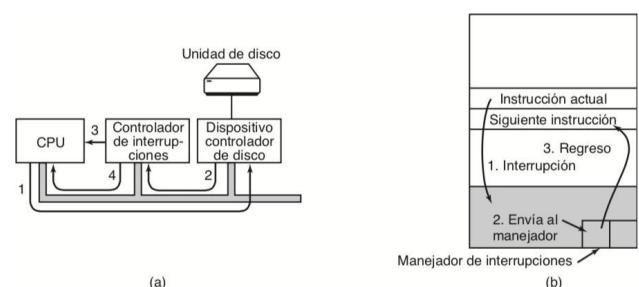


Figura 1-11. (a) Los pasos para iniciar un dispositivo de E/S y obtener una interrupción. (b) El procesamiento de interrupciones involucra tomar la interrupción, ejecutar el manejador de interrupciones y regresar al programa de usuario.

6) Dibujo de hilos en espacio de usuario y kernel. Había que completar las flechas.

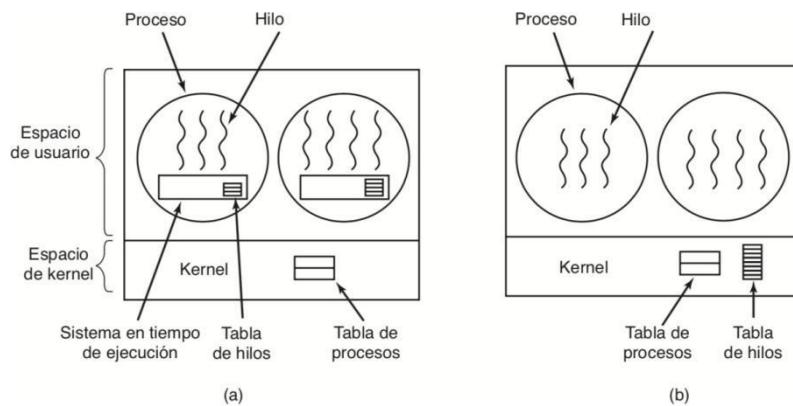
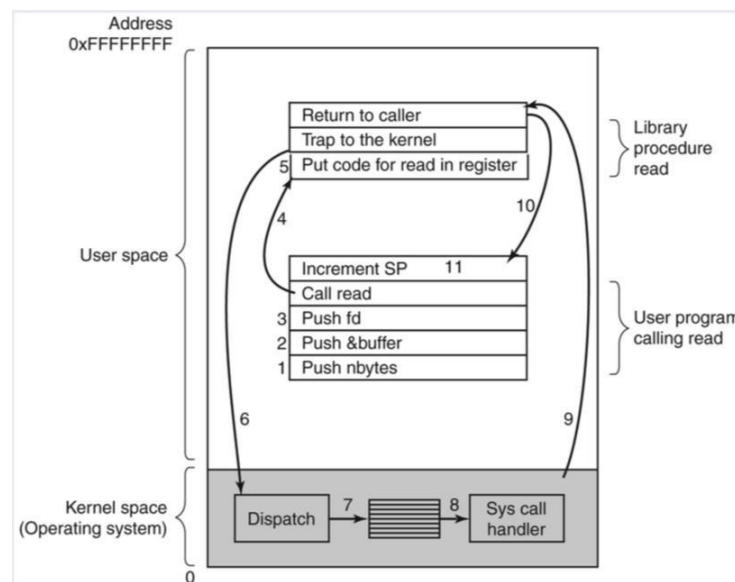


Figura 2-16. (a) Un paquete de hilos de nivel usuario. (b) Un paquete de hilos administrado por el kernel.

Sistemas Operativos I - Junio 2018

1) 2 puntos. Indicar qué hace... (Contestado en el de Enero)

- a) Paso 4.
- b) Paso 5.
- c) Paso 7.
- d) ¿Para qué sirve el manejador (dispatch)?''



2) 1 punto. Ejercicio sobre archivos (nodo-i, disco FAT...) No lo recuerdo muy bien.

Te ponía un nodo-i (una tabla con información desordenada) que apuntaban a otras tablas, tenías que averiguar el tamaño y la dirección física real de algunos datos.

3) 3 puntos. Memoria paginada en **2 niveles**. Direcciones virtuales de **40** bits. Direcciones físicas de **32** bits. Páginas de **16KB**. Unidad de direccionamiento **4 Bytes**. Entradas de **4 Bytes**. Indicar la **secuencia de fallos de página**, si: dados un proceso A y otro B, el proceso A accede a (*ponía 6 direcciones hexadecimales*) y el proceso B accede a (*ponía otras 6 direcciones hexadecimales*). Si se da un fallo de página, se conmuta el proceso por el otro, y se resuelve por orden de secuencia. (Contestado en el de Enero)

4) 4 puntos. Preguntas de respuesta breve:

a) ¿Puede producirse una interrupción debido a un fallo de página? (Contestado en el de Enero)

b) ¿Por qué a la TLB se le llama caché de direcciones?

Porque es un tipo de memoria caché que asocia direcciones virtuales a direcciones físicas.

c) ¿Qué método de gestión de E/S no utiliza IN y OUT? ¿Por qué? (Contestado en el de Enero)

d) ¿Qué es una interrupción de reloj?

Es una interrupción generada por el reloj de interrupciones del SO, con el objetivo de que ningún proceso acapare el uso de la CPU. (Se generan cada cierto tiempo)

- e) ¿Qué hace la instrucción *pause* y *sigpending*? (Contestado en el de Enero)
- f) Dada la instrucción *lw \$t0, 0(\$s3)*, número máximo y mínimo de fallos de TLB y fallos de páginas, dada una memoria paginada en 2 niveles. (*Si no recuerdo mal, creo que decía que *\$s3* leía una posición de memoria y almacenaba su valor en el registro *\$t0**).(Contestado en el de Enero)

- g) ¿Qué es el Working-Set?

El Working-Set (o conjunto de trabajo) de un proceso es el conjunto de páginas que usa en un momento dado.

- h) ¿Qué hace *execv()*?

Proporciona un vector de punteros a cadenas de caracteres terminadas en null, que representan la lista de argumentos disponible para el nuevo programa.

*La familia de funciones *exec* reemplaza la imagen del proceso en curso con una nueva*

2017

1. *1,5 puntos* Indica cuál será la salida del código siguiente. Indica cuál es la salida si se quita el comentario que aparece en ella.

```
int varglobal=10;
void hilo1(void *ptr)
{
    int a;
    a=*((int *) ptr);
    a=3;
    varglobal=5;
    printf("En el hilo a=%d %p\n",a, &a);
    printf("En el hilo varglobal=%d %p\n",varglobal, &varglobal);
    sleep(10);
}

main()
{
    pthread_t th1;
    int irl, a=1, b=2;
    irl = pthread_create(&th1,NULL,(void*)&hilo1, (void*) &a);
    // pthread_join(th1,NULL);]
    printf("varGlobal=%d en %p\n",varglobal,&varglobal);
    printf("a=%d en %p\n",a,&a);
}
```

Con el comentario puesto se ejecutará primero el main() por lo que quedaría algo como esto:

```
varGlobal=10 en 0x1071c5028
a=1 en 0x7ffee8a3ba50
En el hilo a=3 0x700002d6ef04
En el hilo varglobal=5 0x1071c5028
```

si termina el proceso principal, se cortan todos los hilos

<https://stackoverflow.com/questions/11875956/when-the-main-thread-exits-do-other-threads-also-exit>

Con el comentario quitado, el pthread_join() obliga a esperar a que termine el hilo, por lo que se ejecutará antes el código de *hilo1()*. Quedará así:

```
En el hilo a=3 0x700003299f04
En el hilo varglobal=5 0x103575030
varGlobal=5 en 0x103575030
a=1 en 0x7ffeedc68ba50
```

2. *1,5 puntos* Consideremos un sistema con memoria virtual paginada en el que la asignación de marcos es global a todos los procesos (todos compiten por el uso de la memoria principal). Existen 5 marcos de página y tenemos solamente dos procesos, A y B. El proceso A requiere acceder sucesivamente a las siguientes páginas virtuales 7, 15, 17, 13, 15, 8, 15, 11, 9 y 9. Mientras que B requiere sucesivamente acceder a las páginas virtuales 3, 7, 8, 6, 7, 5, 7, 5, 5 y 5. Se supone que la primera página que se carga es la primera requerida por el proceso A, después la primera del B, a continuación la segunda de A y así se alternan hasta el final. Si inicialmente la memoria principal está vacía, determina el número de fallos de página que se producen si se aplica un algoritmo de reemplazo LRU.

****Hay que tener en cuenta 2 cosas:** lo que se buscan son páginas virtuales, propias de cada proceso por lo que la página 7 del proceso A no es la misma que la del B. El algoritmo LRU es global en este caso pues nos dice que todos los procesos compiten por el uso de la memoria principal (**Si B tiene que remplazar una página de A, lo hará****)

A busca la 7, no está, fallo. La carga.
 B busca la 3, no está, fallo. La carga.
 A busca la 15, no está, fallo. La carga.
 B busca la 7, no está, fallo. La carga
 A busca la 17, no está, fallo. La carga.

7 _a
3 _b
15 _a
7 _b
17 _a

B busca la 8, no está, fallo.
 La carga en la posición menos usada recientemente(LRU), la 7_a

8 _b
3 _b
15 _a
7 _b
17 _a

A busca la 13, no está, fallo. ||
 La carga

8 _b
13 _a
15 _a
7 _b
17 _a

B busca la 6, no está, fallo. ||
 La carga

8 _b
13 _a
6 _b
7 _b
17 _a

A busca la 15, no está, fallo. ||
 La carga

8 _b
13 _a
6 _b
15 _a
17 _a

B busca la 7, fallo.
 A busca la 8, fallo.

|| B busca la 5, fallo. ||
 A busca la 15, acierto. ||

B busca la 7, acierto. ||
 A busca la 11, fallo. ||

8 _a
13 _a
6 _b
15 _a
7 _b

8 _a
5 _b
6 _b
15 _a
7 _b

8 _a
5 _b
11 _a
15 _a
7 _b

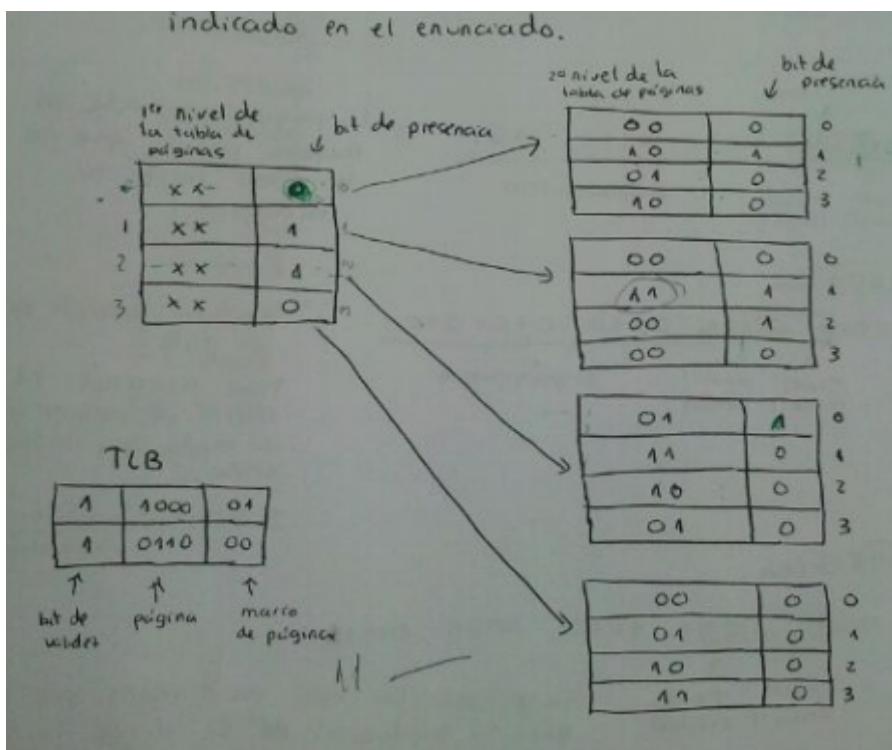
B busca la 5, acierto. ||
 A busca la 9, fallo. ||

B busca la 5, acierto. ||
 A busca la 9, acierto. ||

B busca la 5, acierto.

9 _a
5 _b
11 _a
15 _a
7 _b

Se producen 14 fallos de página.



3. 2 puntos Contesta razonadamente a las siguientes cuestiones:

- Sea un sistema con palabras de 64 bits. Sus direcciones virtuales son de 48 bits y sus direcciones físicas de 32 bits. Las páginas son de 8KB y la unidad direccionable es la media palabra (es decir, 32 bits). Haz una estimación razonada del tamaño de la tabla de páginas.
- Si para este sistema se quiere usar un sistema páginado en dos niveles de modo que las tablas del segundo nivel quepan en una página (supón ahora que cada entrada de dicha tabla ocupa 64 bits). ¿Cuántos bits se deben asignar para direccionar la tabla del primer nivel? Supón que el sistema tiene 10 procesos activos, ¿Cuántas tablas de primer y segundo nivel hay en total como máximo?

$$a) \text{ Tam.TP} = \text{nºPáginas} * (\text{bits para marcos} + \text{bit A/P})$$

$$\text{memoriaFísica} = 2^{32} * 4\text{bytes (32bits)} = 2^{34} \text{ bytes}$$

$$\text{nºMarcos} = \text{memoriaFisica} / \text{tamPagina} = 2^{34} / 8 * 2^{10} = 2^{21} \text{ marcos}$$

>> 21 bits para marcos

$$\text{memoriaVirtual} = 2^{48} * 4\text{bytes} = 2^{50} \text{ bytes}$$

$$\text{nºPáginas} = \text{memoriaVirtual} / \text{tamPagina} = 2^{50} / 8 * 2^{10} = 2^{37} \text{ páginas}$$

>> 37 bits para páginas (bitsTP1 + bitsTP2)

$$\text{Tam.TP} = 2^{37} * (21 + 1) = 3.02 * 10^{12} \text{ bytes} = 3.02 \text{ TB}$$

$$b) \text{bitsDirVirtuales} = \text{bitsTP1} + \text{bitsTP2} + \text{bitsOFFSET}$$

$$48 = \text{bitsTP1} + \text{bitsTP2} + \text{bitsOFFSET}$$

>> bitsOFFSET= 48 - 37(bits para páginas) = 11 bits

$$\text{entradasTP2} = \text{TamPagina} / \text{TamEntrada} = 2^{13}\text{bytes} / 64\text{bits(8bytes)} = 2^{10} \text{ entradas}$$

>> 10 bits TP2

$$\text{bitsTP1} = 48 - (10 + 11) = 27 \text{ bits}$$

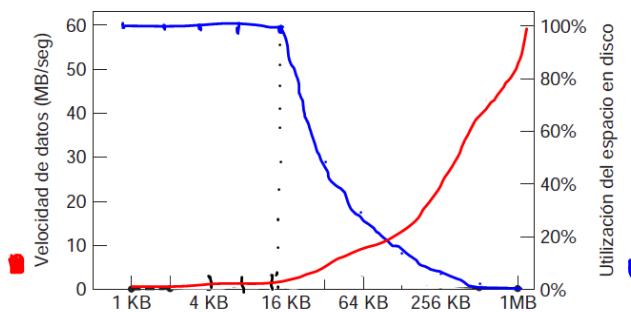
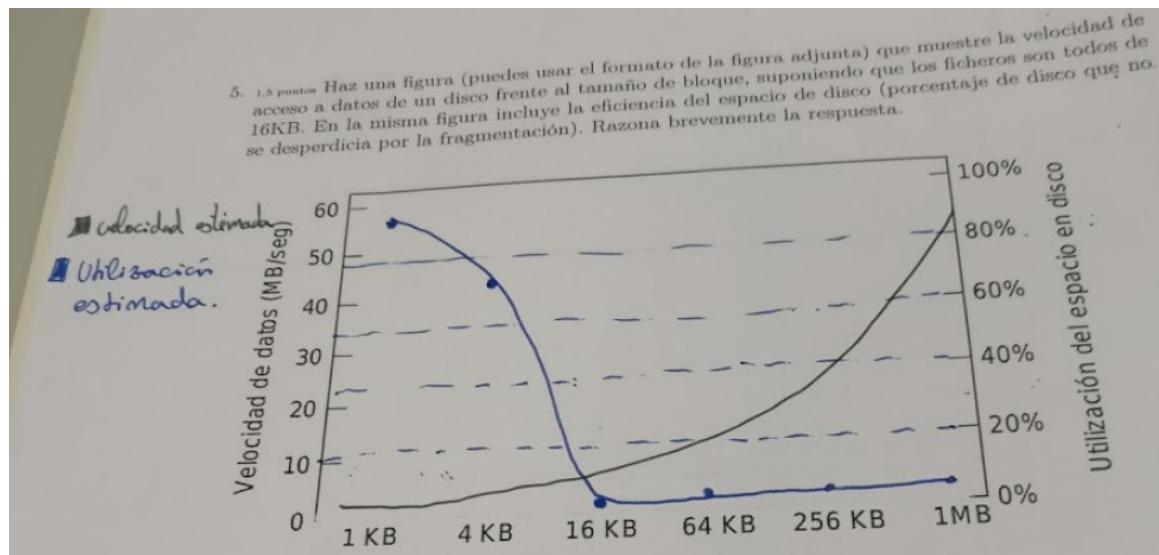
- o Tablas de 1er y 2º nivel con 10 procesos?

Para cada proceso hay 1 tabla de 1er nivel con 2^{27} entradas (una para cada TP2).

Por lo tanto en total habrá: **10 TP1 y $10 \cdot 2^{27}$ TP2**

4. 1 punto ¿Dónde están las páginas que en el proceso de traducción producen fallo de TLB?

Si se produce un fallo de TLB puede que las páginas que se necesitan estén en las tablas de páginas de memoria principal. Si no están ahí puede que estén en el disco. Si no están en el disco entonces es que no existen tales páginas.



Básicamente la **velocidad de datos** se calcula suponiendo que para traer un bloque de datos de tamaño 'x' vas a tardar los mismos milisegundos para cualquier valor de 'x'. Por lo que compensa traer la mayor cantidad posible. Aún así hay que tener en cuenta que esto tiene un límite.

La **utilización del espacio en disco** se explica así: un archivo usa una cantidad de datos de los bloques que llegan, si esos bloques son iguales o menores que el tamaño del archivo entonces se usará el 100% de los datos. Si el bloque trae el doble de datos que el tamaño del archivo (16KB) entonces solo utilizará el 50%, si es 4 veces más grande se usará el 25% y así sucesivamente.

6. 2 puntos Los códigos adjuntos corresponden a la gestión de la Entrada/Salida controlada por interrupciones. Explica su funcionamiento.

```
copiar_del_usuario(bufer, p, cuenta);
habilitar_interrupciones();
while (*reg_estado_impresora != READY);
*registro_datos_impresora = p[0];
planificador();
```

(a)

```
if (cuenta==0) {
    desbloquear_usuario();
} else {
    *registro_datos_impresora = p[i];
    cuenta=cuenta - 1;
    i = i + 1;
}
reconocer_interrupcion();
regresar_de_interrupcion();
```

(b)

El a) pertenece al driver de una impresora y el b) al resolutor de interrupciones.

En una impresora, el tiempo de escritura de cada carácter es tiempo que la cpu está inactiva, por lo que se debe utilizar para otra cosa. Por esto existe la gestión de E/S controlada por interrupciones.

a)

Se lanza la llamada al sistema para imprimir la cadena. Entonces se copia el búfer en el espacio de Kernel (p).

Cuando la impresora esté disponible se copia en ella el primer carácter.

Entonces se llama al planificador, que ejecuta algún otro proceso.

El proceso que pidió imprimir se bloquea hasta que se haya impreso toda la cadena.

Cuando la impresora ha escrito un carácter y está preparada para aceptar el siguiente genera una interrupción que detiene el proceso actual y guarda su estado.

b)

Se ejecuta el resolutor de interrupciones de la impresora:

Este comprueba si el carácter a imprimir es el último, si lo es se desbloquea el usuario.

Si no acabó imprime el siguiente carácter, reconoce la interrupción y regresa al proceso que se estaba ejecutando.

Examen Sistemas Operativos Enero 2014

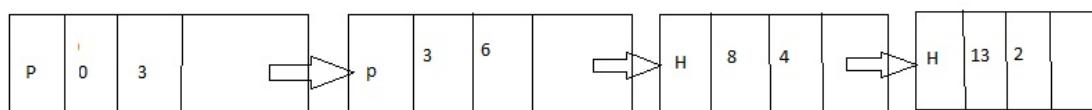
1-Explica brevemente si las siguientes afirmaciones son verdaderas o falsas (2 puntos):

a) La técnica de intercambio de memoria utiliza reubicación dinámica.

b)

Verdadero, para intercambiar bloques de memoria tienes que saber dónde empiezan y dónde acaban. Cuando haces un intercambio y buscas un hueco, los datos que ofrece la reubicación dinámica son cruciales.

c) La lista enlazada para la administración de memoria que se muestra en la figura tiene un error en la posición de los huecos:



Efectivamente, el segundo proceso que tiene valor 6 de longitud y empieza en la 3. Por lo que el siguiente hueco o proceso debería empezar en la 9, no en la 8

d) Siempre que se produce un fallo de TLB se produce un fallo de pagina.

Falso, cuando la información que buscamos no está en la TLB, se busca en la tabla de páginas y puede que se encuentre ahí y que no produzca un fallo de página.

//pero un fallo de página siempre produce interrupción

e) El algoritmo de reemplazo de paginas NFU tiene en cuenta el tiempo que llevan las paginas sin utilizarse para decidir que pagina hay que reemplazar

NFU: no usado frecuentemente

Mentira, no lo tiene en cuenta. Simplemente aumenta el valor del contador de una página cada vez que se usa y sustituye la que menos contador tiene, pudiendo ser esta una página que se acaba de usar (pero que tiene un contador menor que el resto).

f) En un sistema de archivos tipo UNIX los nodos-i de los archivos no siempre están en memoria principal.

Falso, sólo se almacenan en memoria principal cuando el archivo está abierto. (No sería verdadero? ya que solo estan en memoria principal cuando el archivo esta abierto)

g) En la cache de bloques, el área de intercambio (o área swap) y la tabla de páginas están siempre en memoria principal.

El área de intercambio se encuentra en el disco. La tabla de páginas no están en memoria principal porque hacen referencia a más direcciones de memoria que las que hay en memoria principal.

h) Una de las diferencias entre un sistema de archivos UNIX y otro de Windows esta en donde se almacenan los atributos de los archivos.

i) La planificación apropiativa es la más recomendable en un sistema operativo por lotes.

No, normalmente se emplea la planificación no apropiativa o, en algún caso excepcional, apropiativa con un quantum muy largo.

2- En un sistema de memoria virtual con direcciones virtuales de 20 bits, páginas de 32kb, una memoria física de 256kb ($2^8 * 2^{10} b$) y una tabla de páginas invertida con los contenidos que se especifican (2 puntos):

Bit primario

7	10001	0
6	00001	1
5	01011	1

4	10001	1
3	10010	0
2	01111	1
1	10000	0
0	00100	1

- a) ¿Cuántas páginas del espacio de direcciones virtuales hay en memoria principal, y a que marcos de página están asociadas?

5. Asociadas a los marcos 0, 2, 4, 5 y 6

- b) Indica el tamaño del espacio de direcciones virtuales, el número de bits de las direcciones de memoria físicas y tamaño de marcos de página. Dibuja la dirección virtual y la dirección física especificando los campos en que se dividen y el número de bits de cada campo.

2^{18} = direcciones de MV

2^{18} = tam.mem.fisica = 18 bits de direcciones (máximo, nos falta la Unidad Direccional)

$2^{18} / 2^{15} = 2^3$ marcos de página (8)

*En el c) sería F8F0F en vez de FF8FF0FF y 0A061, en vez de 07061

c) Traduce las direcciones virtuales c v FF8FF0FF_{heeee}, 07061_{heeee}, 90940_{heeee} y 88888_{heeee}. Indica que fallos se producen y explica el motivo.

d) Si se añade un TLB de dos entradas que en un momento dado tiene el contenido (página virtual, modificado, marco de página) = (17,0,3),(1,0,1), indica cómo se hace la traducción de la dirección 8AA00_{heeee}. *Es 8A000

3- Al compilar el código C del programa de nombre pre3 que se muestra en la figura 2 se crea el ejecutable pre3.out desde la línea de comandos se le asocia el PID=260 y que la asignación de los PID de los procesos hijos, si se llegan a crear, se realiza incrementando en una unidad el PID del padre para el primer hijo, diez unidades para el segundo hijo, en veinte para el tercer hijo etc. Contesta razonadamente los siguientes apartados (2 puntos):

- a) Explica las sentencias numeradas.
- b) Explica detalladamente el funcionamiento de este programa si se invoca desde el interprete de comandos con la orden >pre3.out

Figura 2

```
#include <signal.h>
main (int argc, char *argv[]){
    int v[2],N;
    v[0]=0;v[1]=0;
    if (argc!=2)
        exit (4);
    else {
        N=atoi(argv[1]);
        signal(SIGUSR1,SIG_IGN);
        while(fork()==0&& N>0){
            N=N-1;
            v[0]=getpid();
            v[1]=getppid();
            sleep(1);
        }
        printf("%d,%d\n",v[0],v[1]);
    }
}
```

4- Considera un disco magnético con 20 cabezas y 400 cilindros. El disco está dividido en 4 zonas de 100 cilindros, donde los cilindros en distintas zonas contienen 100, 200,300 y 400 sectores, respectivamente. Supón que cada sector tiene 512 bytes, que el tiempo de búsqueda promedio entre cilindros adyacentes es de 1ms y que el disco gira a 7200rpm. Calcula (2 puntos):

- a) Capacidad del disco.
- b) El desajuste óptimo de pistas en cada una de las cuatro zonas.
- c) La velocidad máxima de transferencia de datos en bytes/s.

5- Un cierto sistema operativo cuando comienza o continúa con la ejecución de un proceso realiza la pre paginación de su conjunto de trabajo en los marcos asignados al proceso. En un cierto instante de tiempo t1 un proceso A, cuyo conjunto de trabajo en dicho instante esta formado por los marcos de pagina 0, 1 y 3 se bloquea y pasa a ejecutar otro proceso B. En t2 el proceso A vuelve a ser planificado para su ejecución y realiza la siguiente secuencia de referencias a páginas : 3,1,2,1,6,7,9,5,2,3,2,9,3,3,6 y 2. Responder(2 puntos):

- a) Determina el número de fallos de página producidos por la ejecución del proceso A partir del instante t2, suponiendo que el sistema operativo asigna 3 marcos a la ejecución del proceso A y utiliza el algoritmo FIFO con una cola de tres entradas. Supón que inicialmente el contenido

de la cola FIFO, se haya con los números de pagina 0,1 y 3, que la primera posición de la cola está ocupada por la pagina 0 para entrar y la ultima por 3.

3 --> acierto; 1 --> acierto; 2 --> fallo (1,3,2); 1 --> acierto; 6 --> fallo (3,2,6) ;... Así todo el rato.

b) Indica el contenido final de la cola FIFO.

c) La traducción de direcciones se realiza usando una MMU con un TLB. Suponiendo un tiempo media de acceso al TLB despreciable (igual a 0), una tasa de acierto de TLB del 95%, tasa de fallos de página del 10%, tiempo medio de acceso a memoria de 200ms y un tiempo medio de gestión de un fallo de página de 50ms, determina el tiempo medio necesario para atender una referencia a memoria. No se tiene memoria caché.

2013

1.- preguntas de entrada y salida

a) si un proceso ejecuta una instruccion de e/s, a que estado pasa?

b) si un proceso termina de utilizar e/s y como consecuencia hay interrupcion a que estado pasa?

c) eventos que produzcan que un x pase de estado ready a running

2.-para que sirven los quantums en el algoritmo de round-robin del planificador de procesos

3.-un proceso con muchos hilos ejecutandose:

a) señala cuales de estas cosas son globales o particulares a cada hilo: registros, pc, pila, archivos abiertos y procesos hijos.

b) otra opcion que no me acuerdo

4.- codigo de programa: describir los que hacen unas sentencias entre corchetes, si se le pasan a la consola este comando como se ejecutaria el proceso: prog1 5

prog 1 es el nombre del programa. codigo

mas o menos:

```
int main(int argc, char * argv[]){ int
```

```
    z,N,u[2];
```

```
    N=atoi(argv[1]; if(argc!=2) exit(1);
```

```
    while(fork()==0 && N>0){
```

```
        u[0]=getpid();
```

```
        u[1]=getppid(); sleep(1);
```

```
}
```

```
pause (1);
```

```
}
```

algo asi xD

5.- traducir memoria y todo eso:

páginas de 8KB, espacio de direcciones virtuales 64KB, memoria física de 32KB.

- a) señalar tamaños bits y todo eso que se suele hacer en un ejercicio de estos:
- b) traducir la dirección creo k era 61ADhex
- c) decir una dirección virtual que fallaría en su traducción explicando porque
- d) te dan un tlb de dos entradas: (2,0,1) y (5,0,2) teniendo en cuenta que el formato es (pagina, modificado, marcodepagina) traduce la dirección virtual 4001hex

tabla de asociación:

0 00 0

1 00 0

2 01 1

3 01 1

4 10 0

5 11 1

6 10 0

7 10 1

mas o menos, la importante es la 2 y esta bien jaja

6.- sistema de archivos dos preguntas k ni las lei casi pero la segunda era que info tenía el superbloque.

7.- preguntas sobre proyectos:

- a) para qué introduciríamos hilos en el trabajo del método quicksort
- b) k señal mandaríamos para que un padre matara a su hijo

2012

1) Preguntas breves

a) Diferencia entre proceso y programa

Un programa es un conjunto de instrucciones y estructuras de datos necesarias para la ejecución y un proceso es un programa en ejecución.

b) ¿Que es una interrupción de reloj?

Es una interrupción que manda el reloj del SO con la intención de medir el tiempo de alguna manera y evitar que un proceso acapare la CPU.

c) ¿Que es la tabla de procesos y la tabla de regiones por proceso?

La tabla de procesos es una tabla del kernel donde se recogen todos los procesos para facilitar los cambios de contexto y demás acciones y se encuentra en la ram

d) ¿Cuantas tablas de regiones y tablas de proceso hay por cada proceso?

2) Por qué se utiliza la paginación?

3) Para que se utiliza el bit R en WS?

4) Ejercicio de direccionamiento de memoria virtual

5) Ejercicio de protección de archivos

6) Preguntas sobre nodos-i

7) Que son las listas de control de acceso?

8) Preguntas sobre los proyectos

El ejercicio sobre direccionamiento de memoria virtual es como los ejemplos que hay en el libro, como se direcciona la memoria virtual, una en dos etapas, en donde se parte con unas tablas con los índices.

EXAMEN SISTEMAS OPERATIVOS I (XANEIRO 2011)

1) Responde brevemente a las siguientes preguntas:

- a) ¿Cuál es la diferencia entre un proceso y un programa?
- b) ¿Qué es una interrupción de reloj?
- c) ¿Qué información se guarda en la tabla de procesos? ¿Y en la tabla de regiones por proceso?
- d) ¿Cuántas tablas de procesos y tablas de regiones por proceso mantienen el núcleo del Sistema Operativo?

2) Explica brevemente porqué es necesario dividir en páginas el espacio de direcciones de un proceso en un sistema de memoria virtual.

3) En el algoritmo de reemplazo buscado en working –set, ¿para qué se utiliza el bit R que está asociado a cada página en la tabla de páginas?

4) En un sistema de memoria virtual con un espacio de direcciones virtual con 16 páginas de 8KB, una memoria física de 4 páginas, una tabla de páginas en dos niveles y una TLB de dos entradas, con los contenidos que se especifican en la figura, suponiendo que la tabla de páginas no ocupa espacio en el espacio de direcciones físicas. (Figura del final).

- a) Indica el tamaño del espacio de direcciones virtual y físico, el número de bits de direcciones virtuales y físicas y el tamaño de los marcos de página.
- b) Dibuja la dirección virtual, especificando los campos en que se divide y el número de bits de campo.
- c) Traduce las direcciones virtuales $0B55_{hex}$, 11654_{hex} y $09E17_{hex}$. Indica que fallos se producen y explica el motivo. Si el número de bits de la dirección virtual **b** es menor que 20, **b<20**, la dirección se especifica con los **b** bits menos significativos del número hexadecimal indicado en el enunciado.

5) Dado el código mostrado al final, considerando que en un cierto directorio el programa ejecutable pertenece a USUARIO 1 y tiene permisos **rws rwx rwx** (tiene el bit activado **S_ISUID**), el fichero **fichero1.txt** pertenece también a USUARIO 1 con permisos **rw**, el fichero **fichero2.txt** pertenece a USUARIO 2 con permisos **rw-- -- --** y que USUARIO 1 tiene **uid=501**, y USUARIO 2 TIENE **uid=503**, indica cuál será la salida por pantalla si USUARIO 2 ejecuta el programa.

6) En un sistema de archivos basado en nodos-i.

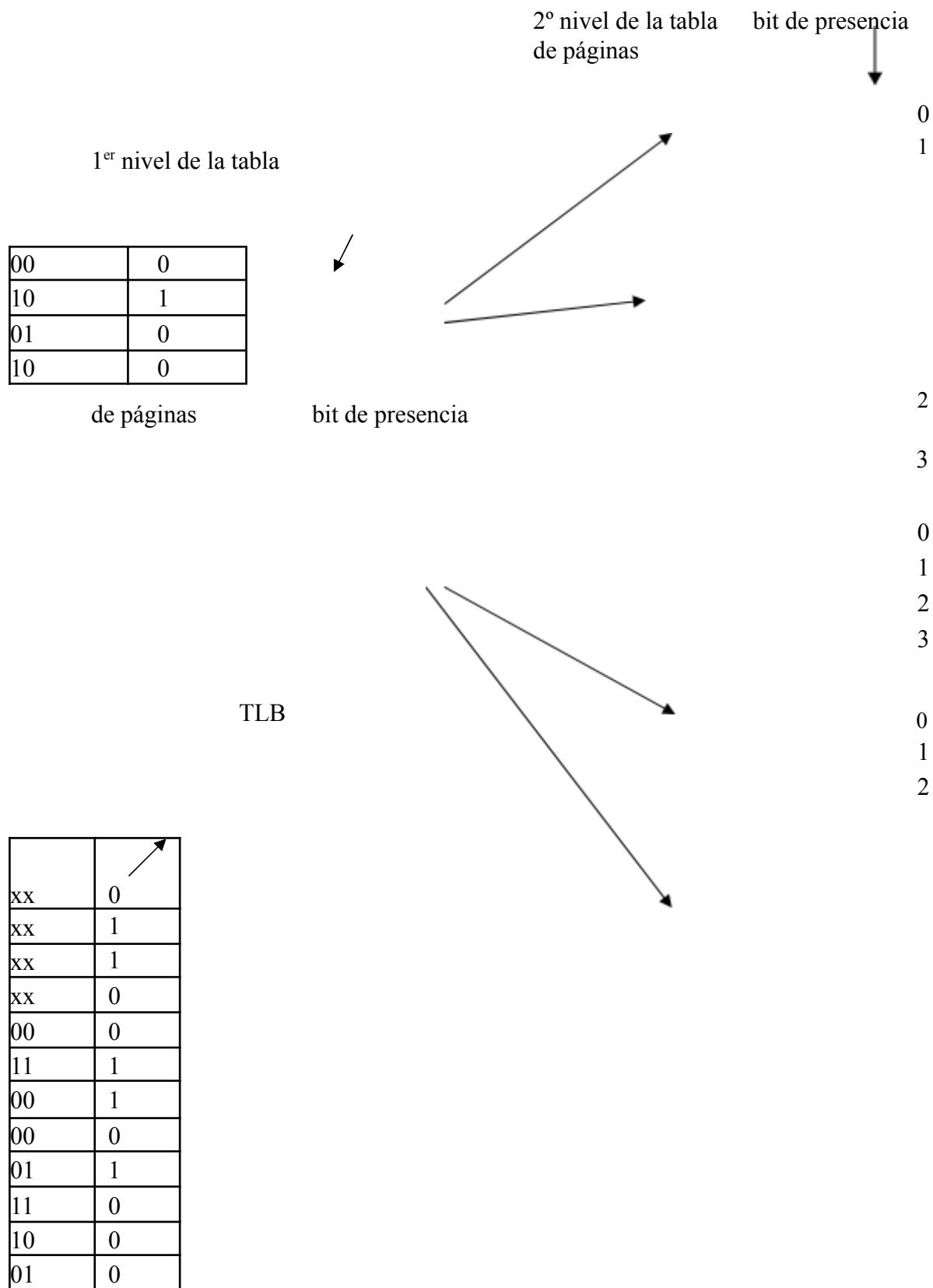
- a) ¿Qué información contiene el nodo-i de un archivo?
- b) ¿Dónde se almacena el nodo-i de un archivo? ¿Cuándo está abierto?
- c) ¿Qué es un nodo-i?

7) Explica brevemente que son las listas de control de acceso e un entorno de seguridad.

8) Preguntas proyectos expuestos en clase:

- Dado un proceso con varios hilos ejecutándose concurrentemente, ¿qué ocurre cuando se ejecuta la llamada al sistema **exit** en uno de los hilos?
- ¿En qué directorio se encuentra la información sobre el tipo de modelo de CPU de un computador?

Figura pregunta 4



1	1000	01
1	0110	00

bit de validez

3
0
1

Código pregunta 5

2

00	0
01	0
10	0
11	0

#include <fcntl.h>

3

```
main() {
    int x, y, fd1, fd2;
    x = getuid();
    y = geteuid();
    printf("\nUID = %d, EUID = %d\n", x,
y); fd1 = open ("fichero1.txt",
O_RDONLY); fd2 = open ("fichero2.txt",
O_RDONLY);y
    printf("fd1 = %d, fd2 = %d\n", fd1, fd2); setuid(x);
    printf("x = %d, UID = %d, EUID = %d\n", x, getuid(), geteuid());
    fd1 = open ("fichero1.txt", O_RDONLY);
    fd2 = open ("fichero2.txt", O_RDONLY);
    printf("fd1 = %d, fd2 = %d\n", fd1, fd2); setuid(y);
    printf("y = %d, UID = %d, EUID = %d\n", y, getuid(), geteuid());
}
```

EXAMEN SISTEMAS OPERATIVOS I (FEBREIRO 2010)

- 1) ¿Cuál es la diferencia entre un proceso y un programa? ¿Y entre un hilo y un proceso?
- 2) Indique que eventos pueden hacer que un proceso que un proceso en estado *listo* pase a estado en *ejecución* y de estado en *ejecución* a estado *listo*.
- 3) Responde brevemente a los siguientes preguntas:
 - a) ¿Qué es una interrupción de reloj?
 - b) ¿Qué información se guarda en la tabla de procesos?
 - c) ¿Qué información se guarda en la tabla de regiones de procesos?
 - d) ¿Cuándo se utiliza la pila del núcleo?
 - e) ¿Cuántas tablas de procesos y tablas de regiones por procesos mantiene el núcleo del sistema operativo?
- 4) Explique brevemente que información podemos extraer de la lista enlazada para administración de memoria que se muestra en la siguiente figura:

< < <

¿Dónde se le asignaría memoria a un proceso de tamaño 1 al utilizar algoritmos de asignación de primer ajuste, de mejor ajuste y de peor ajuste?

- 5) Explique brevemente porqué es necesario dividir en páginas el espacio de direcciones de un proceso en un sistema de memoria virtual.
- 6) En un sistema de memoria virtual con un espacio de direcciones virtual de 64 KB, páginas de 8 KB, una memoria física de 32 KB y una tabla de páginas con el siguiente contenido: (00,0), (00,0), (01,1), (10,1), (10,0), (11,1), (00,1), (10,0).
 - a) Indicar el número de bits de las direcciones virtuales y físicas, el número de páginas y marcos de página, el tamaño de las páginas y marcos de página.
 - b) Traducir la dirección virtual 61AD_{hex}
 - c) Indicar una dirección virtual que produciría un fallo de página y explicar el motivo
 - d) Si añadimos un TLB de 2 entradas al sistema de la pregunta 7 y que en un momento dado tiene le siguiente contenido (página virtual, modificación, marco de página)=(2,0,1), (5,0,3). Indicar como se haría la traducción de la dirección virtual 4001_{hex}
- 7) En un sistema de E/S con DMA
 - a) ¿Es necesario utilizar interrupciones?
 - b) ¿Qué hardware adicional se necesita?
 - c) Indicar las ventajas de la E/S programada y la E/S controlada por interrupciones
- 8) Explicar que es un superbloque del sistema de archivos y el nodo-i de un archivo.

Puntuación:

<i>Preguntas 1,2,4,5,8</i>	<i>① 1 punto</i>
<i>Preguntas 3, 7</i>	<i>① 1,5 puntos</i>
<i>Pregunta 6</i>	<i>② 2 punt</i>

ANEXOS EXERCICIOS

Ejercicio 7

Ejercicio 7d

OUTRAS IMAXES QUE PODEN CAER NOS EXAMES

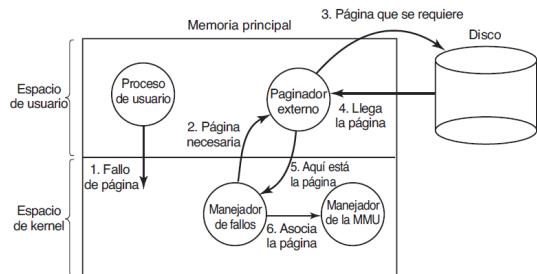


Figura 3-30. Manejo de fallos de página con un paginador externo.

