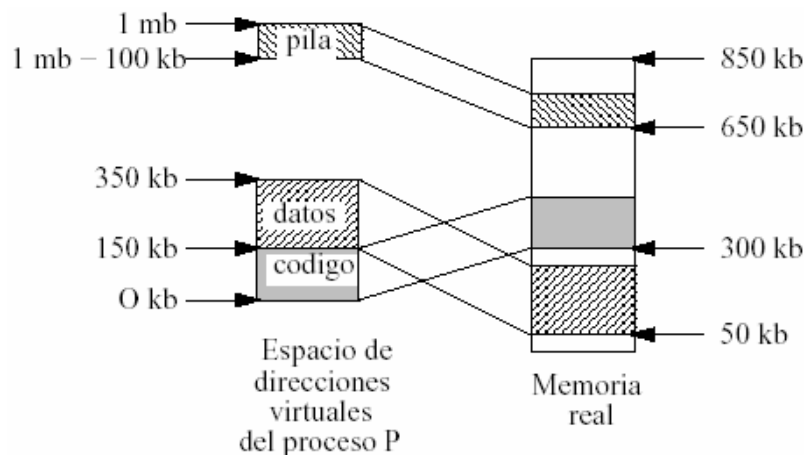


Auxiliar 6 – CC41B
Prof.: Luis Mateu Aux.: Juan Manuel Barrios
23 de octubre de 2007

Año 2001, Control 2, Pregunta 2



La figura muestra a la izquierda el espacio de direcciones virtuales de un proceso Unix que ocupa 450 KB de memoria. El proceso corre en un computador de 1 MB con arquitectura segmentada. A la derecha se muestra la ubicación actual de los segmentos en la memoria real. Se dispone en total de 850 KB para segmentos (el resto se ocupa en estructuras de datos del núcleo).

a.- Suponga que el proceso invoca a *sbrk* para hacer crecer su área de datos de 200 KB a 400 KB. Modifique la figura para mostrar una posible asignación de memoria después de la llamada a *sbrk*. Muestre en su figura las direcciones que ocupan ahora los segmentos en el espacio de direcciones virtuales y la memoria real.

b.- Haga la tabla de segmentos del proceso después de invocar a *sbrk*. Indique base virtual, límite virtual, desplazamiento y atributos para cada segmento.

c.- Suponga que ahora el proceso invoca a *fork*. Explique como logra el núcleo del sistema operativo satisfacer este nuevo requerimiento a pesar que no hay memoria disponible para todo los segmentos del nuevo proceso.

Solución

La tabla de segmentos que representa la figura es la siguiente:

Nombre	Inicio Virtual	Fin virtual	Desplazamiento	
Código	0 k	150 k	300 k	R
Datos	150 k	350 k	-100 k	RW
Pila	924 k	1024 k	-274 k	RW

a.- Como no hay espacio para hacer crecer el segmento de datos, será necesario hacer compactación en la memoria real para reunir el espacio. Por ejemplo, se podría mover el segmento de código al principio de la memoria real, luego dejar la pila y luego el segmento datos, o también podría moverse sólo el segmento de código para pegarlo con el segmento pila, etc.

b.- Suponiendo que elegimos mover sólo el segmento de código junto con la pila, la tabla de segmentos luego del *sbrk* sería la siguiente:

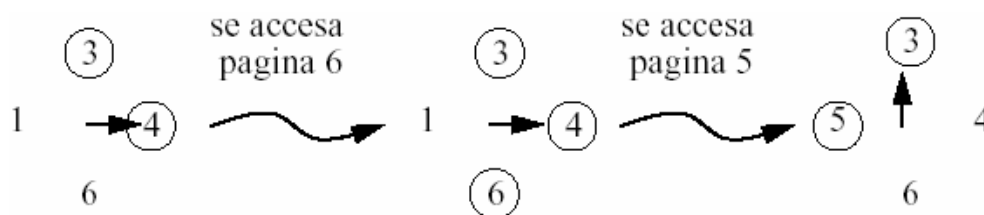
Nombre	Inicio Virtual	Fin virtual	Desplazamiento	
Código	0 k	150 k	500 k	R
Datos	150 k	550 k	-100 k	RW
Pila	924 k	1024 k	-274 k	RW

c.- Si el *fork* fuese antes del *sbrk* alcanzaría el espacio en memoria para tener los dos procesos en memoria si es que comparten el segmento de código. Si el *fork* se realiza después del *sbrk* entonces sería necesario hacer swap a disco de uno de ellos.

Año 2002, Examen, Pregunta 1

Parte a:

La siguiente figura muestra los estados sucesivos de la memoria al realizar 2 accesos en un sistema de memoria virtual que usa la estrategia del reloj para el reemplazo de páginas.



Las páginas que tienen el bit de referencia en 1, aparecen encerradas en una circunferencia. La posición del puntero está señalada por la flecha. En la figura se observa una primera transición de estados cuando se accesa la página 6 (residente) y una segunda transición al accesar la página 5 (no residente). A continuación se accesan las siguientes páginas de memoria:

4 7 5 3 4 1 5

Siguiendo el mismo esquema de la figura, muestre los estados de la memoria después de realizar cada uno de los accesos indicados.

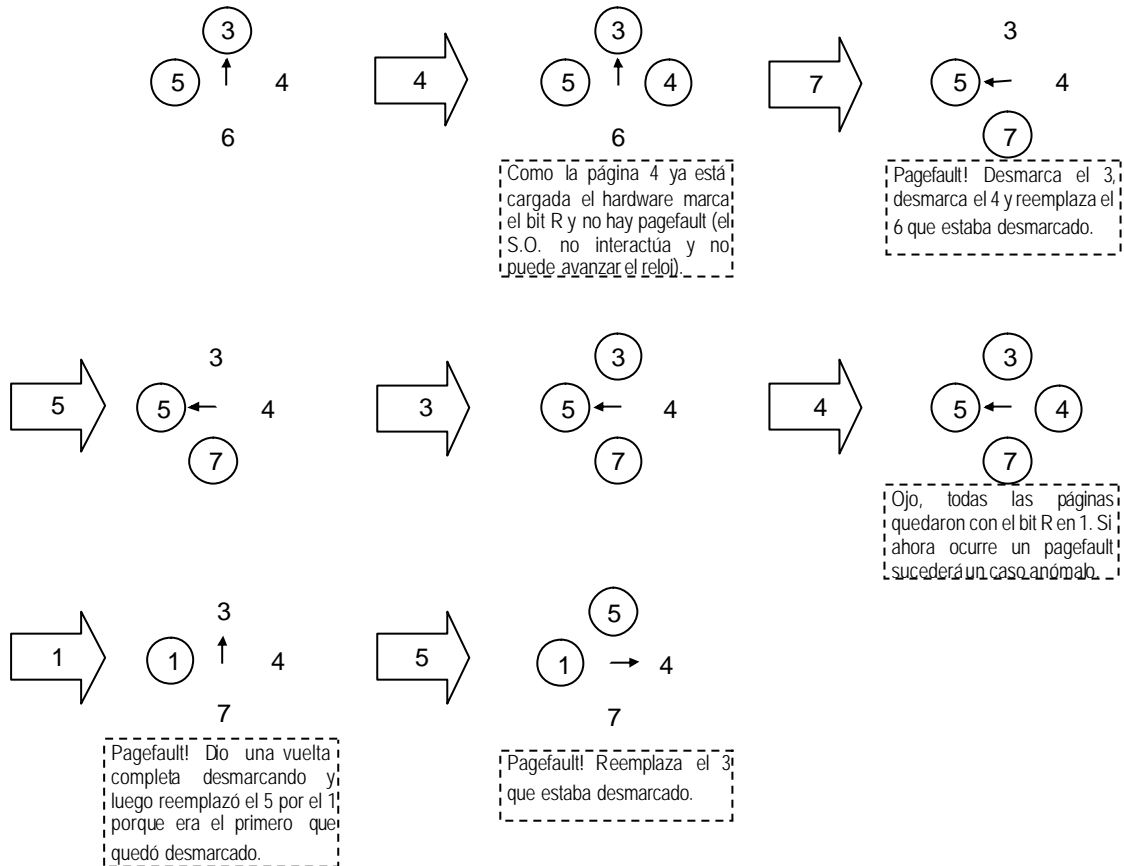
Parte b:

En un computador se dispone de 10 páginas reales para los procesos y 4 para el núcleo del sistema operativo. El sistema operativo utiliza la estrategia del *working set* para administrar las páginas. En él se ejecutan 3 procesos simultáneamente. El proceso A ocupa 5 páginas virtuales y su *working set* es el conjunto de páginas {0, 2, 3, 4}, el proceso B ocupa 6 páginas y su *w-s* es {1, 2, 4, 5} y el proceso C ocupa 4 páginas y su *w-s* es {0, 1, 3}.

Haga un diagrama mostrando una posible asignación de la memoria y escriba el contenido de las tablas de páginas de c/u de los procesos. En las tablas Ud. debe indicar si la página virtual se encuentra residente o no y el número de página real en que se encuentra.

Solución

a.-

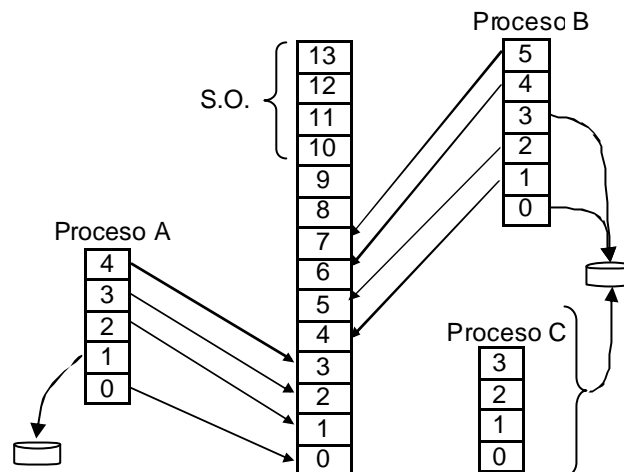


b.-

Para que un proceso esté en ejecución debe estar todo su working set en memoria, si no hay espacio para éste entonces al proceso completo se le hace swap a disco.

En el problema del enunciado hay 10 páginas disponibles en memoria real, pero para poder contener los w-s de los tres procesos se requiere de al menos 11 páginas. Por tanto no se pueden ejecutar los tres procesos simultáneamente y hay que elegir uno para enviarlo a disco (normalmente sería el que tenga menor actividad).

Supongamos que es el proceso C el que fue elegido para ser enviado a disco, el diagrama con la asignación de memoria sería el siguiente:



Las tablas de páginas para los procesos según el diagrama anterior son las siguientes:

Tabla Proceso A:

	Pág. real	v
4	3	1
3	2	1
2	1	1
1	(disco)	0
0	0	1

Tabla Proceso B:

	Pág. real	v
5	7	1
4	6	1
3	(disco)	0
2	5	1
1	4	1
0	(disco)	0

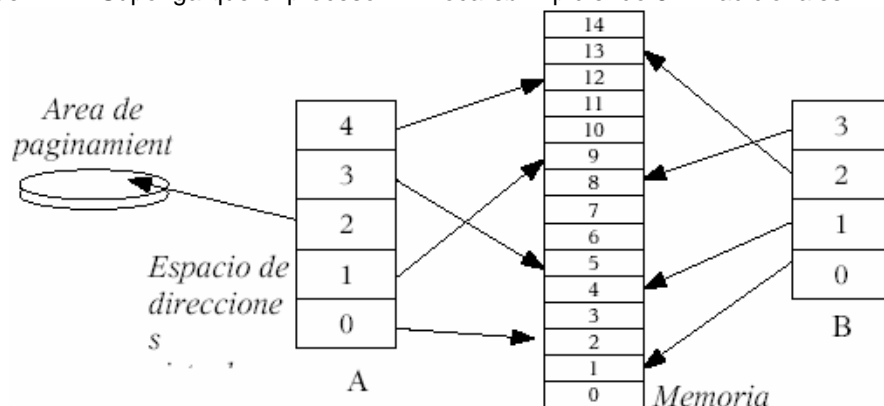
Tabla Proceso C:

	Pág. real	v
3	(disco)	0
2	(disco)	0
1	(disco)	0
0	(disco)	0

Pregunta 2, Control 2, 2004

i.- Explique por qué en un núcleo de sistema operativo *multi-threaded* (i.e. para multiprocesadores) la única herramienta de sincronización que puede ser usada para manipular la cola de procesos *ready* es el *spin-lock*. ¿Por qué no es posible usar un semáforo o un lock normal?

ii.- El diagrama muestra la asignación de páginas en un sistema Unix que ejecuta los procesos A y B. Las páginas son de 4 KB. Suponga que el proceso A invoca *sbrk* pidiendo 8 KB adicionales.



- Haga un nuevo diagrama para la asignación de páginas justo después de invocar *sbrk*.
- Incluya en su diagrama la tabla de páginas para el proceso A indicando los atributos de cada página.

iii.- Considerando el mismo diagrama del punto anterior (sin considerar *sbrk*), suponga que el proceso B invoca *fork* y que el núcleo utiliza la estrategia *copy-on-write* para implementar *fork*:

- Haga un nuevo diagrama para la asignación de páginas después de invocar *fork* y luego de que el proceso B modificó la página 2.
- Incluya en su diagrama la tabla de páginas para el padre y el hijo indicando los atributos de cada página.

iv.- El siguiente diagrama muestra (con una x) los accesos a la memoria para un proceso que se ejecuta en un sistema Unix que utiliza la estrategia del working set.

6	x	x	x	x		x	
5	x			x			
4	x	x			x	x	x
3	x	x		x	x	x	
2	x		x		x	x	
1					x		
0							
	A	B	C	D	E	F	G

Las letras A, B, C, etc. denotan los intervalos en los que se calcula el working set. Los números 0, 1, 2, etc. denotan las páginas del proceso.

- Indique para los períodos C a F qué accesos pueden producir pagefaults (utilice coordenadas del estilo (A, 4)).
- Indique el valor del atributo Referencia para todas las páginas al inicio del intervalo E y al final de ese intervalo.
- Considere que el computador en donde se ejecuta este programa posee solo 5 páginas reales. Explique en qué momento la estrategia del working set no funcionará adecuadamente.

Solución

i.- Debido a que dentro del núcleo pueden existir múltiples threads no se puede implementar una zona de exclusión a través de variables compartidas o inhibiendo las interrupciones en un procesador, si no que debe ser a través de un spin-lock el cual es un busy-wait que utiliza una instrucción de hardware especial (como test-and-set o compare-and-swap).

Para no degradar el sistema las secciones críticas deben contener una cantidad reducida de instrucciones (como por ejemplo el manejo de la cola *ready*). Para el caso de necesitar secciones críticas de mayor tiempo es necesario usar los spin-locks para diseñar otras herramientas de sincronización para esperas más largas que involucren listas de espera como locks o semáforos.

ii.- Las tablas de páginas para los procesos en el diagrama son las siguientes:

Tabla Proceso A:

	Pág. real	v	w
4	12	1	1
3	4	1	1
2	en disco	0	0
1	9	1	1
0	2	1	1

Tabla Proceso B:

	Pág. real	v	w
3	8	1	1
2	3	1	1
1	4	1	1
0	1	1	1

El proceso A invoca un *sbrk* de 8 KB => 2 nuevas páginas en la tabla del proceso A:

	Pág. real	v	w	
6	10	1	1	← páginas nuevas
5	6	1	1	
4	12	1	1	
3	4	1	1	
2	en disco	0	0	
1	9	1	1	
0	2	1	1	

iii.- El proceso hijo referencia las páginas del proceso padre quedando todas las páginas como sólo lectura. Cuando se intenta modificar una página primero se duplica luego las páginas nuevas quedan como lectura y escritura y finalmente la página es modificada sin afectar el otro proceso.

Tabla de páginas del Proceso Padre:

	Pág. real	v	w	
3	8	1	0	← Página modificada
2	3	1	1	
1	4	1	0	
0	1	1	0	

Tabla de páginas del Proceso Hijo:

	Pág. real	v	w	
3	8	1	0	← Página duplicada
2	13	1	1	
1	4	1	0	
0	1	1	0	

iv.-

- Ocurred pagefaults en las siguientes coordenadas: (C,2) (D,3) (D,5) (E,1) (E,2) (E,4) (F,6).
- Al principio del período E todas las páginas tienen el bit R en 0. Al final del período E quedan con el bit R en 1 las páginas 1, 2, 3 y 4.
- En el intervalo E el working set definido es {3, 5 6} pero se deberán cargar además las páginas {1, 2, 4}. Cuando se intente cargar la sexta página ocurrirá un pagefault pero todas las páginas disponibles están ocupadas, por lo que el núcleo forzosamente deberá elegir una página que está en el working set para ser reemplazada.