

PED2

1. Explique razonadamente si las siguientes afirmaciones son verdaderas o falsas:

I) (1 p) La sobrepaginación aumenta el porcentaje de uso del procesador.

Falso. Cuando hay pocas páginas de procesos cargadas en memoria principal el procesador se encuentra con poco trabajo. Para aprovechar el procesador es necesario aumentar el número de páginas cargadas, llegando a un punto determinado donde el aprovechamiento es el máximo. El problema llega cuando se sobrepasa este punto consiguiendo que la memoria principal sea incapaz de contener los conjuntos de trabajo de todos los procesos cargados, produciendo fallos de página constantemente –sobrepaginación- y que el procesador se encuentre más ocioso ya que hay procesos bloqueados esperando mientras ocurre la lectura y escritura de páginas en memoria secundaria.

II) (1 p) 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.

Falso. La forma más rápida de actuar cuando ocurre un fallo de página es utilizar un marco libre de memoria principal para cargar la página a la que hace referencia la dirección virtual que produjo dicho fallo. La estrategia de *buffering de páginas* consiste en tener una lista de marcos libres. Estos marcos pueden estar libres o contener una página que es susceptible de ser reemplazada. Cuando ocurre el fallo de página se utiliza uno de estos marcos para cargarla, aunque antes siempre se comprueba que la página que debe ser cargada no está ya en uno de esos marcos libres pero marcada como reemplazable, para así ahorrarse la nueva carga.

2. (2 p) Un sistema con memoria virtual mediante demanda de páginas utiliza el algoritmo LRU para la sustitución de páginas. Un proceso genera la siguiente secuencia de referencias a páginas de memoria:

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

a) Determinar cuántos fallos de página se producen cuando se dispone de 4 o 5 marcos de página para este proceso.

- 4 marcos de página:

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

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

F F F F A F F A F F F F F F A A F F F F

	1	1	1	1	1	1	1	1	3	3	3	3	4	4	4	4	4	4	3	3	Marco j
		3	3	3	3	5	5	5	5	2	2	2	2	5	5	5	5	8	8	8	Marco k
			2	2	2	2	7	7	7	7	8	8	8	8	8	8	1	1	1	1	Marco l
				4	4	4	4	4	4	4	4	9	9	9	9	9	9	9	9	2	Marco m

TOTAL FALLOS DE PÁGINA CON 4 MARCOS = 16 FALLOS (F) Y 4 ACIERTOS (A)

- 5 marcos de página:

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

						5	7	4	3	2	8	9	4	5	4	9	1	8	3	2	PILA
				4	1	1	5	7	4	3	2	8	9	4	5	4	9	1	8	3	
			2	2	4	4	1	5	7	4	3	2	8	9	9	5	4	9	1	8	
		3	3	3	2	2	4	1	5	7	4	3	2	8	8	8	5	4	9	1	
	1	1	1	1	3	3	2	2	1	5	7	4	3	2	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	2	1	1	1	1	Marco j
		3	3	3	3	3	7	7	7	7	7	9	9	9	9	9	9	9	9	9	9	Marco k
			2	2	2	2	2	2	3	3	3	3	3	5	5	5	5	5	5	3	3	Marco l
				4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	Marco m
						5	5	5	5	5	8	8	8	8	8	8	8	8	8	8	8	Marco n

TOTAL FALLOS DE PÁGINA CON 5 MARCOS = 14 FALLOS (F) Y 6 ACIERTOS (A)

b) Explicar razonadamente si mejoraría la tasa de fallos de página si se aumentase el número de marcos de página a N , siendo $N > 5$.

Cuanto mayor sea N ocurrirán menos fallos de página puesto que el número de páginas cargadas también será mayor y la probabilidad de no encontrarla, es decir, de producirse un fallo, será menor. El caso extremo sería aquel en el que el número de marcos fuese igual al número de páginas y por lo tanto no habría nunca fallos (salvo en la primera vez que se cargase cada página) puesto que todas las páginas estarían cargadas. Obviamente tener cargadas todas las páginas de todos los procesos en memoria principal no es posible.

3. (2 p) Explique razonadamente las funciones que realizan las capas de software de E/S del núcleo de un sistema operativo.

Como norma general, el software del núcleo del SO necesario para gestionar las operaciones de E/S se organiza en tres capas:

- El subsistema de E/S: Es el encargado de realizar todas las operaciones E/S que son comunes a todos los dispositivos e independientes de los mismos, es decir, le da lo mismo el dispositivo para el que trabaja puesto que trata a todos de la misma forma. Entre las tareas a las que se dedica el subsistema de E/S podemos diferenciar:

- *La asignación y liberación de dispositivos dedicados*: si un dispositivo sólo permite ser usado por un proceso cada vez –dispositivo dedicado-, el subsistema de E/S se encargará de conseguir que así sea y dos procesos distintos no puedan interactuar a la vez con dicho dispositivo.

- *Bloqueo de procesos que solicitan una operación de E/S*: si le llega una petición de un proceso para la realización de una operación de E/S puede pasarlo al estado bloqueado dependiendo de si el proceso realizó una llamada al sistema de tipo bloqueante o del tipo y estado del dispositivo involucrado.

- *Planificación de la E/S*: Esta capa también es la encargada de organizar las operaciones de E/S para optimizar el uso de los recursos. Puede hacerlo mediante el uso de las colas de dispositivos siguiendo un determinado criterio pero siempre teniendo en cuenta que algunas peticiones de E/S tienen mayor prioridad que otras; por ejemplo las del núcleo frente a las de usuario o las de escritura frente a las de lectura en caso de no disponer de suficiente espacio en memoria.

- *Invocación del driver adecuado*: aunque el subsistema de E/S trate a todos los dispositivos de igual manera, cada uno de ellos implementa las operaciones de forma distinta. Para que cada dispositivo funcione correctamente hay que elegir el driver que le corresponde, el driver que conoce las operaciones que ese dispositivo efectúa. Esta tarea también es cosa del subsistema de E/S.

- *Almacenamiento temporal de datos de E/S o buffering*: los datos que se leen o se escriben en cada operación son almacenados temporalmente en un área de la memoria principal (llamado buffer). El subsistema de E/S es el encargado de asignar este espacio de memoria.

- *Proporcionar un tamaño de bloque uniforme a los niveles superiores de software*: hacer que las capas superiores de software no tengan que conocer el tamaño de los bloques físicos del dispositivo de E/S y siempre trabajar con tamaños de bloques lógicos.

- *Gestión de los errores producidos en una operación de E/S*: decidir qué hacer cuando ocurre un error. Si el error es de programación el subsistema de E/S notificará al proceso involucrado el error ocurrido. Si el error es en el dispositivo el driver,

después de intentar solucionarlo, notificará al subsistema el error ocurrido y éste decidirá qué hacer; abortar el proceso, ignorar el error, notificar el error al proceso, ...

Además, el subsistema de E/S debe proporcionar una interfaz para los drivers. La interfaz consta de dos tipos de rutina: una formada por las rutinas que puede invocar el subsistema de E/S para interactuar con el driver y pedirle, por ejemplo, que lea o escriba; la otra está formada por las rutinas del núcleo del SO que un driver puede ejecutar. Esta interfaz suele ser uniforme para que sean los dispositivos los que dependan del sistema y no al revés, y así evitar que haya que recompilar y enlazar el código del SO cada vez que haya un nuevo dispositivo de E/S.

- Drivers de dispositivos de E/S: Cada dispositivo de E/S trabaja de manera diferente y posee distintas capacidades. Un driver de dispositivo contiene el código necesario para hacer posible que el SO interactúe con dicho dispositivo. Así es posible tener un driver para el teclado, otro para el ratón, otro para la pantalla, etc. Estos drivers son proporcionados por los fabricantes de los dispositivos y son los encargados de ofrecer al subsistema de E/S las funciones necesarias para hacerlo funcionar, siendo a su vez capaz de invocar ciertas rutinas del SO como se ha comentado en el anterior apartado.

Una de las ventajas de los drivers es que su código no se modifica y, por lo tanto, es posible ejecutar varias instancias a la vez de un mismo driver.

- Manejadores de interrupciones: Los manejadores de interrupciones forman parte del núcleo del SO y son extremadamente dependientes del hardware, además de tener una alta prioridad de ejecución para conseguir un rendimiento óptimo. Las acciones que los manejadores llevan a cabo dependen del tipo de interrupción. No se actúa de la misma manera cuando finaliza una operación de E/S salida que cuando un driver se bloquea esperando que el controlador se encuentre listo para procesar otra operación de E/S. Su código suele ser pequeño y rápido de ejecutar, una reacción automática a un determinado suceso –interrupción–.

4. En un computador con una capacidad de memoria principal de 64 kibipalabras se utiliza gestión de memoria mediante segmentación. La tabla de segmentos (todos los datos numéricos están en decimal) es la siguiente:

Nº de segmento	Base	Longitud
0	0	7230
1	16384	8191
2	32768	1024
3	8192	356
4	24576	4200

Se pide:

a) (1 p) Supuesto que una dirección lógica tiene el mismo tamaño en bits que una dirección física y que consta de los campos [nº de segmento, desplazamiento], determinar el tamaño en bits de cada uno de estos campos.

Asumiendo que una palabra es igual a 8 bits = 1 byte:

nº de segmento = s, desplazamiento = d.

Puesto que hay 5 segmentos s tendrá un tamaño de 3 bits: $2^2 < 5 < 2^3$

- Segmento 0:

Longitud 7230 palabras; $7230 < 8192 = 2^{13} \rightarrow d = 13$ bits.

Tamaño = $s + d = 3 + 13 = 16$ bits.

- Segmento 1:

Longitud 8191 palabras; $8191 < 8192 = 2^{13} \rightarrow d = 13$ bits.

Tamaño = $s + d = 3 + 13 = 16$ bits.

- Segmento 2:

Longitud 1024 palabras; $1024 = 2^{10} \rightarrow d = 10$ bits.

Tamaño = $s + d = 3 + 10 = 13$ bits.

- Segmento 3:

Longitud 356 palabras; $356 < 512 = 2^9 \rightarrow d = 9$ bits.

Tamaño = $s + d = 3 + 9 = 12$ bits.

- Segmento 4:

Longitud 4200 palabras; $4200 < 8192 = 2^{13} \rightarrow d = 13$ bits.

Tamaño = $s + d = 3 + 13 = 16$ bits.

b) (1 p) Determinar a qué direcciones físicas expresadas en decimal corresponden las siguientes direcciones lógicas expresadas en hexadecimal:

i) 11AE₁₆,

Pasamos a binario = 0001 0001 1010 1110.

De acuerdo al apartado anterior sabemos que los 3 bits más significativos corresponden al nº de segmento:

$000_2 = 0_{10} =$ segmento 0.

El resto es el desplazamiento:

$1000110101110_2 = 4526_{10}$

Dirección: [0, 4526]

ii) 6190₁₆,

Pasamos a binario = 0110 0001 1001 0000.

De acuerdo al apartado anterior sabemos que los 3 bits más significativos corresponden al nº de segmento:

$011_2 = 3_{10} =$ segmento 3.

El resto es el desplazamiento:

$0000110010000_2 = 400_{10}$

Dirección: [3, 400]

Como el desplazamiento de 400 es mayor que la longitud total del segmento 3 que es de 356, se produciría un error de direccionamiento por violación del tamaño del segmento.

5. La política de gestión de memoria de un cierto sistema es del tipo demanda de página. El tamaño de una página es de 1 KiB, el tamaño máximo de la memoria virtual es de 4 MiB y el tamaño de la memoria física es de 1 MiB. Se pide:

a) (1 p) Determinar el tamaño de cada uno de los campos de una dirección virtual y de una dirección física.

Dirección física = número de marco de página de f bits y desplazamiento dentro del marco de d bits:

Tamaño total n en bits de una dirección física = $\min_n \{C_{MP} \leq 2^n\}$

$C_{MP} = 1 \text{ MiB} = 2^{20} \text{ bytes}$ dividido por la longitud de 1 palabra (que en este caso una palabra = 8 bits = 1 byte) = $2^{20}/1 = 2^{20}$ palabras $\rightarrow n = 20$ bits.

Del tamaño de una página S_P expresado en palabras se puede determinar el tamaño d en bits del campo desplazamiento tanto de una dirección física como de una virtual resolviendo: $\min_d \{S_P \leq 2^d\}$

$S_P = 1 \text{ KiB} = 2^{10} \text{ bytes}$ dividido por la longitud de 1 palabra (que en este caso una palabra = 8 bits = 1 byte) = $2^{10}/1 = 2^{10}$ palabras $\rightarrow d = 10$ bits.

El tamaño f del campo número de página se puede obtener de la siguiente forma: $f = n - d = 20 - 10 = 10$ bits.

Dirección virtual = número de página de p bits y desplazamiento dentro de la página de d bits.

Tamaño total en bits de una dirección virtual = $\min_m \{C_A \leq 2^m\}$

$C_A = 4 \text{ MiB} = 2^{22} \text{ bytes}$ dividido por la longitud de 1 palabra (que en este caso una palabra = 8 bits = 1 byte) = $2^{22}/1 = 2^{22}$ palabras $\rightarrow m = 22$ bits.

El tamaño del campo desplazamiento de una dirección virtual tiene el mismo tamaño que una dirección física = $d = 10$ bits.

El tamaño p del campo número de página se puede obtener de la siguiente manera: $p = m - d = 22 - 10 = 12$ bits.

b) (1 p) Determinar la capacidad mínima que debe tener la tabla de páginas del proceso de mayor tamaño que se puede ejecutar en el sistema. ¿Qué tanto por ciento de la memoria principal ocuparía dicha tabla?

Como mínimo, una entrada de una tabla de páginas contiene el marco de página donde se aloja la página y un bit denominado presente/ausente o validez/invalides para indicar si la página está cargada en memoria principal.

Si a esto añadimos que en el anterior apartado calculamos que el marco de página consta de 10 bits, obtenemos que cada entrada de la tabla = $E = 10 + 1 = 11$ bits.

El número de páginas $N_P = \text{ceil} \left(\frac{C_A}{S_P} \right) = \text{ceil} \left(\frac{4 \text{ MiB}}{1 \text{ KiB}} \right) = 2^{12}$ páginas.

El tamaño de la tabla = $E * N_P = 11 * 2^{12} = 45056 \text{ bits} = 5632 \text{ bytes/palabras}$.

$$\text{El porcentaje P de memoria principal} = \frac{\text{CTP}}{\text{CMP}} * 100 = \frac{5632 \text{ bytes}}{1 \text{ MiB}} * 100 = \frac{5632}{1048576} * 100 = 0,537109375 \text{ \%}.$$