RELACIÓN TEMA 2

GRUPO 4

Sara Martín Rodríguez
Laura Salas López
Marta Zhao Ladrón de Guevara Cano
Leandro Jorge Fernández Vega

1. Dado un proceso en un SO con su información de contexto, de datos y de código según se muestra en la figura y que ya ha sido atendido en un 50% y le resta la otra mitad para finalizar su ejecución. Con la idea de optimizar el espacio de memoria para que el SO pudiera disponer de un mayor número de procesos en ésta, ¿podría reducirse el espacio que ocupa en memoria en alguna de las siguientes instancias?

No, porque para reducir el espacio se debe borrar información y las siguientes opciones no se pueden eliminar por las siguientes razones

(a) La lista de procesos. Porque el sistema operativo podría necesitar en un futuro dichos procesos
(b) Información del contexto del proceso. Porque guarda información de los registros del procesador por tanto en caso de pausar el proceso el sistema operativo al volver al proceso no sabría por donde se había quedado.
(c) Tamaño de los datos. Porque guarda en qué parte de la memoria principal se encuentran los datos del proceso, por tanto en lo que queda de proceso no se podrían consultar dichos datos
(d) Tamaño del código. Porque hay instrucciones que aún se pueden necesitar.

En general, el PBC de un proceso almacena el estado completo del proceso en un instante, por tanto no se debe eliminar para liberar memoria hasta que no se haya completado el proceso. Esta estructura permite el desarrollo de técnicas potentes que aseguren la coordinación y la cooperación entre los procesos.

2. ¿Por qué cuando un proceso está en modo "ejecutándose" y pretende acceder a una dirección de memoria fuera del área asignada, se informa de que se ha producido un error en la ejecución? ¿Quién informa de ello? Razone la respuesta.

Cada proceso tiene un espacio en memoria reservado. Un proceso no puede acceder a direcciones de memoria fuera de ese espacio porque podría escribir en una celda de memoria que no le corresponde y provocar fallos en el sistema. Es la Unidad de Gestión de Memoria (MMU) la que se da cuenta de que la dirección de memoria está fuera del área asignada.

El SO informa de ello al usuario.

3. ¿Tiene sentido un modelo de 5 estados de los procesos en un SO monousuario? Razone la respuesta. perfecto :)

Sí, ya que un SO monousuario puede ser multiprogramado, lo que implica que más de un proceso puede ejecutarse simultáneamente y los datos e instrucciones se encuentran en memoria principal. Por tanto, es perfectamente compatible con cargar los programas en memoria y las pausas que incluye este modelo.

4. Dado un proceso que está en modo "ejecutándose" y pretende acceder a una dirección de memoria fuera del área asignada, lo cual sería un error en la ejecución, ¿a qué modo pasaría dicho proceso? Razone la respuesta.
(a) Bloqueado.
(b) No cambia de modo.
(c) Finalizado.
(d) Preparado.

Se produce el paso de "Ejecutándose" a "Finalizado" pues al intentar acceder a una zona de memoria no permitida, se produce un error no recuperable en tiempo de ejecución.

5. Un planificador de procesos tiene una tarea concreta dentro de un SO multiprogramado. ¿Tiene sentido disponer de un planificador de procesos en un SO monoprogramado? Razone la respuesta. perfecto :)

No, un sistema operativo monoprogramado ejecuta a la vez un solo proceso por tanto no tiene sentido que disponga de un planificador de procesos.

6. Dado un SO multiprogramado, ¿bajo qué circunstancias se podría prescindir del planificador de procesos? Razone la respuesta.

El SO multiprogramado es el que permite que se ejecute más de un proceso simultáneamente.

La respuesta más general es que no. Sin embargo, la única excepción es cuando solo se esté llevando a cabo un proceso (pues si solo hay uno no hay que realizar intercambio de procesos) o cuando haya tantos procesadores como procesos (ya que cada procesador se encarga de un proceso).

7. Diga cuales de las siguientes operaciones pueden realizarse únicamente en modo supervisor, o modo kernel:

a) Consultar la hora del sistema.
b) Cambiar la fecha del sistema. Porque multitud de procesos dependen de ella.
c) Leer una pista/sector de un disco magnético. Como es una operación de bajo nivel también es necesario cambiar a modo kernel.
d) Generar una interrupción software. **Es posible en ambos modos.**

e) Generar una interrupción. Solo si dicha interrupción es hardware
f) Modificar la dirección de un vector de la tabla de vectores de interrupción.
g) Deshabilitar las interrupciones.

8. En el caso de un ordenador que se vaya a usar únicamente para un único usuario, ¿qué interés puede tener la existencia de los modos de funcionamiento supervisor/usuario?

El hecho de que un ordenador sea monousuario no implica que no sea multiprogramado, de hecho esto es una gran ventaja pues permite simultanear tareas, aumentar el rendimiento y la eficacia. Para ello, debe velar por la seguridad y eso implica la existencia de los modos de funcionamiento supervisor/usuario.

La existencia de 2 modos de funcionamiento es una gran herramienta para la prevención de errores y fallos, protección del procesador, de los dispositivos de E/S y la memoria. El funcionamiento dual del sistema operativo evita que haya procesos que puedan acceder a ciertas zonas de la memoria y puedan modificar datos importantes. Esto se debe a que el modo usuario tiene restringido el acceso; tan solo puede acceder a un subconjunto de registros del procesador, a un subconjunto de instrucciones máquina y a un área de la memoria.

Por el contrario, el modo supervisor o kernel tiene acceso a todos los recursos del sistema y por tanto el sistema operativo debe de tener cuidado a la hora de permitir este tipo de modo. De hecho, solo se cambia a modo kernel cuando hay una interrupción, una excepción o una llamada al sistema; pasando de ejecutar código de proceso a código del sistema operativo.

9. Cuestiones sobre procesos, y asignación de CPU:

a) ¿Es necesario que lo último que haga todo proceso antes de finalizar sea una llamada al sistema para finalizar? ¿Sigue siendo esto cierto en sistemas monoprogramados?

Sí, las llamadas al sistema son la forma de comunicación entre los programas de usuario y el Sistema Operativo en tiempo de ejecución que se llevan a cabo a través de las denominadas trap o interrupciones software. En consecuencia, sí es necesario que cuando se termine un proceso se avise al sistema operativo para que pueda reactivar otro proceso en espera o empezar otro nuevo.

Esto no cambia si el sistema es monoprogramado, ya que aunque solo esté en marcha un proceso el sistema debe conocer cuando ha terminado para iniciar el siguiente.

b) Cuando el controlador de un dispositivo produce una interrupción ¿se produce necesariamente un cambio de contexto?, ¿y cuando se produce una llamada al sistema?

No necesariamente, solo cuando el Sistema Operativo pueda ejecutarse y decida llevarlo a cabo. Sin embargo, siempre que el S.O. se ejecute es decir se produzca una interrupción o una llamada se produce un cambio de modo (de usuario a kernel).

c) Cuando un proceso se bloquea, ¿deberá encargarse él directamente de cambiar el valor de su estado en el descriptor de proceso o PCB?

No, cuando un proceso se bloquea significa que requiere alguna información del SO y debe esperar por lo que se produce un cambio de modo. En este cambio, el hardware se encarga automáticamente de salvar como mínimo el PC y PSW y de restaurar en el procesador dicha información. LO hace el SO

d) Sea un proceso que cambia de Ejecutándose a Bloqueado, ¿puede este cambio provocar un cambio de estado en otros procesos? Si es así, ¿en qué casos?

Sí, siempre que los recursos sean suficientes otros procesos en cola podrían cambiar de estado mientras que el actual se encuentra bloqueado. Por ejemplo, de “Preparado” a “Ejecutándose”. Además, si hubiese un proceso anterior bloqueado debería haber pasado a “Preparado” para permitir que este pase a ese estado.

e) Idem para el cambio de estado Bloqueado a Ejecutable.

Sí, ya que para que un proceso pase de “Bloqueado” a “Ejecutable” según el diagrama de los estados debe pasar primero a estar “Preparado” por tanto en el camino se ha tenido que detener o cambiar de estado a otros procesos que estaban en estado “Preparado” o “Ejecutable” .

10. En los primeros ordenadores, cada byte de datos leído o escrito, era manejado directamente por la CPU (es decir, no existía DMA - Acceso Directo a Memoria). ¿Qué implicaciones tenía esta organización para la multiprogramación?

El hecho de que los datos leídos o escritos fueran manejados directamente por la CPU suponía un ralentizamiento. El Acceso Directo a Memoria libera a la CPU de toda esa tarea, y esta puede invertir ese tiempo en otros procesos.

11. ¿Por qué no es el intérprete de órdenes (shell) parte del propio sistema operativo? ¿Qué ventajas aporta el no serlo?

Si formase parte sería peligroso ya que podría ejecutar órdenes en modo kernel y alterar el funcionamiento interno del sistema.

Entre las ventajas encontramos no tener que cambiar de intérprete de instrucciones, no necesitar llamar al SO en cada orden, lo que supone más rapidez de ejecución, y por tanto mayor rapidez de actualización del SO.

Cuando hay que cargar el SO no hay que cargar la shell

Se puede actualizar la shell y se pueden tener varias versiones independientemente del SO.

Se requieren menos recursos para cargar el SO.

12. Para cada una de las llamadas al sistema siguientes, especificar y explicar si su procesamiento por el sistema operativo implica un cambio de contexto:

El cambio de contexto supone el paso de "Ejecutandose" a otro estado o el paso de "Preparado" a "Ejecutándose". NO la hemos corregido, hay que pedirla, la d y e creo que están mal

a) Crear un proceso. NO. Crear un proceso implica cargar la información del proceso en el PCB y sus datos en memoria; es decir, el paso de "Nuevo" a "Preparado".

b) Abortar un proceso, es decir, terminarlo forzosamente. SI. Pues supone el paso de "Ejecutándose" a "Finalizado".

c) Suspender o bloquear un proceso. SI. Pues supone el paso de "Ejecutándose" a "Bloqueado".

d) Reanudar un proceso (inverso al caso anterior). NO. Pues supone el paso de "Bloqueado" a "Preparado". Hay que recuperar el contexto

e) Modificar la prioridad de un proceso. SÍ. Supone el paso de "Ejecutándose" a "Bloqueado".

13. ¿Tiene sentido mantener ordenada por prioridades la cola de procesos bloqueados? Si lo tuviera, ¿en qué casos sería útil hacerlo?

No, porque los procesos bloqueados pueden estarlo por diversas razones que no tengan relación entre sí, es decir puede que cada uno necesite de unos recursos. En el caso de que la causa del bloqueo de todos los procesos fuese la misma, sí sería útil ordenar la cola de los procesos por prioridad con el objetivo de agilizar la ejecución.

14. ¿Por qué se utilizan potencias de dos para los tamaños de página, número de páginas en el espacio lógico de un proceso, y números de marcos de página?

Como las direcciones están en binario, al utilizar potencias de dos para los tamaños se facilita la traducción de la dirección lógica al número de páginas y desplazamiento. Los bits de menor peso indican el desplazamiento dentro de la página, mientras que los de mayor peso indican el número de página.

Se pueden ahorrar operaciones.

15. Sitúese en un sistema paginado, en donde la memoria real tiene un tamaño de 16 Mbytes, una dirección lógica ocupa 32 bits, de los cuales los 22 de la izquierda constituyen el número de página, y los 10 de la derecha el desplazamiento dentro de la página. Según lo anterior.

a) ¿Qué tamaño tiene cada página?
El tamaño es el desplazamiento de la página, es decir, 2^10 Bytes=1KB.

b) ¿En cuántos marcos de página se divide la memoria física?
16 Mbytes = 2^24 bytes. Como el desplazamiento es el mismo tanto para páginas como para sus marcos, 24 - 10 = 14 bits para el número de marco. Por tanto, hay 2^14 marcos.

c) ¿Qué tamaño deberá tener el campo Número de Marco de la Tabla de Páginas?
Deberá de tener 14 bits , que es el número de marco de cada marco de página, para poder direccionar 2^14 marcos.

d) Además de dicho campo, suponga que la Tabla de Páginas tiene los siguientes campos con los siguientes valores: Protección: 1 bit (1= Sólo se permite leer; 0= Cualquier tipo de acceso). ¿Cuál es el tamaño de la Tabla de Páginas para un proceso cuyo espacio de memoria lógico es de 103 Kbytes?
2^10 bytes = 1 KB, por lo que 103 Kbytes/1 Kbyte = 103 páginas, y como cada página presenta 14 bits (el tamaño de página es el mismo que el de marco de página) más ahora uno nuevo de protección , el tamaño de la tabla de páginas será 103*(14+1) = 1545 bits = 193,125 Bytes.

16. Suponga que la tabla de páginas para el proceso actual se parece a la de la figura. Todos los números son decimales, la numeración comienza en todos los casos desde cero, y todas las direcciones de memoria son direcciones en bytes. El tamaño de página es de 1024 bytes.

Número de página virtual	Número de marco de página
0	4
1	7
2	1
3	2
4	10
5	0

¿Qué direcciones físicas corresponderán con cada una de las siguientes direcciones lógicas del proceso?

Tamaño de página = 1024

Dirección física = marco de página*tamaño de página + desplazamiento

a) 999

1º Calcular el número de la página virtual

999/1024 = 0.97559

Nº de página = 0

2º Calcular el marco de página, fijándonos en la tabla y la correspondencia de la página virtual

Nº de marco = 4

3º Calcular el desplazamiento. Lo que queda libre en el número de marco

Desplazamiento = Dirección lógica - nº de página*tamaño de página

Desplazamiento = 999-0*1024 = 999

4º Calcular la dirección física

Dirección física = 4*1024 + 999 = 5095

b) 2121

1º Calcular el número de la página virtual

2121/1024 = 2,0712899

Nº de página = 2

2º Calcular el marco de página, fijándonos en la tabla y la correspondencia de la página virtual

Nº de marco = 1

3º Calcular el desplazamiento. Lo que queda libre en el número de marco

Desplazamiento = Dirección lógica - nº de página*tamaño de página

Desplazamiento = 2121-2*1024 = 73

4º Calcular la dirección física

Dirección física = 1*1024 + 73 = 1097

c) 5400

1º Calcular el número de la página virtual

5400/1024 = 5,273437

Nº de página = 5

2º Calcular el marco de página, fijándonos en la tabla y la correspondencia de la página virtual

Nº de marco = 0

3º Calcular el desplazamiento. Lo que queda libre en el número de marco

Desplazamiento = Dirección lógica - nº de página*tamaño de página

Desplazamiento = 5400-5*1024 = 280

4º Calcular la dirección física

Dirección física = 0*1024 + 280 = 280

17. ¿Qué tipo de fragmentación se produce en un sistema de gestión de memoria paginado? ¿Qué decisiones de diseño se pueden tomar para minimizar dicho problema, y cómo afectan estas decisiones al comportamiento del sistema?

Existen dos tipos de fragmentaciones:

1) Interna: se asignan bloques de memoria de tamaño fijo al proceso sin importar el tamaño de memoria del proceso. Paginación. SI se reduce el tamaño de página, los espacios son menores, pero la tabla de páginas ocupa más
2) Externa: se asignan bloques de memoria de tamaño variable de forma dinámica

La paginación es un tipo de fragmentación de la memoria en dos bloques:

- La memoria física: el espacio hardware necesario para almacenar las direcciones. Esta se divide en bloques de tamaño fijo, denominados marcos de página, dados en potencia de dos. Las direcciones físicas están compuestas por el número de marco (m, marco donde está almacenada la página) y el desplazamiento (d).
- El espacio lógico de cada proceso: el espacio software que ocupa cada proceso. Este se divide en bloques del mismo tamaño, denominados páginas. Las direcciones lógicas están compuestas por el número de página (p) + el desplazamiento en la página(d).

El problema de este tipo de fragmentación es que cuando la CPU genera una dirección lógica es necesario traducirla a la dirección física correspondiente. Para ello se crean las la tabla de páginas que mantiene la información necesaria para realizar dicha traducción.

Esto provoca que se requiera de un doble acceso a memoria: uno a la propia memoria y otro a dicha tabla. Con el fin de solucionar este problema, se implementa el búfer de traducción adelantada o TLB que es una caché hardware de consulta rápida formada por un conjunto de registros que realizan una búsqueda en la tabla en paralelo, de forma que para traducir una dirección, si ya existe en alguno de dichos registros se da directamente el marco (la dirección en la memoria física) y si no se busca en la tabla y se actualiza el TBL.

18. Suponga que un proceso emite una dirección lógica igual a 2453 y que se utiliza la técnica de paginación, con páginas de 1024 palabras

a) Indique el par de valores (número de página, desplazamiento) que corresponde a dicha dirección.

Número de página: 2453/1024 = 2,3955 → página 2
Desplazamiento: 2*1024-2453 = 405

b) ¿Es posible que dicha dirección lógica se traduzca en la dirección física 9322? Razónelo.
Dirección física = NºMarco*Tamaño página + Desplazamiento
9322 = num marco * 1024 + 405 → num marco*1024 = 8917
Como el número no es par, no es posible que la dirección lógica se traduzca en la dirección física 9322.

19. Suponga que tenemos 3 procesos ejecutándose concurrentemente en un determinado instante. El sistema operativo utiliza un sistema de memoria con paginación. Se dispone de una memoria física de 131072 bytes (128K). Sabemos que nuestros procesos al ser ejecutados tienen los parámetros que se muestran en la tabla.

Proceso	código	pila	datos
A	20480	14288	10240
B	16384	8200	8192
C	18432	13288	9216

Los datos indican el tamaño en bytes de cada uno de los segmentos que forman parte de la imagen del proceso. Sabiendo que una página no puede contener partes de dos segmentos diferentes (pila, código o datos), hemos de determinar el tamaño de página que debería utilizar nuestro sistema y se barajan dos opciones: páginas de 4096 bytes (4K) o páginas de 512 bytes (1/2K). Se pide:
a) ¿Cuál sería la opción más apropiada, 4096 bytes o 512 bytes? Justifica totalmente la respuesta mostrando todos los cálculos que has necesitado para llegar a dicha conclusión.

1) 4096 bytes.

Dividimos cada uno de los segmentos necesarios entre 4096 y nos quedamos con el resultado entero inmediatamente mayor:

20480/4096=5 14288/4096=4 10240/4096=3 ---> 12
16384/4096=4 8200/4096=3 8192/4096=2 —-> 9
18432/4096=5 13288/4096=4 9216/4096=3 —--> 12

33

33 páginas * 4096 bytes = 132 Kbytes > Memoria física

2) 512 Bytes:

20480/512=40 14288/512=28 10240/512=20 ---> 88
16384/512=32 8200/512=17 8192/512=16 —-> 65
18432/512=36 13288/512=26 9216/512=18 —--> 80

233

233 páginas * 512 bytes= 116,5 Kbytes < Memoria física

Por tanto, es más adecuado un tamaño de página de 512 bytes.

b) ¿Cuál es el formato de cada entrada de la Tabla de Páginas con el tamaño de página elegido? Justifica el tamaño de los campos con direcciones. Puedes añadir los bits que consideres necesarios para el buen funcionamiento del sistema indicando para qué van a ser utilizados.

Como 131072 Bytes /(512 Bytes/Página) = 256 Páginas = 2^8, habrá 8 bits para identificar el número de página. También habrá 1 bit de protección más 2 para el tipo de página. En total, 11.

c) ¿Cuántas Tablas de Páginas habrá en este sistema?¿Cuántas entradas hay en cada tabla de páginas (filas)?

Para cada proceso debe existir una tabla de páginas distinta, por lo que habrá 3. Cada una tendrá tantas entradas como páginas tenga el proceso. En la del proceso A habrá 88 entradas, en la del B 65 y en la del C 80.

20. En la gestión de memoria en un sistema paginado, ¿qué estructura/s de datos necesitará mantener el Sistema Operativo para administrar el espacio libre?

Se hará uso de una tabla de páginas que permita la traducción de direcciones de memoria virtuales a direcciones de memoria físicas, una para cada proceso. Es el mapa de memoria de un proceso que guarda el tamaño total del espacio de direcciones lógicas y la correspondencia entre las direcciones lógicas y las físicas.

Lista enlazada conteniendo los números de páginas que están libres.

21. Estamos trabajando con un sistema operativo que emplea una gestión de memoria paginada. Cada página tiene un tamaño de 2.048 bytes. La memoria física disponible para los procesos es de 8 MBytes. Suponga que primero llega un proceso que necesita 31.566 posiciones de memoria (o bytes) y, después, llega otro proceso que consume 18.432 posiciones cuando se carga en memoria. Se pide calcular la fragmentación interna provocada en cada proceso.

Memoria física = 8MBytes = 8 388 608 bytes

PROCESO 1: sí cabe en memoria física

$\frac{31\,566}{2048} = 15{,}41\ páginas$ → 16 páginas - 15,41 = 0,5869 páginas que quedarían libres

Por tanto, sí se produce fragmentación interna porque se necesitan 16 páginas que no se ocupan completamente

PROCESO 2: sí cabe en memoria física

$\frac{18\,432}{2048} = 9\ páginas$ → 0 páginas libres

Por tanto, no se produce fragmentación interna porque las 9 páginas que se necesitan se ocupan completamente.

22. Considere la siguiente tabla de segmentos:

Segmento	**dirección base**	**longitud**
0	**219**	**600**
1	**2300**	**14**
2	**90**	**100**
3	**1327**	**580**
4	**1952**	**96**

¿Qué direcciones físicas corresponden a las direcciones lógicas (nº_segmento, desplazamiento) siguientes? Si no puede traducir alguna dirección lógica a física, explique el por qué.
a) 0, 430
b) 1, 10
c) 3, 400
d) 4, 112

a) Segmento 0 → 430 < 600 → 219 + 430 = 649

b) Segmento 1 → 10 < 14 → 2300 + 10 = 2310

c) Segmento 3 → 400 < 580 → 400 + 1327 = 1727

d) Segmento 4 → ¡¡ 112 > 96 !! → no se puede traducir la dirección lógica porque con un desplazamiento de 112 nos salimos del segmento 4. Salta una excepción.

23. ¿Qué cambio de contexto tardará menos y por qué?
a) El producido entre dos hebras del mismo proceso.
b) El producido entre dos hebras de distintos procesos.

El producido entre dos hebras del mismo proceso, ya que al no tener que cambiar el contexto del proceso las hebras comparten datos y espacios de direcciones, lo que lo hace mucho más rápido que un cambio de contexto entre procesos diferentes. Es decir, los datos que van a ser utilizados ya han sido cargados en memoria para poder ejecutar estas subtareas, mientras que si hubiera que ejecutar otro proceso habría que reservar más espacio en memoria para cargar nuevos datos.