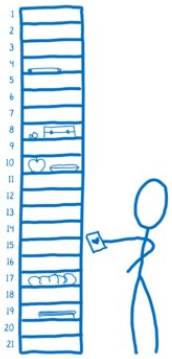
Funciones de los sistemas operativos

3. Gestión de la memoria

La parte del sistema operativo que administra la memoria es el administrador o gestor de memoria (memory manager). Su función es:

- Llevar registro de las partes de la memoria que se están utilizando y de las que no.
- Reservar memoria para los nuevos procesos y liberarla para los procesos que hayan finalizado.
- Protege las zonas ocupadas en memoria de otros procesos.
- Gestiona el intercambio de datos entre memoria y disco.



La memoria suele estar dividida en dos partes; la ocupada por los procesos del sistema operativo, normalmente ubicada al principio de las misma y la ocupada por los procesos de los ususuarios que normalmente se van ocupando desde el final hacia el principio de la misma.

Cuando un programa se ejecuta, la información del mismo que estaba almacenada en memoria secundaria (disco) es lllevada a la memoria principal o RAM.

La memoria RAM es un recurso limitado. Al ir poniendo procesos en ejecución llegará un momento en el que esta no tenga tamaño suficiente para almacenar los BCP de dichos procesos y habrá que recurrir al almacenamiento secundario. Para gestionar el intercambio de información entre la memoria y el disco se utilizan dos técnicas:

- Memoria virtual
- Intercambio (swapping)

Para realizar el intercambio de datos entre la memoria y el almacenamiento secundario cuando la memoria está llena se puden utilizar diferentes algorítmos a políticas.

También se pueden aplicar diferentes tipos de algoritmos para decidir en que hueco de la memoria de los existentes se ubican los datos de un nuevo programa que se ejecuta,

3.1. Algoritmos de sustitución

Cuando se ejecuta un proceso y no hay espacio en memoria principal para ubicarlo, el gestor de memoria ha de mover a almacenamiento secundario los datos de uno o varios procesos que están en memoria y no se están ejecutando para hacer sitio.

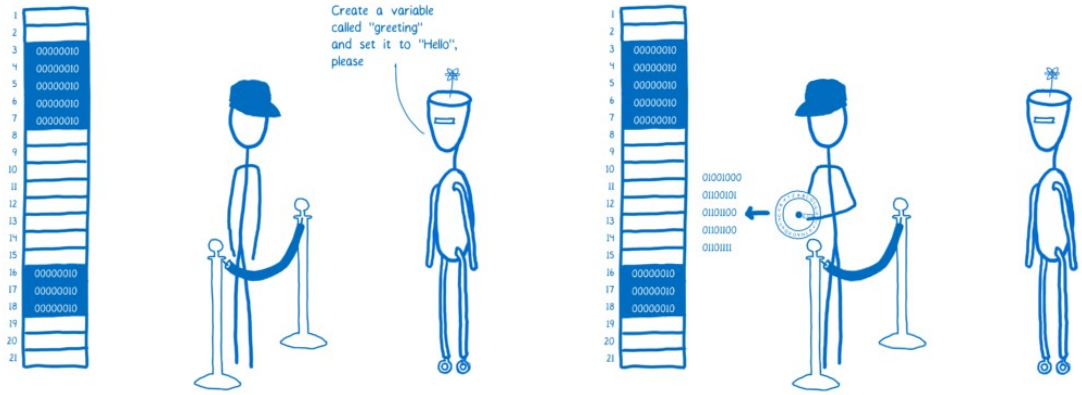
A la hora de realizar esta operación de intercambio entre alamacenamiento secundario y principal se pueden seguir diferentes políticas o algorítmos:

- **FIFO (First In First Out)** - Se selecciona el primer proceso que se almacenó en RAM, o lo que es lo mismo, el que lleva más tiempo en memoria.

- **LRU (Less Recentrly Used)** - La estrategia consiste en llevar a disco los datos que ha permanecido por más tiempo sin ser accedidos. El fundamento de esta estrategia es que estadísticamente se observa que mientras más tiempo permanece una información sin ser accedida, menos probable es que se acceda en el futuro inmediato.
- **LFU (Less Frecuentrly Used)** - En este caso se lleva a disco los datos que han sido accedidos con menos frecunecia. El gestor de memoria almacena un contador para cada dato que se incrementa cada vez que es accedido.
- **Óptimo** - Se llevan a disco los datos que menos probablemente vayan a ser usados. Es el ideal, pero es imposible de llevar a la práctica. Se utiliza en entornos simulados controlados en los que si se puede utilizar para compararlo con otros algoritmos y evaluarlos.

3.2. Algoritmos de reubicación

Cuando iniciamos el sistema los procesos se van ubicando en memoria de forma contigua. A medida que se van ejecutando y finalizando procesos se van generando huecos en memoria que pueden ser utilizados para ubicar nuevos procesos que se ejecuten.



Para ubicar el nuevo proceso en memoria se puede seguir alguna de las siguientes políticas a la hora de seleccionar el hueco en que ubicar el nuevo proceso:

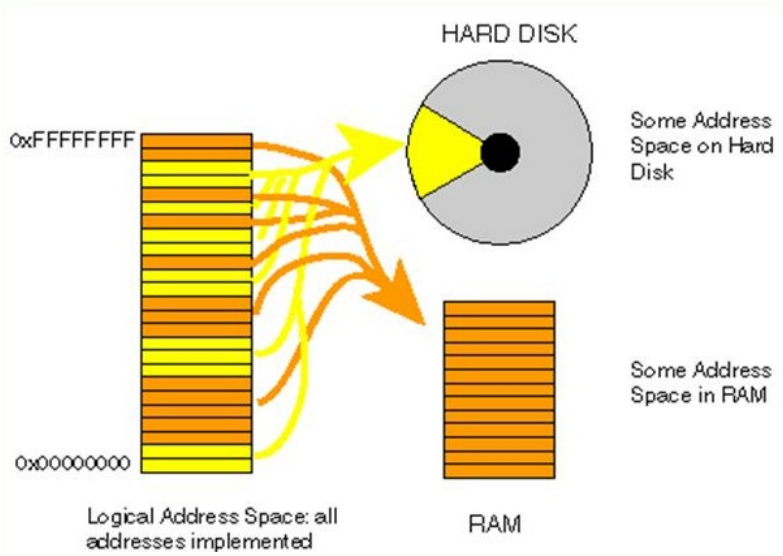
- **BEST FIT (Mejor ajuste)** - Se coloca donde vaya a sobrar menos espacio
- **FIRST FIT (Primer ajuste)** - Se coloca en el primer hueco que se encuentre
- **WORST FIT - (Peor ajuste)** - Se coloca donde mas espacio vaya a sobrar

3.3. Memoria virtual

Está técnica consiste en reservar un espacio en disco que será utilizado como extensión de la memoria RAM. Con ello conseguimos de forma **virtual** aumentar el tamaño físico del hardware de RAM de nuestro equipo.

Esa memoria virtual ha de ser gestionada de forma eficiente ya que el aceso a disco siempre va a ser bastante más lento que el acceso a RAM.

En un sistema de memoria virtual se mantiene en disco un archivo con la imagen del proceso completo. En cambio en la memoria únicamente deben estar los datos del proceso que en ese momento deba estar en ejecución, intercambiándose datos entre disco y memoria principal cuando sea necesario.



Cuando se utiliza memoria virtual el gestor de memoria virtual suele utilizar una de las siguientes técnicas:

- Gestión de memoria **segmentada**
- Gestión de memoria por **páginas**
- Gestión de memoria **segmentada-paginada**

3.3.1. Memoria virtual segmentada

Los procesos ocupan segmentos de tamaño variable según sea su tamaño y en caso de no caber se aplica alguna de las políticas de sustitución.

Ejemplo:

Tenemos un sistema con 1GB de RAM en el que inicialmente tenemos los siguientes procesos en memoria:

SSOO	**250MB**
Office	100MB
Skype	200MB
Spotify	150MB
Buscaminas	100MB
Libre	**200MB**

El sistema operativo ocupa los 250 primeros MB de la RAM y el resto lo ocupan diferentes procesos. Hay 200MB libres

Si ejecutamos **Gimp** que ocupa **300 MB** en memoria, como sólo tenemos **200MB libre** hemos de quitar un proceso de memoria. Si utilizamos como política de sustitución FIFO, por ejemplo, como el que lleva más tiempo es el Office y no está en ejecución lo seleccionamos

SSOO	250MB
Office pasa a disco y deja espacio libre	**100MB**
Skype	200MB
Spotify	150MB
Buscaminas	100MB
Espacio libre	**200MB**

A continuación reubicamos los procesos para compactar la memoriay tener espacio continuo para el nuevo proceso:

SSOO	250MB
Skype	200MB
Spotify	150MB
Buscaminas	100MB
Libre	300MB

Ya podemos ubicar Gimp en memoria:

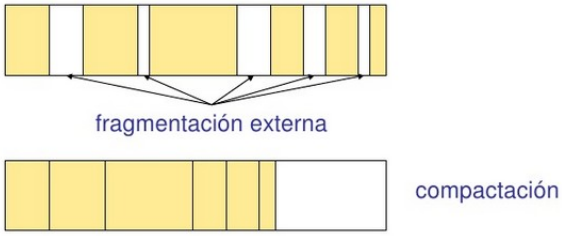
SSOO	250MB
Gimp	300MB
Skype	200MB
Spotify	150MB
Buscaminas	100MB
Libre	0MB

3.3.1.1. Fragmentación externa

Espacios libres en memoria que quedan entre los procesos. A medida que se van reubicando procesos en los huecos de memoria van quedando una serie de pequeños espacios que no pueden ser aprovechados por ningún proceso.

El espacio total ocupado por dichos huecos en memoria no contigua se llama fragmentación externa y es uno de los inconvenientes de la memoria virtual segmentada. Ya que acabamos teniendo una cantidad de memoria ocupada mayor que la suma de la memoria ocupada por todos los procesos en memoria.

Para solventar este problema se debe compactar periódicamente la ubicación de los procesos en memoria.



3.3.1.2. Ejercicio resuelto

Tenemos un sistema que gestiona la memoria por **segmentación**. La memoria tiene 4200 KB. En un momento determinado la memoria está ocupada por 3 procesos de la forma:

Dirección inicial	longitud	Proceso
1000	1000	P1
2900	500	P2
3400	800	P3

La estrategia de reemplazo cuando se carga en memoria un proceso es la del mejor ajueste (**best fit**). Si falla (no existe un hueco en memoria) se crea un hueco desplazando los bloques en memoria **hacia la dirección 0**. Esta acción siempre empieza con el bloque actualmente en la dirección de memoria más baja, y prosigue hasta encontrar un hueco del suficiente tamaño para el nuevo bloque. Si no hubiera hueco suficiente se utiliza la estrategia de reemplazo FIFO.

Si hay que cargar tres procesos P4, P5 y P6 que ocupan **500** , **1200** y **100KB** (en ese orden). Describir el contenido final de la memoria. ¿Qué fragmentación externa tenemos al final de la ejecución?

Memoria inicial

Dirección inicial	0000	1000	2000	2900	3400
Libre/ocupado	1000	P1 1000	900	P2 500	P3 800

Se carga proceso de **500KB**. Pasos:

1. En dirección 2000 hueco que menos espacio libre queda. Se inserta

Dirección inicial	0000	1000	2000	2500	2900	3400
Libre/ocupado	1000	P1 1000	P4 500	400	P2 500	P3 800

Se carga proceso de **1200KB**

1. No hay hueco libre suficiente. Se desplaza proceso en dirección 1000 a dirección 0

Dirección inicial	0000	1000	2000	2500	2900	3400
Libre/ocupado	P1 1000	1000	P4 500	400	P2 500	P3 800

2. Sigue sin haber hueco. Se rueda proceso en dirección 1900 a dirección 1000

Dirección inicial	0000	1000	1500	2900	3400
Libre/ocupado	P1 1000	P4 500	1400	P2 500	P3 800

3. Ya tenemos hueco, insertamos proceso en dirección 1500

Dirección inicial	0000	1000	1500	2700	2900	3400
Libre/ocupado	P1 1000	P4 500	P5 1200	200	P2 500	P3 800

Se carga proceso de **100KB**

1. Se carga en el principio del hueco que queda de 200KB

Dirección inicial	0000	1000	1500	2700	2800	2900	3400
Libre/ocupado	P1 1000	P4 500	P5 1400	P 6 1 0 0	1 0 0	P2 500	P3 800

Al terminar tenemos la memoria totalmente ocupada con el siguiente contenido final:

Dirección inicial	0000	1000	1500	2700	2800	2900	3400
Libre/ocupado	P1 1000	P4 500	P5 1400	P 6 1 0 0	1 0 0	P2 500	P3 800

La fragmentación externa es de 100KB

Actividad. Para el mismo supuesto. ¿Qué resultado final obtendríamos si hay que cargar tres procesos P4, P5 y P6 que ocupan **500** , **200** y **1300KB** (en ese orden) y utilizamos como estrategia de reemplazo la del peor ajuste (**worst fit**)?

3.3.2. Memoria virtual paginada

Se divide la memoria en porciones llamadas marcos de página de tamaño fijo. En función de las necesidades, el gestor de memoria se encarga de e intercambiar la información de páginas de memoria entre la memoria principal y el disco duro.

Ejemplo:
Tenemos la memoria del ejemplo anterior de 1GB paginada con la siguiente estructura:

SO	50MB
SO	50MB
SO	50MB
SO	50MB
SO	50MB
OFFICE	50MB
OFFICE	50MB
SKYPE	50MB
SKYPE	50MB
SKYPE	50MB
SKYPE	50MB
SPOTIFY	50MB
SPOTIFY	50MB
SPOTIFY	50MB
BUSCAMINAS	50MB
BUSCAMINAS	50MB

Ejecutamos Gimp que ocupa 300 MB (6 marcos de 50MB). Faltan 2 marcos para poder ubicarlo en memoria. Se ubican 4 marcos en memoria y los 2 restantes en disco.

SO	50MB
SO	50MB
SO	50MB
SO	50MB
SO	50MB
OFFICE	50MB
OFFICE	50MB
SKYPE	50MB
SKYPE	50MB
SKYPE	50MB
SKYPE	50MB
SPOTIFY	50MB
SPOTIFY	50MB
SPOTIFY	50MB
BUSCAMINAS	50MB

BUSCAMINAS	50MB
GIMP	50MB
GIMP	50MB
GIMP	50MB
GIMP	50MB

Si el proceso necesita alguno de esos marcos que están en disco se produce un **fallo de página**. El fallo hace que se traiga de disco los marcos necesarios y los intercambia con otros mediante el algoritmo de sustitución que utilice el gestor de memoria.

3.3.2.1. Fragmentación interna

Trozos de marco de página que no se aprovechan

Una solución a la fragmentación interna es hacer los marcos de página lo más pequeños posibles

3.3.2.2. Ejercicios resueltos

a) Tenemos 3 procesos (P1,P2 y P3): P1 ocupa 1KB, P2 ocupa 4 KB y P3 ocupa 2 KB. El tamaño de las páginas es de 4 KB. ¿Cuál es la **fragmentación interna**?

Memoria	P1				P2	P2	P2	P2	P3	P3		
Marcos	0				1				2			

```
P1 → 3KB
P2 → 0KB
P3 → 2KB
---------
     5KB de fragmentación interna
```

b) Dibuja cómo quedaría el mapa de una memoria paginada con marcos de páginas de 4Kb si se tienen cuatro procesos, llamados A, B, C y D, que ocupan respectivamente 3, 2, 2 y 3 páginas después de ejecutar las siguientes acciones:

1. El programa A se carga en memoria (se le asignan los marcos 0, 1 y 2)
2. El programa B se carga en memoria (se le asignan los marcos 3 y 4)
3. El programa C se carga en memoria (se le asignan los marcos 5 y 6)
4. El programa B termina, liberando sus páginas
5. El programa D se carga en memoria (se le asignan los marcos 3 y 4 que usaba el proceso B y el marco 7 que permanecía libre)

1.

Procesos	A	A	A						
Marcos	0	1	2	3	4	5	6		

2.

Procesos	A	A	A	B	B				
Marcos	0	1	2	3	4	5	6	7	8

3.

Procesos	A	A	A	B	B	C	C		
Marcos	0	1	2	3	4	5	6	7	8

4.

Procesos	A	A	A			C	C		
Marcos	0	1	2	3	4	5	6	7	8

5.

Procesos	A	A	A	D	D	C	C	D	
Marcos	0	1	2	3	4	5	6	7	8

c) Supóngase que, después de haberse reservado espacio para el Sistema, sólo queda sitio suficiente en la memoria principal para cuatro páginas de los programas de usuario. Supóngase también que inicialmente se han situado en la memoria física las páginas virtuales 1, 2, 3 y 4 de un programa de usuario, en este orden.

1) Aplicando las políticas de sustitución de páginas: **LRU** (la página menos recientemente usada) y **FIFO** (la página que ha estado en memoria durante más tiempo). ¿qué páginas estarán en la memoria al final de la siguiente secuencia de accesos a páginas virtuales (6, 3, 2, 8, 4) ?

LRU

Marco	1,2,3,4	6	3	2	8	4
0	**1**	**6**	6	6	6	**4**
1	**2**	2	2	2	2	2
2	**3**	3	3	3	3	3
3	**4**	4	4	4	**8**	8

FIFO

Marco	1,2,3,4	6	3	2	8	4
0	**1**	**6**	6	6	6	6
1	**2**	2	2	2	**8**	8
2	**3**	3	3	3	3	3
3	**4**	4	4	4	4	4

2) ¿Qué estrategia de sustitución de las dos trabajará mejor cuando la máquina acceda a las páginas en el siguiente orden : (3, 4, 5, 6, 7, 6, 5, 4, 3, 4, 5, 6, 7, 6, ...)?

LRU

Marco	1,2,3,4	3	4	5	6	7	6	5	4	3	4	5	6	7	6
0	**1**	1	1	**5**	5	5	5	5	5	5	5	5	5	5	5
1	**2**	2	2	2	**6**	6	6	6	6	6	6	6	6	6	6
2	**3**	3	3	3	3	**7**	7	7	7	**3**	3	3	3	**7**	7
3	**4**	4	4	4	4	4	4	4	4	4	4	4	4	4	4

N.° fallos de página: 5

FIFO

Marco	1,2,3,4	3	4	5	6	7	6	5	4	3	4	5	6	7	6
0	**1**	1	1	**5**	5	5	5	5	5	5	**4**	4	4	4	4
1	**2**	2	2	2	**6**	6	6	6	6	6	6	**5**	5	5	5
2	**3**	3	3	3	3	**7**	7	7	7	7	7	7	**6**	6	6
3	**4**	4	4	4	4	4	4	4	4	**3**	3	3	3	**7**	7

N.º fallos de página: 7

Trabaja mejor la estrategia de sustitución LRU

Actividad. Para el mismo caso inicial de los apartados anteriores ¿Qué estrategia de sustitución trabajará mejor cuando la máquina acceda a las páginas en orden totalmente aleatorio, tal como: (3, 4, 2, 8, 7, 2, 4, 5, 6, 3, 4, 8, ...)?

3.3.3. Memoria virtual segmentada-paginada

Consiste en dividir la memoria en varios segmentos y a su vez cada segmento en páginas de tamaño fijo. Con esta técnica, un segmento está formado por un conjunto de páginas que no tiene que estar contiguas en memoria. Tiene las ventajas de ambos esquemas de gestión.

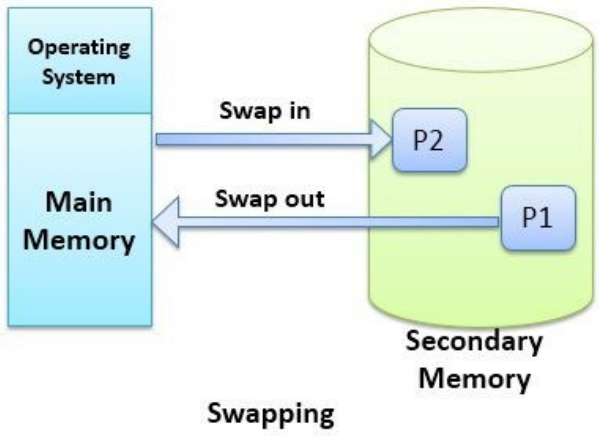
3.4. Intercambio o swapping

De forma similar a la técnica de memoria virtual se reserva un espación disco, normalmente llamado swap. A diferencia de la memoria virtual, dicho espacio no es gestionado como una extensión de la memoria principal, sino que es un espacio reservado en memoria secundaria, normalmente en una partición con un sistema de archivos propio.

Dicho espacio en disco es utilizado para el intercambio de todos los datos de los procesos dependiendo del estado de los mismos.

Si se interrumpe la ejecución de un proceso los datos del mismo pasarán a la zona swap, este procedimiento se llama **swap-out**. Con ello se consigue que la memoria principal libere parte de su espacio que podrá ser utilizado por otro proceso en ejecución.

Si el proceso vuelve a pasar al estado de ejecución se realiza un procedimiento de **swap-in** que consiste en pasar el programa de la zona swap a la memoria pricipal.



Swapping

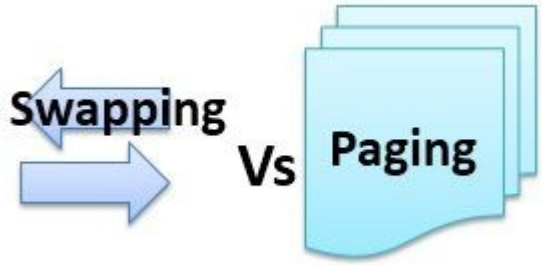
Para ver la información de ls RAM y swap así como el estado de las mismas podemos utilizar en Linux la utilidad free:

```
$ free -h
              total usado libre compartido búfer/caché disponible
Memoria:      7,2G  2,3G  4,0G  303M       906M        4,4G
Swap:         2,0G  345M  1,7G
```

Normalmente el swapping no lo activa el gestor de memoria en todo momento porque las operaciones de movimiento de información entre disco y RAM penalizan el rendimiento del sistema. Se puede configurar el porcentaje de memoria libre a partir del cual se empieza a aplicar la técnica de swap. En Linux la variable que almacena este porcentaje se llama swappiness y podemos comprobar su valor ejecutando:

```
$ cat /proc/sys/vm/swappiness
```

La técnica de intercambio se suele utilizar en sistemas operativos como Unix y Linux. Suele implementarse una **partición de disco** reservada para dicha función.



Se suele recomendar que el tamaño de la misma sea el doble de la capacidad RAM del ordenador, aunque en los equipos actuales que tienen memorias RAM de gran tamaño, no es necesario que la swap sea tan grande.

3.4. Comparativa entre swapping y memoria virtual

Una ventaja de la técnica de swapping frente a la técnica de gestión mediante memoria virtual es que, mediante memoria virtual puede llegar a ocurrir que el disco esté tan lleno que la gestión sea difícil o imposible, ya que el espacio destinado en disco para la gestión de memoria es compartido con los archivos del sistema, las aplicaciones y los datos de los usuarios. En el Swapping no puede ocurrir esto, ya que esta zona siempre estará disponible al ser un espacio reservado, normalmente en una partición dispositivo aparte y no puede ser utilizado para almacenar otro tipo de información.

Por el contrario la técnica de memoria virtual es más flexible que la de swap porque utilizando memoria virtual se pueden mover entre disco y memoria parte de la estructura de datos de un proceso, mientras que con swap se mueven los datos completos del proceso. Por tanto la memoria virtual permite alojar más procesos en la memoria principal que el swapping.

3.5. Credits

- Cartoons: https://hacks.mozilla.org/2017/06/a-crash-course-in-memory-management/
- https://techdifferences.com/difference-between-paging-and-swapping-in-os.html
- https://www.slideshare.net/sgpraju/os-swapping-paging-segmentation-and-virtual-memory