



SISTEMAS OPERATIVOS. PED 2.

CURSO 2011-2012

Juan José de Isusi Moure.

DNI: 30691733G

jjisusi@gmail.com

CENTRO ASOCIADO: Bizkaia.

PED2.CUESTIÓN 1.

i) La sobrepaginación aumenta el porcentaje de uso del procesador.

La afirmación es falsa. La sobrepaginación ocurre cuando a un determinado proceso no se le asignan suficientes marcos de página para poder tener cargado en memoria su conjunto de trabajo. Esto provoca que se produzcan constantemente fallos de página. Los procesos se bloquean a la espera de lecturas y escrituras desde/a memoria secundaria, lo que produce una reducción de la utilización del procesador.

ii) Se denomina buffering de páginas a la estrategia consistente en cargar un cierto número de páginas de un proceso antes de iniciar o continuar su ejecución.

La afirmación es falsa. El buffering de páginas consiste en que, cuando se selecciona una página como reemplazable, el contenido del marco de memoria en el que se aloja no se borra directamente. De este modo, en el caso de que se vuelva a referenciar la página en un futuro próximo, y en ese tiempo el marco no haya sido finalmente reemplazado, puede reutilizarse la información no borrada. De esa manera se produce un ahorro en la lectura en memoria secundaria.

La técnica por la cual se carga un cierto número de páginas de un proceso antes de iniciar o continuar su ejecución, porque está previsto que vayan a utilizarse en un futuro próximo y evitar en dicho momento el fallo de página, recibe el nombre de **prepaginado o paginación por adelantado**.

PED2. CUESTION 2.

a) Determinar el número de fallos.

El algoritmo de reemplazamiento LRU, consiste en la substitución de la página que se ha utilizado menos recientemente en caso de fallo de página.

Para implementar el método, una de las formas es mantener una pila con las páginas que se van utilizando, de modo que se coloque en la cima la última página utilizada.

En el caso de que se haga referencia a una página que se encuentra en uno de los marcos, esto quiere decir que se encuentra cargada en memoria principal, de modo que no es necesario el reemplazamiento. Sin embargo, para garantizar que en pila se mantengan las páginas ordenadas por orden de reciente uso, se extrae la página en cuestión y se coloca de nuevo en la cima de pila (aunque esta no sea la forma más eficiente ni habitual de acceso a pilas y que va a producir la sobrecarga del sistema).

Por el contrario, cuando la nueva página solicitada no se encuentra en ninguno de los marcos, nos encontramos con un fallo de página. En dicho caso eliminamos el elemento que se encuentra en la base de la pila y lo seleccionamos como víctima del reemplazamiento.

A continuación se presenta el proceso para el caso de 4 y 5 marcos.

4 marcos.

Las cuatro primeras referencias producen fallo dado que no tenemos ningún marco cargado y ninguna de las referencias está repetida. La quinta referencia solicita acceso a página 1, que se encuentra localizada en el marco 1, por lo que no hay fallo. Para refrescar la posición de la página 1 como la más recientemente usada se elimina de la pila y se coloca en su cima.

Sucesivamente se sigue este procedimiento, obteniéndose el siguiente esquema. En la parte superior se representa en estado de la pila cada vez que se referencia a una página, indicando a su vez cuándo ocurre un fallo. Al mismo tiempo en la fila de abajo se representa el contenido de los marcos.

1	3	2	4	1	5	7	4	3	2	8	9	4	5	4	9	1	8	3	2
1	3	2	4	1	5	7	4	3	2	8	9	4	5	4	9	1	8	3	2
1	3	2	4	1	5	7	4	3	2	8	9	4	5	4	9	1	8	3	2
F	F	F	F	A	F	F	A	F	F	F	F	F	F	A	A	F	F	F	F
1	3	2	4	1	5	7	4	3	2	8	9	4	5	4	9	1	8	3	2
1	3	2	4	1	5	7	4	3	2	8	9	4	5	4	9	1	8	3	2

Como podemos observar se producen 16 fallos.

5 marcos.

1	3	2	4	1	5	7	4	3	2	8	9	4	5	4	9	1	8	3	2
1				4	1	5	7	4	3	2	8	9	4	5	4	9	1	8	3
		3	2	2	4	1	5	7	4	3	2	8	9	4	5	4	9	1	8
1	1	1	1	3	3	2	2	1	5	7	4	3	2	2	2	8	5	4	9
F	F	F	F	A	F	F	A	F	F	F	F	A	F	A	A	F	A	F	F

1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	1	1	1	1
	3	3	3	3	3	7	7	7	7	7	9	9	9	9	9	9	9	9	9
		2	2	2	2	2	2	3	3	3	3	3	5	5	5	5	5	3	3
			4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2
				5	5	5	5	5	8	8	8	8	8	8	8	8	8	8	8

En este segundo caso se observa que se producen 14 fallos (dos fallos menos que en el caso anterior).

b) Explicar si mejoraría la tasa de fallos si se aumenta el número de marcos.

A priori, dado que el algoritmo LRU no adolece de la anomalía de Belady se podría esperar que el número de fallos se redujera si se aumenta el número de marcos o al menos lo que sí podemos asegurar que al menos la tasa de fallos se mantendrá igual o menor.

Observando el caso concreto de este ejercicio, dado que el acceso es a un número reducido de páginas (9), veríamos que con 9 marcos obtendríamos 9 fallos (el primer acceso a cada una de las nueve páginas) y posteriormente todos serían fallos. A partir de dicho número de marcos no tendríamos mejora alguna.

PED2.CUESTIÓN 3.

Funciones de las capas de software de E/S.

2.1. Subsistema de E/S.

El subsistema de E/S se encarga de las tareas necesarias para realización de operaciones E/S, que son comunes a todos los tipos de dispositivos. Las funciones principales que realiza son:

- Proporcionar una **interfaz uniforme para todos los controladores**, a la vista de los procesos que quieren utilizar sus dispositivos.
- Invocación de los drivers de un dispositivo determinado.
- Ocultación de diferentes tamaños de bloque de los dispositivos, proporcionando un tamaño de bloque uniforme a los niveles superiores de software.
- Planificación de la E/S con el objetivo de reducir el tiempo promedio de finalización de las operaciones, dado que los procesos solicitan llamadas cuando necesitan acceder a dispositivo pero no suelen hacerlo en el orden más adecuado. Además garantiza el acceso equitativo de las aplicaciones o priorizar algunas determinadas (más sensibles a un posible retardo).
- Asignación y liberación de dispositivos dedicados, que sólo pueden ser accedidos por un único proceso al tiempo.
- Bloqueo de procesos que solicitan operaciones E/S.
- Buffering, o almacenamiento temporal de datos, que permita adaptar velocidades entre procesos productores y consumidores, o entre dispositivos con diferentes tamaños de transferencia.
- Gestión de errores en operaciones E/S, tomando acciones determinadas, en función de si se trata de errores de dispositivo o errores de programación.

2.2. Drivers.

Un driver contiene código específico para el control de un dispositivo de E/S concreto o de una clase de dispositivos estrechamente relacionados. Para ello interactúa con el subsistema E/S y los controladores de E/S, realizando las siguientes funciones:

- Aislar al núcleo de los controladores de dispositivos.
- Recibir peticiones abstractas del Subsistema de E/S para la lectura y/o escritura en el dispositivo, comprobando los parámetros que pasa el subsistema de E/S son correctos y la operación solicitada se puede realizar y avisar en caso de que esto no sea posible.

- Traducción de los parámetros de la función de llamada en parámetros específicos del dispositivo.
- Comprobar la disponibilidad del dispositivo, activación, si alguna otra petición de E/S se encuentra atendiendo y gestionar las colas de las nuevas peticiones que llegan.
- Generar órdenes para el controlador del dispositivo y cargarlas en el controlador del dispositivo y esperar a que se ejecuten, bloqueándose a la espera de una interrupción de E/S. Comprobando que no se produzcan errores en operaciones E/S, resolviendo aquellas que estén en su mano o alertando al S.O. para que tome las medidas necesarias.

2.3. Manejadores de las interrupciones.

Es la capa de E/S de núcleo más dependiente del hardware. Sus principales funciones son:

- Ocultación al resto de capas del S.O. de las interrupciones, de modo que éste desconozca lo máximo de ellas.
- Desbloqueo de los drivers bloqueados a la espera de interrupción por parte del dispositivo.
- Transferencia de datos desde un registro del controlador del dispositivo a un buffer en el espacio del núcleo o viceversa, en caso de no emplear DMA.

PED 2. CUESTION 4.

a) Determinar tamaño de los campos de una dirección lógica.

Puesto que indican que la dirección lógica tiene el mismo tamaño que la dirección física, calculamos primero esta última.

Para ello partimos del tamaño de la memoria principal, que son 64 Kibipalabras ($64 \cdot 2^{10} = 2^6 \cdot 2^{10}$). Esto expresado en potencia de 2 son 2^{16} palabras, que pueden ser direccionadas mediante 16 bit (**n=16**).

Por otra parte la dirección lógica consta de dos campos, un primer campo que indica el número de segmento a que se hace referencia y un segundo campo que indica el desplazamiento.

Dado que el proceso consta de 5 procesos necesitaremos de 3 bits ($s = \lceil \log_2 5 \rceil = 3$) para su codificación (quedando 3 códigos libres sin uso).

El tamaño del campo de desplazamiento vendrá dado por la diferencia **d= n-s= 13**. Mediante este campo se pueden direccionar un máximo de 8192 palabras, lo que es coherente con los datos de longitud de los segmentos (el de mayor tamaño es de 8191).

Segmento	Desplazamiento
s=3	d=13

b) Determinar direcciones físicas correspondientes a las siguientes direcciones lógicas.

b.i) 11AEh

En primer lugar expresamos la dirección en binario:

0001000110101110b

Como se ha mencionado en el apartado anterior los tres bits más significativos nos indican el número de segmento (000, correspondiente al segmento 0).

Los trece bits restantes indican el desplazamiento, (1AEh, que se corresponde con 4526d). Puesto que el desplazamiento obtenido es menor a la longitud máxima del segmento cero, la dirección es válida.

La obtención de la dirección física se obtendrá simplemente sumando a la dirección base del segmento cero el desplazamiento obtenido.

Dirección física = dirección Lógica segmento 0 + desplazamiento = $0 + 4526 = 4526d$.

Expresado en hexadecimal:

11AEh.

Lógicamente coincide con la dirección lógica, al encontrarse el segmento 0 alojado en memoria principal a partir de la dirección 0.

b.ii) 6190h

Expresada en binario: 0110000110010000b

Corresponde al segmento (011b=3).

El desplazamiento en este caso es de 190h=400d. En este caso tenemos un desplazamiento mayor a la longitud del segmento 3 (que tiene una longitud de 356), por lo que el sistema lanzará una **excepción por violación del tamaño del segmento**.

PED 2. EJERCICIO 5.

a) Determinar el tamaño de cada uno de los campos de una dirección virtual y de una dirección física.

Dirección física.

Para determinar el tamaño de los campos de una dirección física necesitamos conocer el tamaño de la memoria principal y el número de marcos en los que esta se divide.

Sabemos que el tamaño de la memoria C_{mp} es de $1\text{MiB}=2^{20}\text{bytes}$, suponiendo que la unidad direccionable es la palabra y que el tamaño de palabra es un byte, el tamaño de la memoria es de 2^{20} palabras, direccionables mediante 20 bit ($n=20$).

El número de marcos (N_f) vendrá dado por el cociente entre el tamaño de la memoria principal y el tamaño de cada marco, que será igual al tamaño de página (S_p):

$$N_f = \text{floor}(C_{mp}/S_p) = \text{floor}(1 \text{ MiB}/1\text{KiB}) = \text{floor}(2^{10}\text{KiB}/1\text{KiB}) = 2^{10} \text{ marcos de página.}$$

Para direccionar dicha cantidad de marcos serán necesarios 10 bits ($f=10$).

Podemos calcular el desplazamiento como: $d=n-f=10$.

marco	Desplazamiento
$f=10$	$d=10$

Dirección virtual:

Para el cálculo de los campos referentes a una dirección virtual necesitamos conocer el tamaño del espacio virtual y el número de páginas en que se descompone dicho espacio.

Sabemos que el tamaño de memoria virtual es de 4 MiB. Para direccionar los $4\text{MiB}=4*2^{20} = 2^{22}$ necesitamos 22 bits ($m=22$).

En este caso el número de páginas en memoria virtual vendría dado por el cociente entre el tamaño total de la memoria virtual (CA) y el tamaño de página:

$$N_p = CA/S_p = \text{ceil}(4\text{MiB}/1\text{KiB}) = 2^{12} \text{ páginas, para lo cual son necesarios 12 bits de dirección (p=12).}$$

El desplazamiento d vendría dado por la diferencia $d=m-p=22-12=10$, que coincide con el tamaño desplazamiento en la dirección física, tal y como era de esperar.

página	Desplazamiento
--------	----------------

p=12	d=10
------	------

b) Capacidad mínima de la tabla de procesos.

La técnica de memoria virtual permite ejecutar procesos con un espacio lógico mayor al tamaño de la memoria principal, permitiendo que no todo el se encuentre cargado en memoria principal.

Por este motivo el proceso de mayor tamaño permitido, será aquel que ocupe el total de la memoria virtual (4MiB), que equivale a 2^{12} páginas, que se corresponden con igual número de entradas de la página en la tabla de procesos.

La información mínima que debe guardarse en cada entrada de página será, el número de marco correspondiente a dicha página en memoria virtual y tres bits adicionales que indiquen si la página ha sido referenciada, modificada y si es válida (está alojada en memoria principal).

Para almacenar el número de marcos, tal como se ha calculado en la sección anterior, son necesarios $f=10$ bits. Por lo tanto, el tamaño mínimo de cada entrada (E) de tabla será:

$$E=10+3=13 \text{ bit.}$$

Sin embargo para que el procesador pueda acceder a la información de cada entrada de forma indizada lo lógico es reservar el múltiplo de palabras mayor a la cantidad E. Es decir, para almacenar la entrada de tabla emplearíamos un mínimo de dos palabras ($E'=16 \text{ bit}=2 \text{ byte}$).

Dado que el proceso más grande tiene 2^{12} páginas, el tamaño mínimo de la tabla para dicho proceso será de:

$$C_{tp}=N_p * E' = 2^{12} * 2 \text{ byte} = 8192 \text{ byte} = 8 \text{ kbytes.}$$

b.2. Tanto por ciento de la memoria principal.

Para calcular el porcentaje de memoria principal que ocuparía la tabla, simplemente se realiza el cociente entre el tamaño de la tabla y el tamaño de la memoria principal obteniéndose el siguiente resultado:

$$\% = C_{tp}/C_{mp} * 100 = 8 \text{ Kbyte}/1\text{MiB} * 100 = 8 \text{ kB}/1024\text{kB} * 100 = \mathbf{0,78\%}$$