Administración de Memoria – Cap. 7 y 8

La memoria son localidades de almacenamiento con una dirección para identificarla.

Tareas del OS en cuanto a administración de memoria

1. Dividir la memoria para alojar los procesos necesarios. El OS debe determinar la cantidad de bloques o particiones administrados. Pueden haber tres escenarios:
    a. Una partición – un solo proceso autoejecutable sin OS.
    b. Dos particiones – un proceso + OS. Es el caso de OS monoproceso.
    c. Múltiples particiones – una partición para el OS y varias para procesos. Es el caso de OS multitarea.

   Los mecanismos de asignación para control de memoria son: particionamiento, paginación, segmentación y memoria virtual.
2. Ubicar el proceso en memoria (asignarle espacio), es decir decidir dónde cargar el nuevo proceso. Algunos algoritmos son:
    a. Best fit: elige la zona libre más pequeña donde quepa la imagen del proceso. Objetivo – elegir el bloque que desperdicie menos. Requiere mucho tiempo de comprobación y ordenamiento.
    b. First fit: busca secuencialmente el primer bloque libre con tamaño suficiente para el proceso. Es el más eficiente.
    c. Next fit: busca el bloque a partir de la última asignación realizada.
3. Determinar cuando una página será llevada a memoria principal. Éstas son las técnicas de carga:
    a. Por demanda
    b. Prepaginación
4. Determinar qué página será llevada a memoria secundaria cuando una nueva página debe ser cargada a memoria principal. Éstos son los algoritmos de reemplazo:
    a. Óptimo
    b. LRU
    c. FIFO
    d. Clock
    e. Page buffering

Técnicas de asignación de memoria

1. Partición fija – obsoleta
2. Partición dinámica – obsoleta
3. Paginación simple – no se usa en solitario
4. Segmentación simple – no se usa en solitario
5. Memoria virtual paginada
6. Memoria virtual segmentada

Partición fija

Permitía dos formas de división de la memoria, a saber:

1. La memoria se divide en segmentos (particiones) fijas de igual tamaño. Cualquier proceso puede ser asignado a una partición aunque no use todo el espacio. Puede

que el proceso no quepa en la partición. Hay desperdicio o fragmentación interna (el programa o proceso es mucho más pequeño que el bloque; sobra espacio dentro de los bloques). Fig. 1

2. La memoria se divide en particiones fijas de distinto tamaño. La asignación de los espacios se puede realizar de dos formas:
    a. Hay colas por cada tamaño de manera que los procesos sean asignados de acuerdo con su tamaño. Puede no haber una partición libre de un tamaño X, pero sí de otros tamaños. Fig. 2ª
    b. Hay una sola cola de manera que los procesos sean asignados a un espacio adecuado o al que esté libre para evitar esperas. Fig. 2b

Suponiendo una RAM total de 512 MB

1	64 KB	64000*8 = 512 MB
2	64 KB	
3	64 KB	
4	64 KB	
5	64 KB	
6	64 KB	
7	64 KB	
8	64 KB	

Fig. 1

Suponiendo una RAM total de 512 MB

Colas X partición

1	25 KB	Total = 512 MB
2	34 KB	
3	40 KB	
4	50 KB	
5	75 KB	
6	60 KB	
7	100 KB	
8	128KB	

Fig. 2ª

Una sola cola

1	25 KB	Total = 512 MB
2	34 KB	
3	40 KB	
4	50 KB	
5	75 KB	
6	60 KB	
7	100 KB	
8	128KB	

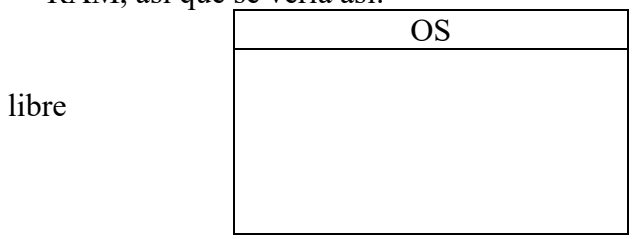
Fig. 2b

Partición dinámica

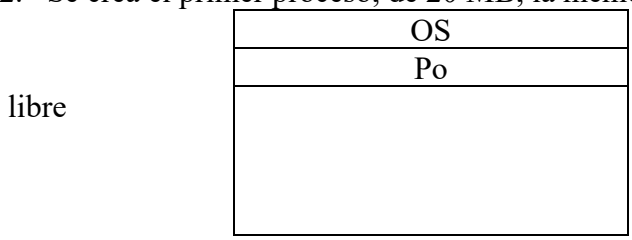
- Las particiones se crean de acuerdo con el tamaño del proceso nuevo.
- Problema: se crean muchos huecos entre bloques o fragmentación externa.
- A través de compactación (desplazamiento de procesos que estén contiguos de forma que queden bloques libres más grandes) se resuelve este problema, pero esta operación consume tiempo; desperdicia tiempo del CPU.

Ejemplo con una memoria principal de 64 MB:

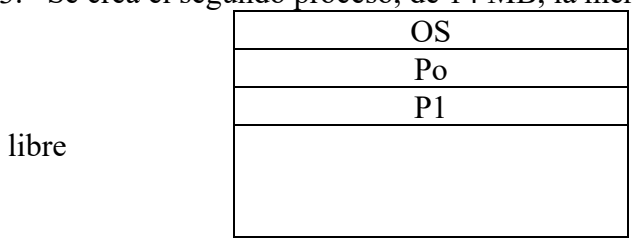
1. En momento recién iniciado el sistema, sólo el OS está en la memoria RAM, así que se vería así:



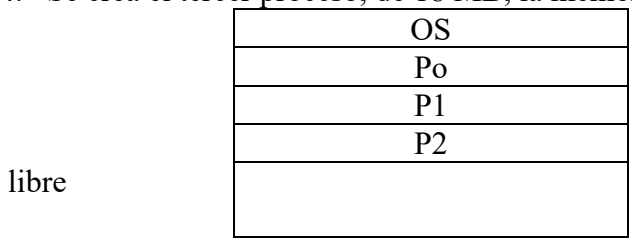
2. Se crea el primer proceso, de 20 MB, la memoria se vería así:



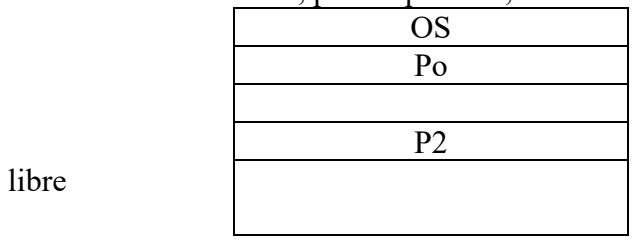
3. Se crea el segundo proceso, de 14 MB, la memoria se vería así:



4. Se crea el tercer proceso, de 18 MB, la memoria se vería así:



5. El proceso 1 se termina o suspende; quedando un segmento de de 14 MB y otro de 4 MB libres, pero separados; la memoria se vería así:



6. Se crea el proceso 3, de 8 MB; quedando un segmento de de 6 MB y otro de 4 MB libres, pero separados, así que sólo procesos de esos tamaños se podrían crear; la memoria se vería así:

	OS	8 MB ocupado
	Po	20 MB ocupado
	P3	8 MB ocupado
		6 MB libre
	P2	18 MB ocupado
libre		4 MB libre

7. P0 termina; quedando tres segmentos libres – 20 MB, 6 MB y 4 MB separados, la memoria se vería así:

	OS	8 MB ocupado
		20 MB libre
	P3	8 MB ocupado
		6 MB libre
	P2	18 MB ocupado
libre		4 MB libre

8. El P1 de 14 MB regresa a la memoria principal; quedando tres segmentos libres – 20 MB, 6 MB y 4 MB separados, la memoria se vería así:

	OS	8 MB ocupado
	P1	14 MB ocupado
		6 MB libre
	P3	8 MB ocupado
		6 MB libre
	P2	18 MB ocupado
libre		4 MB libre

Se requiere compactación para hacerlos más usables; 16 MB.

Ubicación / Reubicación

Como los procesos pueden ser cargados en espacios distintos de memoria cada vez, las ubicaciones de datos e instrucciones cambian, por tanto se requiere un proceso de traducción de las direcciones de los procesos. Hay tres referencias a la dirección de un proceso en memoria:

1. Dirección física – absoluta, es la dirección real en memoria.
2. Dirección lógica – es una referencia a una posición de memoria independiente de la asignación actual de datos a memoria. Requiere traducción a una dirección física.
3. Dirección relativa – es una dirección lógica que se da respecto a un punto conocido, normalmente con respecto al inicio del programa.

Los programas o procesos se cargan en memoria a través de un cargador dinámico, o sea que las direcciones se calcularán con respecto al origen del programa en el área de memoria en el que quedó ubicado, usando un mecanismo de hardware. Proceso:

1. Se le asigna la dirección al registro base (inicio) y al registro límite (posición final del programa) en el momento que el programa se carga a la memoria.
2. Todas las demás direcciones de referencia (registros de instrucción, bifurcación, datos, etc.) se deben verificar antes de ejecutar la instrucción para asegurarse que están en el área correcta. Sino se envía un trap.
    a. BP + Relative address = Absolute address
    b. BP + Relative Address < Stack Pointer
    c. Si BP + Relative Address > Stack Pointer ➔ se envía un trap, sino se le da la dirección absoluta para continuar con la ejecución de la instrucción. Ver Fig. 4

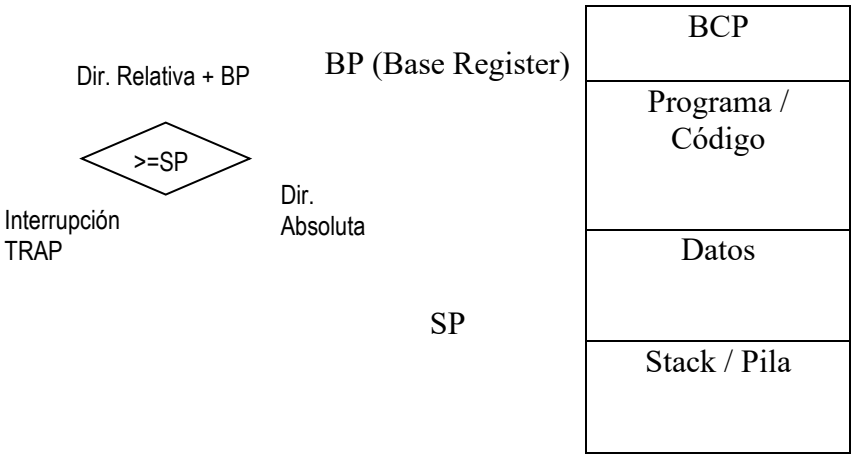


Fig. 4 – Imagen del Proceso

Técnicas de carga

Por demanda: consiste en iniciar la ejecución del proceso sin páginas cargadas; éstas se irán cargando conforme el proceso haga referencia a una posición en dicha página. La tasa de fallos de página es alta al iniciar la ejecución del proceso; luego se estabiliza.

Prepaginación

El OS predice las páginas que serán utilizadas durante la ejecución del proceso. Entnces trae páginas contiguas a la del fallo para evitar traer una a una, lo cual tomaría más tiempo. No es eficiente si las páginas extra no se referencian.

Políticas de reemplazo

1. Óptima: descarga la / las páginas que no se necesitarán en el tiempo más largo en el futuro. Imposible porque se basa en el futuro.
2. LRU (Least Recently Used): descarga la página menos usada recientemente. Por principio de cercanía, ésta tendrá menos probabilidad de ser referenciada en el futuro.

2	3	2	1	5	2	4	5	3	2	5	2
2	2	2	2	2	2	2	2	3	3	3	3
	3	3	3	5	5	5	5	5	5	5	5
			1	1	1	4	4	4	2	2	2
				F		F		F	F		

3. FIFO (First In First Out): reemplaza la página que ha estado más tiempo en memoria. produce errores porque supone que una página introducida hace mucho tiempo pudo haber caído en desuso, lo cual no siempre es cierto.

2	3	2	1	5	2	4	5	3	2	5	2
2	2	2	2	5	5	5	5	3	3	3	3
	3	3	3	3	2	2	2	2	2	5	5
			1	1	1	4	4	4	4	4	2
				F	F	F		F		F	F

4. Clock: una un bit de uso, que se pone en cero (0) la primera vez que se carga la página en memoria y en uno (1) cuando se haga referencia a esta página (cuando hay un fallo de página). Un puntero va pasándose/apuntando al siguiente marco. La primera página encontrada con el bit de uso = 1 se cambia a cero (0). Así se da toda la vuelta y la primera con el bit en cero se reemplaza.

2	3	2	1	5	2	4	5	3	2	5	2
2*	2*	2*	2*	5*	5*	5*	5*	3*	3*	3*	3*
	3*	3*	3*	3	2*	2*	2*	2	2*	2	2*
			1*	1	1	4	4*	4	4	5*	5*
				F	F	F		F		F	

*Bit u=1
Sombra = puntero

5. Page buffering (almacenamiento intermedio):
    a. Usa FIFO.
    b. La página reemplazada se mantiene físicamente en memoria principal, ya sea en la lista de páginas libres o en la de páginas modificadas.
    c. Si el proceso vuelve a hacer referencia a la página, demora menos el activarla y agregar su entrada en la tabla de páginas.
    d. Estas listas son "caché de páginas."