UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela Universitaria de Informática

Calificación
1
2
3
4

SISTEMAS OPERATIVOS - SEGUNDA PARTE
Examen Convocatoria Ordinaria, 22 de junio de 2006

Nombre	Titulación
SOLUCIONES	

Dispone de dos horas y media para realizar el examen

---

1 (2.5 puntos) Supongamos que tenemos una máquina con 16 MB de memoria principal y un esquema de gestión de memoria virtual paginado con páginas de 4 KB. Un proceso produce la siguiente secuencia de accesos a direcciones de memoria (mostradas aquí en hexadecimal):

02D4B8, 02D4B9, 02D4EB, 02D4EB, 02D86F, F0B621, F0B815, F0D963, F0B832, F0BA23, D9D6C3, D9B1A7, D9B1A1, F0BA25, 02D4C7, 628A31, F0B328, D9B325, D73425.

El sistema operativo asigna al proceso 4 marcos de memoria principal. Se pide:

a) Dar la *cadena de referencia* de las páginas accedidas por el proceso. (1 p)

b) Si el sistema operativo utiliza 4 marcos de memoria principal, describir el comportamiento del gestor de memoria utilizando cada uno de los siguientes algoritmos de reemplazo de páginas, indicando cuántos fallos de página se producen con cada algoritmo:

b.1) Algoritmo FIFO. (0.75 p)

b.2) Algoritmo de la segunda oportunidad. (0.75 p)

Solución:

a) A partir del tamaño y organización de la memoria, se calcula el formato de una dirección física:

16 MB = 2^{24} bytes, es decir, se requieren 24 bits para direccionar la memoria física. Como además se nos indica que el tamaño de página es de 4 KB, tendremos que

4 KB = 2^{12} bytes, por lo que se necesitan 12 bits para direccionar un byte de una página.

Por tanto, el formato de una dirección física de memoria será:

página	desplazamiento
12 bits	12 bits

Dado que la cadena de referencias viene dada en hexadecimal, y para expresar un número hexadecimal se necesitan 4 bits en binario, para direccionar la página se

necesitan 3 dígitos hexadecimales y otros 3 para el desplazamiento. Por tanto la una dirección física queda expresada según el siguiente formato:

página			desplazamiento		
12 bits			12 bits		
hex	hex	hex	hex	hex	hex
0	2	D	4	B	8

Así, la secuencia de páginas que formará la cadena de referencias es:
02D, F0B, F0D, F0B, D9D, D9B, F0B, 02D, 628, F0B, D9B, D73

b.1) FIFO

02D	F0B	F0D	F0B	D9D	D9B	F0B	02D	628	F0B	D9B	D73
02D	02D	02D		02D	F0B		F0D	D9D	D9B		02D
	F0B	F0B		F0B	F0D		D9D	D9B	02D		628
		F0D		F0D	D9D		D9B	02D	628		F0B
				D9D	D9B		02D	628	F0B		D73
F	F	F		F	F		F	F	F		F

En total se producen 9 fallos de página.

b.2) Segunda Oportunidad

02D	F0B	F0D	F0B	D9D	D9B	F0B	02D	628	F0B	D9B	D73
$02D_1$	**$02D_1$**	**$02D_1$**		**$02D_1$**	$D9B_1$	$D9B_1$	$D9B_1$	**$D9B_1$**	**$D9B_1$**		$D73_1$
	$F0B_1$	$F0B_1$		$F0B_1$	**$F0B_0$**	**$F0B_1$**	$F0B_0$	$F0B_0$	$F0B_1$		**$F0B_0$**
		$F0D_1$		$F0D_1$	$F0D_0$	$F0D_0$	$02D_1$	$02D_1$	$02D_1$		$02D_0$
				$D9D_1$	$D9D_0$	$D9D_0$	**$D9D_0$**	628_1	628_1		628_0
F	F	F		F	F		F	F			F

En total se producen 8 fallos de página.

Nombre

2 (2.5 puntos) Considérese el siguiente estado seguro de un sistema donde existen cinco procesos (A, B, C, D, E) y cuatro recursos (r1, r2, r3, r4) con un total, respectivamente, de (6, 7, 12, 12) ejemplares:

	recursos asignados			
	r1	r2	r3	r4
A	0	0	1	1
B	2	0	0	0
C	0	0	3	4
D	2	3	5	4
E	0	3	3	2

	recursos que aún se necesitan			
	r1	r2	r3	r4
A	0	0	0	1
B	0	7	5	0
C	6	6	2	2
D	2	0	0	2
E	0	3	2	0

Se pide:

a) Si el proceso C solicita un recurso del tipo r1 y otro del tipo r2, ¿cuál sería la respuesta del sistema si sabemos que utiliza el algoritmo del banquero para tomar su decisión? (0.75 p)

b) A continuación, el proceso B termina. ¿Qué acciones tomaría el sistema? Una vez realizadas dichas acciones, ¿quedaría el sistema en un estado seguro? (1.75 p)

a) A partir de los datos del enunciado, teniendo en cuenta los recursos asignados de cada tipo (4,6,12,11) y los ejemplares totales de cada tipo (6,7,12,12), podemos determinar el número de ejemplares disponibles (2,1,0,1).

Si se concediera la petición del proceso C (1,1,0,0), la situación que tendríamos sería la siguiente:

	recursos asignados				Necesidades máximas			
	r1	r2	r3	r4	r1	r2	r3	r4
A	0	0	1	1	0	0	0	1
B	2	0	0	0	0	7	5	0
C	1	1	3	4	5	5	2	2
D	2	3	5	4	2	0	0	2
E	0	3	3	2	0	3	2	0

Quedando disponibles (1,0,0,1) y para determinar si se le concede será necesario encontrar al menos una secuencia segura. Con los recursos disponibles y las necesidades máximas dadas, el único proceso que podría terminar sería el proceso A. En ese caso, los ejemplares disponibles asumiendo que tomamos A como primer proceso de la secuencia segura serían (1,0,1,2). Con este número de ejemplares no es posible encontrar ningún otro proceso que pueda terminar por lo que podemos

concluir que no existe ninguna secuencia segura y por tanto, el sistema operativo no asignaría los recursos al proceso C, dejándolo bloqueado hasta que sea posible atender dicha solicitud de recursos.

b) Al terminar el proceso B, los recursos asignados a dicho proceso (2,0,0,0) pasarían a estar disponibles:

(2,1,0,1) [disponibles antes de terminar B] + (2,0,0,0) [liberados por B] = (4,1,0,1)

Ante esta nueva situación, el sistema se debería replantear la posibilidad atender la solicitud pendiente del proceso C. Por tanto la nueva situación asumiendo que dicha petición es atendida sería la siguiente:

	recursos asignados				Necesidades máximas			
	r1	r2	r3	r4	r1	r2	r3	r4
A	0	0	1	1	0	0	0	1
C	1	1	3	4	5	5	2	2
D	2	3	5	4	2	0	0	2
E	0	3	3	2	0	3	2	0

Quedando disponibles (3,0,0,1). En este caso si existiría la siguiente secuencia segura {A, D, E, C}

	recursos asignados				Disponibles			
	r1	r2	r3	r4	r1	r2	r3	r4
A	0	0	1	1	3	0	1	2
D	2	3	5	4	5	3	6	6
E	0	3	3	2	5	6	9	8
C	1	1	3	4	6	7	12	12

Por tanto el sistema quedaría en un estado seguro y la solicitud del proceso C sería atendida en este momento, quedando dicho proceso listo para ejecución cuando el planificador lo decida.

---

3 (2.5 puntos) En el nuevo parque que recientemente se ha inaugurado en Siete Palmas y debido a la gran aceptación se ha establecido un aforo máximo de usuarios (N) por razones de seguridad. Para ello se han instalado unos torniquetes a la entrada y a la salida que controlan en todo momento el número de personas en el parque. Cuando el aforo está completo, se debe esperar en la entrada hasta que alguien salga del parque. La afluencia de gente es tal que se forman largas colas de espera, por lo que se ha decidido crear una entrada preferente para los vecinos residentes en la zona, de forma que cuando el parque está lleno, tienen preferencia sobre los no residentes a la hora de acceder a las instalaciones.

Se pide realizar, mediante monitores y variables condición, el código de entrada y salida del parque, de forma que cuando alguien abandona el parque, en caso de que existan personas esperando debido a que el cupo esté agotado, tendrán preferencia los residentes frente a los no residentes.

Nombre

En la implementación que se muestra se ha asumido que las variables condición están implementadas siguiendo la semántica del estilo Hoare.

```
const int N=…;  //Aforo del parque
Monitor ControlParque {
        Public:
                ControlParque(int capacidad);
                ~ControlParque();
                void Entra_Residente();
                void Entra_NoResidente();
                bool Salir();
        Private:
                condition: *espera_residentes, *espera_noresidentes;
                int nper;  // Número de personas dentro del parque
                int aforo;  // Aforo del parque
                int nresbloq; // Número de personas residentes esperando
                int nnresbloq; // Número de personas no residentes esperando
};

ControlParque:: ControlParque( int capacidad) {
        aforo = capacidad;
        nper = 0;
        nresbloq = 0;
        nnresbloq = 0;
        espera_residentes = new condition;
        espera_noresidentes = new condition;
}

ControlParque:: ~ ControlParque () {
        delete espera_residentes;
        delete espera_noresidentes;
}

ControlParque:: Entra_Residente () {
        if ( nper == aforo ) {
                nresbloq++;
                espera_residentes->wait();
                nresbloq--;
        }
        nper++;
}
```

```
ControlParque:: Entra_NoResidente () {
        if (nper == aforo) {
                nnresbloq++;
                espera_noresidentes->wait();
                nnresbloq--;
        }
        nper++;
}

ControlParque:: Salir () {
        nper--;
        if (nresbloq>0)
                espera_residentes->signal();
        else if (nnresbloq>0)
                espera_noresidentes->signal();
}

ControlParque miParque(N);
Código de las personas residentes {
        …
        // Sección de entrada al parque
        miParque.Entra_Residente();
        …  //Jugar en el parque
        // Sección de salida del parque
        miParque.Salir();
}

Código de las personas no residentes {

        …
        // Sección de entrada al parque
        miParque.Entra_NoResidente();
        …  //Jugar en el parque
        // Sección de salida del parque
        miParque.Salir();
}
```

---

4 (2.5 puntos) Responder cada una de las siguientes cuestiones sobre ficheros:

a) Supongamos que tenemos un sistema de archivos que trabaja con bloques de datos de 1 kilobyte y que utiliza una FAT en la que cada enlace ocupa 24 bits. ¿Cuál es el tamaño máximo que podría tener un fichero en este sistema?

Si cada enlace de la FAT ocupa 24 bits, se admiten 2^{24} valores posibles. Si todos los valores representan direcciones de bloque, tenemos que se pueden referenciar como máximo 2^{24} bloques de 1 kilobyte, lo que equivale a un fichero que ocupa $2^{24} \times 2^{10} = 2^{34}$ bytes. Es decir, **16 gigabytes**.

Nombre

En realidad, el tamaño máximo permitido será algo menor, porque algunos de los valores en la entrada de la FAT se usarán para indicar otras circunstancias: final de fichero, bloque dañado, etc.

b) Supongamos ahora que éste mismo sistema de archivos utiliza un bloque de índices, en lugar de una FAT. ¿Cuál sería entonces el tamaño máximo de un fichero?

Si se utiliza un bloque de índices, el tamaño máximo del archivo está condicionado por el número de entradas que caben en dicho bloque. Tenemos que cada entrada ocupa 24 bits, que son 3 bytes. Por tanto, en un bloque de datos de 1 kilobyte caben 1024 / 3 = 341 entradas (sobra un byte, que no se puede aprovechar). Así pues, el fichero más grande que podemos referenciar tiene 341 bloques de 1 kilobyte, que suman un total de **341 kilobytes**.

c) Tenemos un sistema de archivos que utiliza asignación indexada de dos niveles de profundidad. El tamaño de bloque es de 2 kilobytes y cada entrada en la tabla de índices ocupa 2 bytes. Un fichero que necesita siete bloques de índices, ¿qué tamaño puede tener? (nota: calcular el tamaño mínimo y el máximo que podría tener).

Si un fichero tiene asociados siete bloques de índices, necesariamente uno de ellos será el bloque de primer nivel y los otros seis, los bloques de segundo nivel. Cada bloque de índices es capaz de albergar 2048 / 2 = 1024 apuntadores a bloques. Por tanto, los seis bloques de segundo nivel son capaces de referenciar 1024 × 6 = 6144 bloques. Si todas esas entradas estuvieran al completo, el fichero tendría un tamaño de $6144 \times 2048 = 1024 \times 6 \times 2048 = 2^{10} \times 6 \times 2^{11} = 6 \times 2^{21} = 12 \times 2^{20}$ = **12 megabytes**.

El fichero más pequeño que debe consumir seis bloques de segundo nivel sería uno que ocupara cinco bloques de índices completos y sólo una entrada del sexto bloque. En este caso, serían 1 + (1024 × 5) = 5121 bloques. Como estamos hablando del fichero más pequeño posible, tenemos que suponer que del último bloque de datos sólo se utiliza un byte, Por tanto, el tamaño ocupado por el fichero sería: lo que ocupan los primeros 5120 bytes, más un byte del último bloque. Es decir, $1 + 5120 \times 2048 = 1 + (10 \times 2^9 \times 2^{11}) = 1 + 10 \times 2^{20}$ = **10 megabytes más un byte**.

Así pues, el fichero puede tener un tamaño entre **10 y 12 megabytes (exactamente, de 10 485 761 bytes a 12 582 912 bytes).**

d) Tenemos una partición de disco de 4 gigabytes en la que se va a montar un sistema de archivos basado en FAT y que usará un tamaño de bloque de datos de 4 kilobytes. ¿Cuántos bits deberían usarse en cada entrada de la FAT para que el sistema de archivos pudiera direccionar toda la partición?

Cuatro gigabytes son $4 \times 2^{30} = 2^{32}$ bytes. Los bloques son de 4 kilobytes, es decir $4 \times 2^{10} = 2^{12}$ bytes.

Por tanto, el sistema manejará en torno a $2^{32} / 2^{12} = 2^{20}$ bloques de datos.

Eso significa que las entradas de la FAT deberían tener como mínimo **20 bits**.

Este resultado dependería también de cuántos bloques de datos realmente quedarán disponibles para ser utilizados por ficheros, y de los valores reservados de la FAT para señalar otra información que no sean enlaces.