# Comparativa sobre el uso de la memoria en Windows y Linux

*Grupo 9*

# Contenidos



# 1 Autores

El grupo 9 lo componen:

Carlos Miguel Torres Jiménez

Elena Ballano Hernández

Enrique López Mañas

Ismael Salgado Flores

José Abel Tamayo

# 2 Objetivo del estudio

Desde finales de los 90, cuando los sistemas operativos alternativos a Windows con Linux a la cabeza lograron una mayor distribución y crédito han sido y siguen siendo numerosos los debates sobre cuál es más eficiente, estable, rápido, etc..

Uno de los factores clave para la calidad y velocidad de un sistema operativo está en el uso que hace de la memoria tanto virtual como física, que se intercambian información a través del proceso de swapping. El objetivo de este estudio es descubrir cuál de los dos sistemas es más rápido en el uso de su memoria, cuáles pueden ser algunas de las causas que expliquen las diferencias en la efectividad y qué repercusión tiene esto en los diferentes tipos de usuarios del sistema.

# 3 Descripción del problema

Al efectuar una comparación entre los dos sistemas operativos más famosos del mercado, deberíamos primero dar una pequeña introducción sobre ambos. Windows es un sistema operativo de código cerrado, o propietario. Cabría extenderse en la definición sobre que es código libre y propietario, pero no concierne a este documento. Bastará con decir que en el mundo del software, existen dos conceptos antagónicos: software libre y software propietario. El primero sigue las directrices GNU (ha de ser

distribuido con su código fuente, y libertad para modificarlo y distribuirlo siempre y cuando se respeten ciertas condiciones) y el segundo es software “cerrado”: no se ofrece ninguna información sobre los algoritmos o códigos fuente que componen el sistema. Para más información realizar una búsqueda en google sobre “software libre y propietario”.

Al evaluar el aspecto que nos incumbe, la gestión de memoria por parte de ambos sistemas operativos, han de tenerse en cuenta ciertos factores. Queremos medir diversas variables, tales como la velocidad de acceso a memoria, el comportamiento de los sistemas cuando la memoria escasea, la efectividad de la gestión de la misma, la idoneidad de los sistemas para según que usos… Para poder dar respuesta a estos interrogantes, ha de tenerse un conocimiento previo de cómo un ordenador gestiona la memoria, y también como lo hacen a nivel interno ambos sistemas operativos, para poder efectuar una comparación, y emitir un juicio lógico y razonado.

Previamente a abordar el proyecto, decidimos realizar un trabajo de documentación sobre el proyecto en cuestión. Recopilamos información técnica que consideramos nos podría resultar beneficiosa para comprender mejor el objeto de estudio. Tras asimilarla, actualizar nuestros conocimientos sobre la gestión de memoria, decidimos el mejor método para poder evaluar ambos sistemas operativos. La velocidad a nivel de hardware y la fiabilidad pueden ser considerados como los dos puntos clave para poder emitir un análisis de opinión sobre las características de ambos sistemas: es por ello que diseñamos una prueba como la que puede consultarse en el apartado correspondiente.

Existen diferentes políticas descritas en lo que se refiere a la gestión de la memoria: algunos sistemas como Windows se apoyan en las funciones que les ofrecen los procesadores para controlar las direcciones de memoria mientras que Linux utiliza sus propios algoritmos. También la manera en que se reemplaza la información antigua por otra nueva puede responder a diferentes políticas que optimicen la velocidad.

En el caso de Linux todos los detalles de su implementación son conocidos incluídas dichas políticas, pero Microsoft mantiene el secreto de muchos de sus algoritmos de gestión. Utilizando un monitor para diversos factores de la CPU, pudimos llevar a buen término las pruebas propuestas para realizar la evaluación.

# 4 Bases teóricas

Se explica una terminología que será útil comprender, previamente a leer el proyecto, para facilitar su asimilación.

**Espacio de direcciones de un proceso:** conjunto de direcciones a las que hace referencia. Pueden ser de tres tipos:

- Espacio de direcciones físicas: las direcciones físicas son aquellas que referencian alguna posición de la memoria física. Se obtienen después de aplicar una transformación por parte de la MMU (Unidad de manejo de memoria)

- Espacio de direcciones lógicas o virtuales: Son las direcciones utilizadas por los procesos. Sufren algunas transformaciones por parte de la UMM antes de transformarse en direcciones físicas. Una dirección virtual no es una dirección física que apunte a una posición de la memoria principal, sino una dirección utilizada por el sistema para acceder a una determinada página de la tabla de páginas de un proceso. Utilizando posteriormente diferentes algoritmos para traducir esa dirección virtual, el sistema puede acceder a la memoria física donde está contenida la información del proceso. Esto permite que no podamos acceder a las posiciones de otros procesos porque no es posible hacer referencia a dichas posiciones.

  Cuando un proceso es movido de la memoria física a la memoria virtual para dejar a otro que hagas sus operaciones, el sistema se encarga de modificar la tabla de páginas del proceso afectado ya sea para quitar referencias a memoria (cuando sacamos un proceso de ella), o para asignar nuevas direcciones (introduce el proceso en memoria). De ahí que sea imposible acceder a la memoria de otro proceso

- Espacio de direcciones lineales: se obtienen a partir de las direcciones lógicas tras haber aplicado una transformación dependiente de la arquitectura.

La unidad de manejo de memoria es parte del procesador. Sus funciones son:

- Convertir las direcciones lógicas emitidas por los procesos en direcciones físicas.
- Comprobar que la conversión se puede realizar. La dirección lógica podría no tener una dirección física asociada. Por ejemplo, la página correspondiente a una dirección se puede haber intercambiado a una zona de almacenamiento secundario temporalmente.
- Comprobar que el proceso que intenta acceder a una determinada posición de memoria tiene permisos para ello

Se tratará mas de una vez en este documento los sistemas con arquitectura i386. i386 es una arquitectura de hardware con unas características determinadas:

- Microprocesador de 32 bytes
- Hasta 2 Gb de memoria física direccionable.
- Soporta segmentación y paginación (se analizarán posteriormente)
- Soporta mecanismos de multiproceso ( multitarea) y de protección entre procesos, incluido protección de memoria.
- Compatibilidad con 8086 (familias de los primeros procesadores comerciales)
- Soporta un coprocesador x87

Presentado en 1985, la arquitectura se mantiene hasta hoy (Pentium 4) con mejores estructurales y un 100% de compatibilidad con versiones anteriores

Sistema operativo Linux: como gestiona la memoria

## Memoria virtual

El tamaño combinado del programa, datos y pila puede exceder la cantidad de memoria física disponible El sistema operativo guarda aquellas partes del programa concurrentemente en memoria central y el resto en disco. Cuando un programa espera que se le cargue en memoria central de disco otra parte del mismo, la CPU se puede asignar a otro proceso.

El sistema operativo gestiona niveles de memoria principal y memoria secundaria:

- Transferencia de bloques entre ambos niveles(basada en paginación)
- De memoria secundaria a principal: por demanda
- De memoria principal a secundaria: por expulsión

Esto conlleva unos beneficios: aumenta el grado de multiprogramación, permite ejecutar programas que no quepan en memoria principal.

## Paginación

El espacio virtual de direcciones se divide en unidades llamadas páginas, todas del mismo tamaño. La memoria principal se divide en marcos de páginas (page frames) del mismo tamaño que las páginas virtuales, y son compartidas por los distintos procesos del sistema.

**Tablas de páginas:** Relaciona cada página con el marco que la contiene. La UMM usa estas tablas para traducir direcciones lógicas a físicas. Típicamente existen dos, las de usuario y las de sistema. Su tamaño es una potencia de 2 y múltiplo del tamaño de bloque de disco, variando normalmente entre 2K y 16K. No se va a analizar el contenido, la gestión de las TP por parte del ordenador y su implementación, por no considerarse imprescindible para realizar el proyecto: si se desea ampliar información, se pueden consultar los enlaces al final del documento. La UMM no utiliza directamente las TP, sino que consulta las “caché de traducciones”, o TLB. Consiste en un buffer caché con información sobre las últimas páginas accedidas.

Resumiendo: la paginación es el método de traducir direcciones lineales a físicas. Se chequea el tipo de acceso respecto a los derechos de acceso de la dirección lineal, y si el acceso no es válido, se genera una excepción. Una página es un espacio contiguo de direcciones lineales de tamaño fijo (4 kb en i386), estando cada una caracterizada por

tener una dirección base y unos atributos. Para realizar la traducción, se emplean tablas de páginas.

Segmentación

El espacio de direcciones se divide en segmentos, cada uno de los cuales corresponderá a una rutina (procedimiento, función), un programa o un conjunto de datos (una entidad lógica). Todo aquello que se corresponda con sub-espacio de direcciones independientes. Cada programa contiene una cierta cantidad de segmentos. Los primeros se reservan para procedimientos, datos y pila, pertenecientes al programa en ejecución. Los restantes contienen un archivo por segmento, así que los procesos pueden direccional todos sus archivos directamente.

Resumiendo: la segmentación es la traducción de memoria virtual a dirección lineal (o virtual). Un segmento es un espacio contiguo de dirección lineales. Cada segmento está caracterizado por: dirección base, límite y atributos

Consideramos interesante resaltar que existen también dos esquemas más: la **segmentación paginada,** que intenta aunar lo mejor de los dos esquemas, proporcionando soporte directo a las regiones del proceso y permitiendo un mejor aprovechamiento de la memoria; y la **paginación por demanda,** que es adecuada para la memoria virtual.

Intercambio(swapping)

Proceso mediante el cual se intercambian programas entre memoria principal y secundaria. Cuando no existe espacio suficiente en memoria, se “expulsa” un proceso (swap out) copiando su imagen a un área de intercambio, en función de diversos criterios: prioridad o preferencia son los mas comunes. Cuando existe espacio en la memoria principal, se intercambian los procesos a memoria copiando desde el área de intercambio “swap in”.

Gestión de memoria en sistemas basados en Linux

Analizados unos cuantos conceptos que consideramos de interés, abordamos finalmente nuestro objeto de estudio. Los sistemas basados en Unix comparten multitud de aspectos con los basados en Linux, aunque ciertos puntos sustanciales son diferentes, y he preferido hacer la diferenciación. Hay que destacar no obstante que el sistema de gestión de memoria en Linux sigue siendo muy complejo. Respecto a memoria virtual, Linux hace uso de una estructura de tabla de páginas con tres niveles. Para utilizarlas, las direcciones virtuales en Linux se ven como un conjunto de 4 campos.

Para tratar de aumentar la eficiencia al cargar y descargar páginas desde o hacia la memoria, se ha definido un mecanismo peculiar. Sin entrar en demasiados detalles técnicos, basta indicar que se utiliza el *Sistema de colegas*, en el cual las páginas son

agrupadas en marcos de tamaño fijo. Para reemplazar páginas, se utiliza el *algoritmo del reloj*, en el cual las páginas tienen asignada una especie de variable de edad. Cuando es necesario reemplazar una página, aquellas quye no han sido referenciadas en bastante tiempo son las mejores candidatas.

## Direccionamiento de memoria

La memoria es uno de los recursos fundamentales para un proceso. El sistema operativo debe ofrecer la memoria a todos los procesos por igual de una forma sencilla y uniforme. Al mismo tiempo, el sistema operativo debe tratar con el hardware real para realizar dicha función, aprovechándolo al máximo. Desde el punto de vista de la arquitectura, el sistema operativo suele tener asistencia del hardware para realizar la gestión de memoria: por ejemplo, en sistgemas i386, se dispone de una unidad especializada para ello, la *Memory Management Unit.(MMU)*

Existen los tres tipos de direcciones inicialmente comentados, las lógicas, las lineales y las físicas. Las transformaciones y el formato de las direcciones depende de la arquitectura. En Linux, los espacios de direcciones lógico y lineal son idénticos. En los procesadores de la arquitectura i386, el paso de dirección lógica a lineal se denomina segmentación, y de lineal a física paginación. Si deshabilitamos la paginación, la dirección física es igual a la lineal.

## Segmentación en Linux

Linux no aprovecha la segmentación del i386. Sin embargo, no puede desactivarla, la utiliza de una forma muy limitada. Se ha preferido la paginación sobre la segmentación porque la gestión de memoria es mas simple, y uno de los objetivos de Linux es la portabilidad: muchos procesadores soportan malamente la segmentación, así que la independencia de la plataforma se vería mermada en Linux.

Paginación en Linux

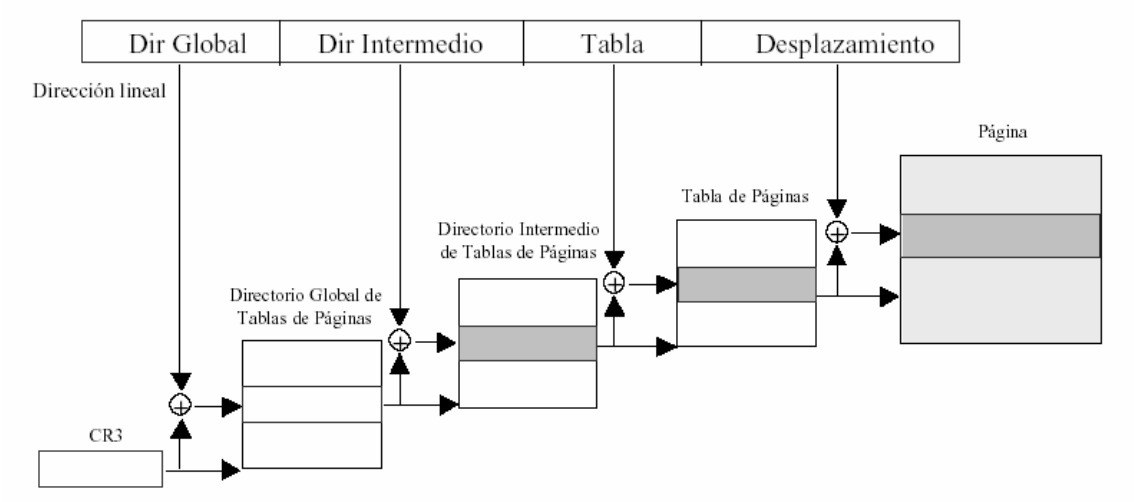



*Diagrama correspondiente al modelo de paginación de Linux.*

Linux gestiona la memoria central y las tablas de páginas utilizadas para convertir las direcciones lineales (virtuales) en direcciones físicas. Implementa una gestión de la memoria ampliamente independiente de la plataforma sobre la que se ejecuta. Este modelo no siempre se corresponde con el de la plataforma sobre el cual se ejecuta (ejemplo, el anteriormente visto procesador i386). Es tan sumamente extenso el modelo de paginación en linux que se hace imposible de abordar. No obstante, podemos consultar en el archivo fuente mm/memory.c la gestión de las tablas de página (las encargadas de realizar la traducción entre dos tipos de direcciones). Las funciones dependientes de la arquitectura se definen en asm/potable.h.

**Profundizando:**

Linux hace uso de las ventajas de la segmentación y de los circuitos de los procesadores i386 para traducir direcciones lógicas en direcciónes físicas. Podemos también decir que alguna porción de RAM está permanentemente asignada al kernely utilizada para almacenar el código del kernel y estructuras de datos estáticas del mismo. La restante parte de la RAM se denomina memoria dinámica, y ésta es un recurso muy valioso y necesitado no sólo por los procesos, sino también por el propio kernel. De hecho, el rendimiento global de un sistema depende de cómo de eficientemente se gestiona la memoria dinámica. Por lo tanto, hoy día todos los sistemas operativos multitarea tratan de optimizar el uso de la memoria dinámica, asignándola sólo cuando es estrictamente necesario y liberándola tan pronto como sea posible.

**Políticas de asignación de memoria:** se puede observar la asignación de memoria desde dos puntos de vista:

- Peticiones por parte del kernel: s un componente de alta prioridad, si solicita memoria no tiene sentido retardar su asignación. Confía en si mismo, se asume que no tiene errores de programación

- Peticiones de usuario: No tiene porque usar el espacio solicitado inmediatamente.No es confiable, y el kernel debe estar listo para capturar todos los posibles erroes de programación

Cuando un proceso solicita memoria, no se les asigna realmente páginas, sino áreas de memoria. Se les da rango de direcciones lineales válidos que se asignarán en el momento que se vayan a usar.

**Memoria para procesos:** Una de las funciones del kernel es obtener memoria dinámica de forma sencilla, llamando a una variedad de funciones. En estos casos, si la petición puede realizarse satisfactoriamente, cada una de estas funciones devuelve una dirección del descriptor de página o una dirección lineal. Cuando se asigna memoria a procesos en modo usuario, los procesos que demandan memoria dinámica se consideran no urgentes. Como regla general, el kernel trata de aplazar la asignación de memoria dinámica a procesos en modo usuario. Debido a que los programas de usuario no pueden ser fiables, el kernel debe estar preparado para capturar todos los errores de direccionamiento provocados por un proceso en modo usuario. El espacio de direcciones de un proceso son las direcciones lineales que el proceso puede utilizar. Cuando un proceso pide memoria dinámica, se le da el derecho a usar un nuevo rango de direcciones lineales. Por poner un ejemplo: una situación en la que un proceso obtiene una nueva región de memoria es cuando se crea (o se sustituye).

Linux organiza los intervalos de memoria utilizados por un proceso en lo que se denominan regiones de memoria. Estas regiones están caracterizadas por una dirección lógica inicial, una longitud y unos ciertos permisos de acceso.

Organización de las regiones de memoria: una operación que se realiza muy frecuentemente es la búsqueda de la región correspondiente a una dirección lineal. Las regiones se utilizan para representar intervalos lógicos con los mismos permisos o el mismo comportamiento.

**Flag:**La traducción literal es bandera. Consisten en etiquetas que ayudan al sistema a determinar eventos, opciones… No es indispensable ni necesario conocer su funcionamiento, aunque si interesante. Referentes a la gestión de memoria, el kernel reconoce tres tipos de flags: los incluidos en cada entrada de la tabla de página (READ/WRITE, PRESENT y USER/SUPERVISOR), los incluidos en el descriptor de página, y los relacionados con todas las páginas de una región de memoria o con la región en si. Estos flags son fundamentales para que el gestor de excepción de falta de pagina determine que tipo de fqalta ha ocurrido, y que hacer al respecto. Algunos flags importantes de región de memoria son:

VM_EXEC: Las páginas pueden ejecutarse

VM_EXECUTABLE: Las páginas contienen código ejecutable (la región mapea un archivo ejecutable)

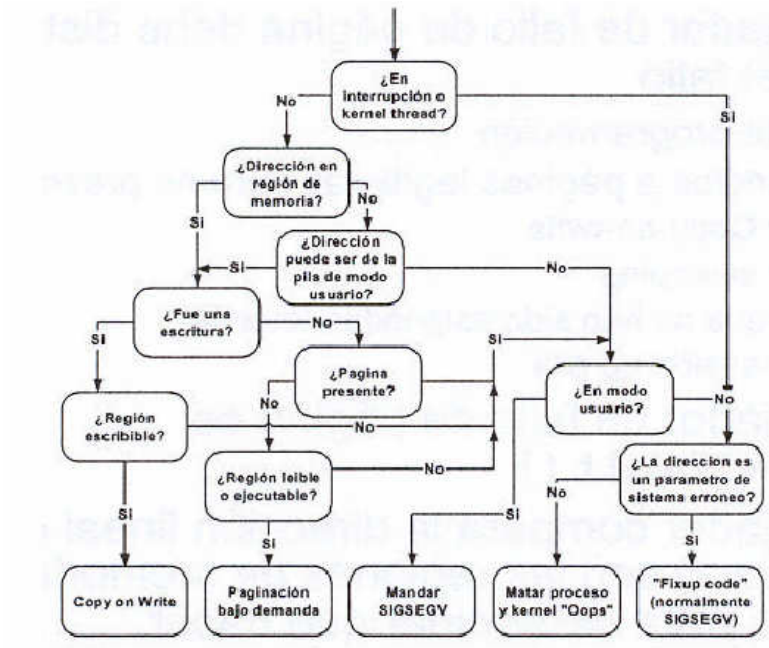
VM_READ: Las páginas pueden leerse

VM_SHARED: Las páginas pueden ser compartidas

VM_LOCKED: La región está bloqueada y no permite swap

**Gestor de faltas de página:** su función es distinguir la causa de la falta: error de programación, referencias a páginas legítimas del proceso pero no presentes, por swapping o intercambio. El gestor compara la dirección lineal que causó la falta con las regiones de memoria del proceso actual para determinar que hacer, de acuerdo con el siguiente esquema


En general, el gestor de faltas suele ser mucho más complejo, ya que ha de tener en cuenta situaciones particulares, según se muestra en la siguiente figura



## Paginación (swapping) en Linux

Si un proceso necesita cargar una página de memoria virtual a memoria física y no hay ninguna página de memoria física libre, el sistema operativo tiene que crear espacio para la nueva página eliminando alguna otra página de memoria física. Si la página que se va a eliminar de memoria física provenía de un archivo imagen o de un archivo de datos sobre el que no se ha realizado ninguna operación de escritura, entonces la página no necesita ser guardada. Tan sólo se tiene que desechar y si el proceso que estaba utilizando la vuelve a necesitar simplemente se carga nuevamente desde el archivo imagen o datos.

Por otra parte, si la página ha sido modificada, el sistema debe preservar su contenido para que pueda volver a ser accedido. Este tipo de página se conoce como página modificada y para poderla eliminar de memoria se ha de guardar en un dispositivo de swap. El tiempo de acceso al dispositivo de intercambio es muy grande en relación a la velocidad del procesador y la memoria física, y el sistema operativo tiene que conjugar la necesidad de escribir páginas al disco con la necesidad de retenerlas en memoria para ser usadas posteriormente.

Si el algoritmo utilizado para decidir que páginas se descartan o se envían a disco (algoritmo de intercambio) no es eficiente, entonces se produce una situación llamada hiperpaginación (trashing). En este estado, las páginas son continuamente copiadas a disco y luego leídas, con lo que el sistema operativo está demasiado ocupado para hacer trabajo útil. El conjunto de páginas que en el instante actual están siendo utilizadas por un proceso se llama páginas activas (working set). Un algoritmo de intercambio eficiente ha de asegurarse de tener en memoria física las páginas activas de todos los procesos. Linux utiliza la técnica de reemplazo de marco de página por antigüedad para escoger de forma equitativa y justa las paginas a ser intercambiadas o descartadas del sistema. Este esquema implica que cada página ha de tener una antigüedad que ha de actualizarse conforme la página es accedida. Cuanto más se accede a una página mas joven es; cuanto menos se utiliza más antigua e inútil. Las páginas antiguas son las mejores candidatas para ser intercambiadas.

## Dispositivos de swap

Cuando el kernel necesita memoria, puede eliminar páginas en memoria principal. Si el contenido de estas páginas ha sido modificado, es necesario guardarlas en disco: una página correspondiente a un archivo proyectado en memoria se rescribe en el archivo, y una página correspondiente a los datos se guarda en un dispositivo de swap. Puede ser una partición en disco, o un disco normal. Previamente se debe inicializar mediante la orden *mkswap*

Linux es capaz de utilizar varios dispositivos de swap. Cuando una página debe guardarse, los dispositivos de swap activos se exploran para encontrar un lugar donde

escribir la página . La activación de un dispositivo de swap se efectúa llamando a la vez a la función de sistema swapon.

## 3.2.Sistema operativo Windows: como gestiona la memoria

### Memoria en Win32

Cuando un proceso se ejecuta, el sistema establece un espacio de direcciones virtuales propio de 32 bits, que permite habilitar un espacio de hasta 4 gigabytes de memoria. Éste está formado por la suma de la memoria RAM instalada, más la memoria virtual asignada.

Windows es un sistema multiproceso, ya que permite la ejecución de varios procesos a la vez. Por norma general, no todos los procesos caben en memoria a la vez, ya sea porque existan muchos procesos, como que el tamaño de éstos sea demasiado grande. Cuando esto sucede, Windows alterna la permanencia de éstos en memoria sacando unos y poniendo otros para que todos puedan ejecutarse. Utiliza la memoria virtual asignada a cada proceso para guardar los datos cada vez que se saca de la memoria. A éste cambio entre memoria física y memoria virtual se le conoce como **Swapping**.

Para aumentar la velocidad, el cambio no se realiza byte a byte, sino página a página. cuyo tamaño en Windows es de 4 KB. De ahí que toda la memoria virtual y física esté paginada. Memoria Virtual en forma de páginas, y memoria física en forma de marcos de página.

### *Espacios de direcciones virtuales*

Una dirección virtual no es una dirección física que apunte a una posición de la memoria principal, sino una dirección utilizada por el sistema para acceder a una determinada página de la tabla de páginas de un proceso. Utilizando posteriormente diferentes algoritmos para traducir esa dirección virtual, el sistema puede acceder a la memoria física donde está contenida la información del proceso. Esto permite que no podamos acceder a las posiciones de otros procesos porque no es posible hacer referencia a dichas posiciones.

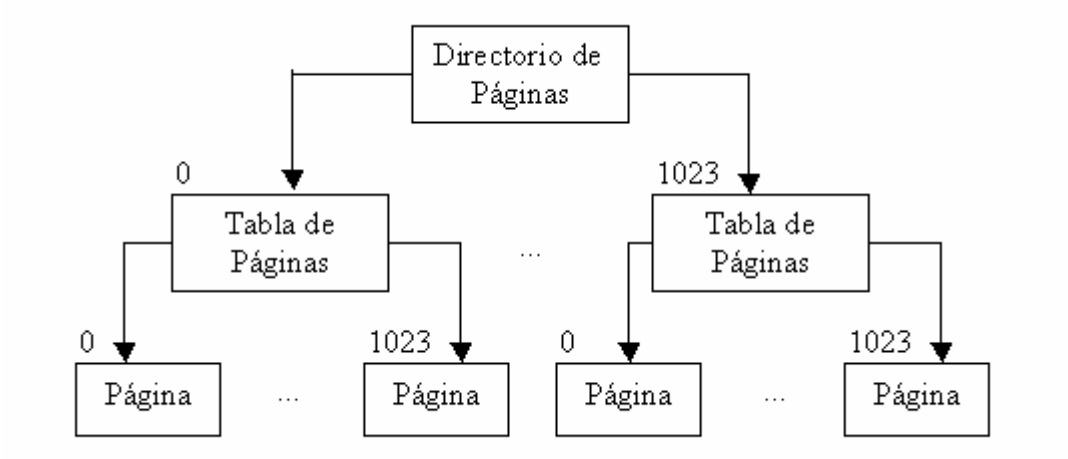
Cuando un proceso es movido de la memoria física a la memoria virtual para dejar a otro que hagas sus operaciones, el sistema se encarga de modificar la tabla de páginas del proceso afectado ya sea para quitar referencias a memoria (cuando sacamos un proceso de ella), o para asignar nuevas direcciones (introduce el proceso en memoria). De ahí que sea imposible acceder a la memoria de otro proceso.

## *Estructura de la memoria*

La estructura de la memoria en Windows es en forma de árbol, en el cual se definen claramente 3 partes:

- Directorio de Páginas (Page Directory): Cada proceso en ejecución, dispone de un solo Directorio de Páginas. Se trata de una tabla con 1024 entradas que almacena los punteros a las Tablas de Página.
- Tabla de Páginas (Page Table): Cada Tabla de Página es a su vez otra tabla que contiene otras 1024 entradas que ya apuntan a las propias páginas.
- Página (Page Frame): Cada Página es un bloque de 4 KB donde se almacenan los datos de cada proceso.

Un esquema quedaría así:




Más sencillo representarlo como una tabla de n filas (de 0 a 4.xxx.xxx.xxx) que representan páginas.

| Página | Rango de direcciones virtuales | Rango de direcciones físicas | Almacenado en... |
|---|---|---|---|
| 56 | 0x00000123 - 0x00001123 | 0x0FF231AB - 0x0FF241AB | RAM |
| 102 | 0x00000FFF - 0x00001FFF | 0x0AB124AA - 0x0AB134AA | C:\WINDOWS\Win386.swp |
| 5798 | 0xA10002FB - 0xA10012FB | 0xC000FF12 - 0xC0010F12 | Proyección C:\PEPE.DAT |
| 9487 | 0xCFBB02FB - 0xCFBB12FB | 0xAFFABC11 - 0xAFFACC11 | RAM |
| 9510 | 0xAFAB0F37 - 0xAFAB1F37 | NULL | NULL |
| [...] | [...] | [...] | [...] |

- Página: Nº de la página.

- Rango de direcciones virtuales: Rango de direcciones virtuales que componen la página. Va desde la dirección X hasta la dirección X + 4096 (4 KB).
- Rango de direcciones físicas: donde está almacenada la página. Puede estar en memoria RAM o en Memoria Virtual.
- Almacenado en... :indica donde está almacenada la página.

### *Estado de las páginas*

Las páginas pueden estar en uno de los siguientes estados:

- Libre: Una página libre no puede ser accedida por ningún proceso, pero sí puede ser reservada o encargada.
- Reservada: Una página reservada es un bloque del espacio de dirección virtual que ha sido fijada para usos futuros. No se puede acceder a una página reservada, ni tiene datos almacenados. Simplemente bloquea el rango y no permite que sea asignado por otras funciones de asignación.
- Encargada: Aquella que ya ha sido asignada a un espacio físico, ya sea en memoria física o en memoria virtual. No podemos saber donde está almacenada la página, pues de eso se encarga el sistema y puede que esté constantemente cambiándola de posición, pero lo que sí sabemos es que cuando la necesitemos, ahí lo tendremos.

Funciones para el manejo de Memoria Virtual

Windows provee a los procesos de varias funciones para realizar reservas de memoria, asignación de espacio físico, cambio de permisos, etc...

## VirtualAlloc

Utilizaremos esta función para reservar y asignar la memoria que queremos utilizar. Su prototipo es:

```
LPVOID VirtualAlloc( LPVOID lpAddress,
                     SIZE_T dwSize,
                     DWORD flAllocationType,
                     DWORD flProtect);
```

- LpAddress: Indica la dirección inicial de la región que queremos reservar.

- DwSize: Tamaño de la región, en bytes.
- FlAllocationType: La acción que queremos llevar a cabo. Puede ser cualquiera de las siguientes constantes.:
    - MEM_COMMIT: Asigna el almacenamiento físico.
    - MEM_RESERVE: Reserva la memoria que necesitemos.
    - MEM_RESET:

- LpProtect: Indica el tipo de protección que hay que aplicar a la memoria encargada. Solo cuando se usa la opción MEM_COMMIT. Puede ser cualquiera de las siguientes constantes: PAGE_EXECUTE, PAGE_EXECUTE_READ, PAGE_EXECUTE_READWRITE, PAGE_EXECUTE_WRITECOPY, PAGE_NOACCESS, PAGE_READONLY, PAGE_READWRITE, PAGE_WRITECOPY

## VirtualFree

Esta función se utiliza para liberar la memoria reservada o para des-asignar la memoria asignada.

```
BOOL VirtualFree( LPVOID lpAddress,
                  SIZE_T dwSize,
                  DWORD dwFreeType);
```

- LpAddress: Indica la dirección inicial de la región que queremos reservar.
- DwSize: Tamaño de la región, en bytes.
- DwFreeType: Tipo de liberación:
    - MEM_DECOMMIT: Des-asigna un rango de memoria que ha sido asignado. Si no hay memoria asignada de antemano no ocurre nada.
    - MEM_RELEASE: Libera el rango solicitado. Si en el rango hay memoria asignada, primero des-asigna la memoria y luego libera.

**Otras funciones**:

- VirtualLock: Bloquea un grupo de páginas (previamente comprometidas) en memoria física, evitando que sean pasadas al archivo de intercambio cuando el sistema necesita memoria RAM. Lo normal es dejar a Windows que se encargue de todo.

- VirtualUnlock :Desbloquea un grupo de páginas que han sido previamente bloqueadas con VirtualLock.
- VirtualProtect: Cambia el estado de protección de un grupo de páginas del proceso activo.
- VirtualProtectEx : Cambia el estado de protección de un grupo de páginas de un proceso dado.
- VirtualQuery: Consulta el estado y el tipo de un grupo de páginas del proceso activo.
- VirtualQueryEx: Consulta el estado y el tipo de un grupo de páginas de un proceso dado.

.

# 5 Pruebas diseñadas

La idea principal gira en torno al tiempo que tarda un sistema en cargar datos a Memoria Principal (memoria RAM) en función de los datos, comprobando si el aumento de estos provoca un aumento del tiempo con una progresión lineal, logarítmica o exponencial.

El experimento se divide en dos partes:

**Primera parte.**

Teniendo memoria RAM libre, suficiente para contener los datos sin provocar el uso de swap, reservar una cantidad de memoria pequeña (varias veces, previa liberación de los datos) y medir el tiempo. Aumentar un poco la memoria que queremos reservar y volver a medir tiempo (nuevamente, varias veces), y así tantas veces como queramos pero sin llegar a sobrepasar la memoria libre, recordemos que no queremos provocar el swapping.

En principio, da igual la cantidad de memoria libre que quede, solo es importante que no existan otros procesos utilizando el procesador.

**Segunda parte.**

El objetivo de ésta segunda parte es el mismo que para la primera, con la salvedad de que ahora obligamos al Sistema Operativo a no tener memoria libre para hacer frente a las peticiones de reserva de datos. Esto, en teoría, debería provocar la utilización del swapping.

Para llevar a cabo este proceso, utilizamos previamente un programa que reserve una cantidad de memoria suficiente para dejar libre la cantidad que

nosotros queramos. A continuación, seguimos el mismo proceso que para la parte primera de ésta prueba.

Para una mayor fiabilidad de los datos, se repite cada medida 100 veces y se utiliza la media ponderada para la elaboración de las gráficas.

Para poder comparar posteriormente si la cantidad de tiempo requerido para operación variaba en una función lineal, logarítmica o exponencial, se hizo una estimación del tiempo que debería tardar si fuera una progresión lineal en base a una medida con una cantidad de memoria pequeña.

Se usó el siguiente hardware tanto bajo Windows como bajo Linux:

Pentium 4 a 1'7 GHz, 259 MB de RAM y caché L1 de 20 KB, y L2 de 256 KB

- Sistema Operativo Windows 2000 Pro, Service Pack 4 (vers. 50.2195 )
- Debian Linux con el kernel 2.6.8-2-686

# 6 Resultados experimentales

Se obtuvieron los siguientes datos y sus correspondientes tablas comparativas:

**Para windows:**

Sin swapping 100 repeticiones

| Cantidad (KB) | Tiempo real | Tiempo estimado |
|---|---|---|
| 10000 | 3,687 | 3,100 |
| 20000 | 7,526 | 6,200 |
| 30000 | 11,302 | 9,300 |
| 40000 | 15,083 | 12,400 |

Con swapping 100 repeticiones

| Cantidad (KB) | Tiempo real | Tiempo estimado |
|---|---|---|
| 10000 | 7,375 | 3,100 |
| 20000 | 10,406 | 6,200 |
| 30000 | 11,240 | 9,300 |
| 40000 | 18,334 | 12,400 |

**Para Linux:**

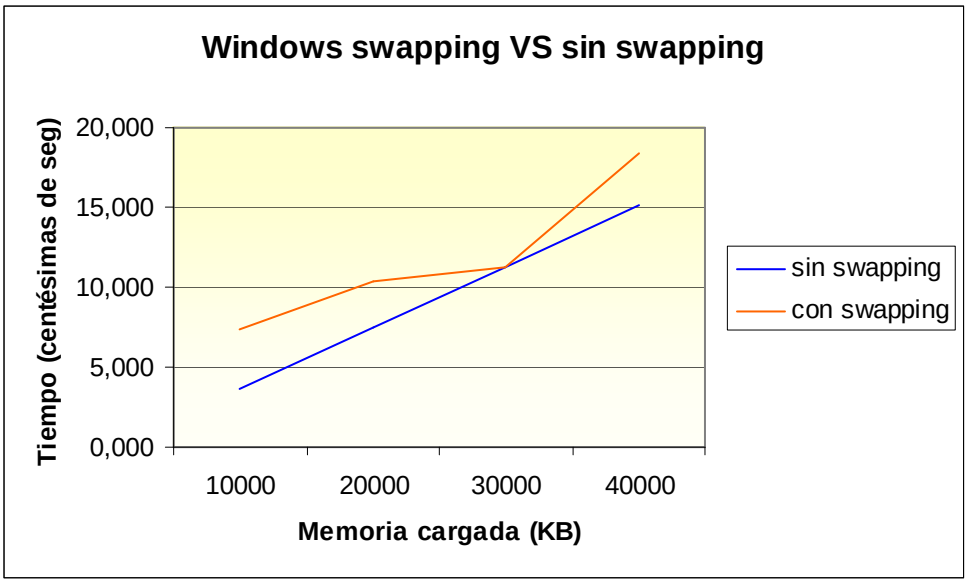
Sin swapping 100 repeticiones

| Cantidad (KB) | Tiempo real | Tiempo estimado |
|---|---|---|
| 10000 | 5,520 | 9,000 |
| 20000 | 10,563 | 18,000 |
| 30000 | 15,600 | 27,000 |
| 40000 | 20,647 | 36,000 |

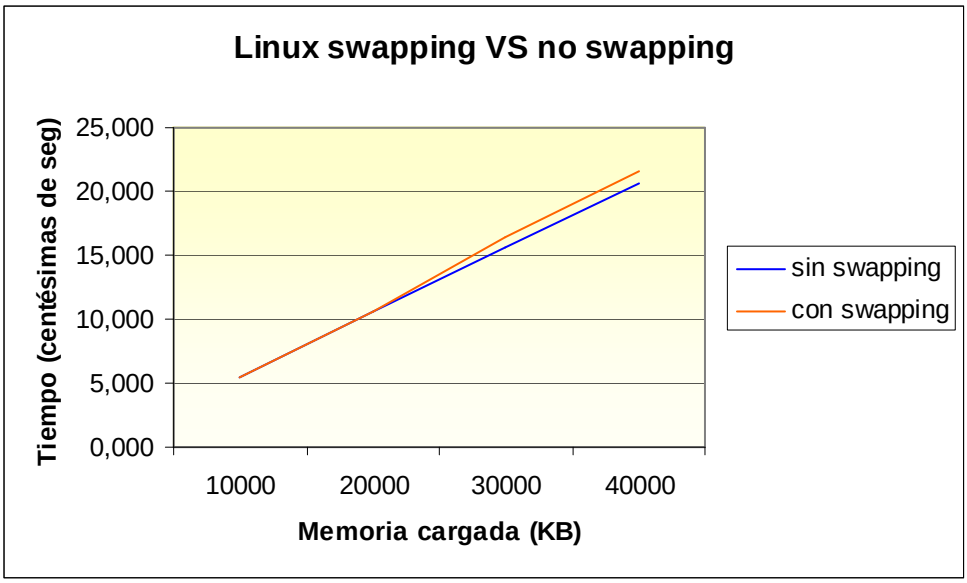
Con swapping 100 repeticiones

| Cantidad (KB) | Tiempo real | Tiempo estimado |
|---|---|---|
| 10000 | 5,373 | 9,000 |
| 20000 | 10,886 | 18,000 |
| 30000 | 16,176 | 27,000 |
| 40000 | 22,224 | 36,000 |

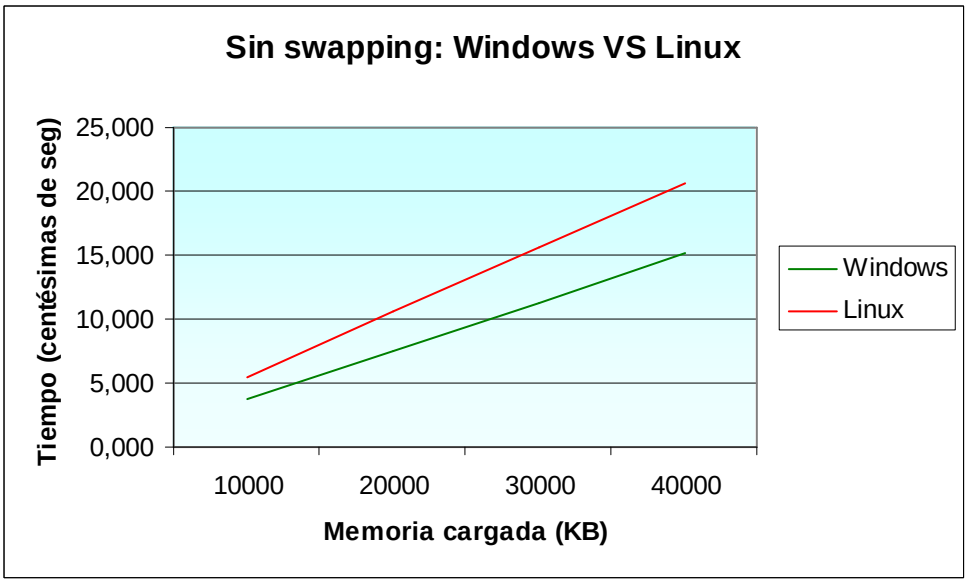
**Tablas comparativas:**



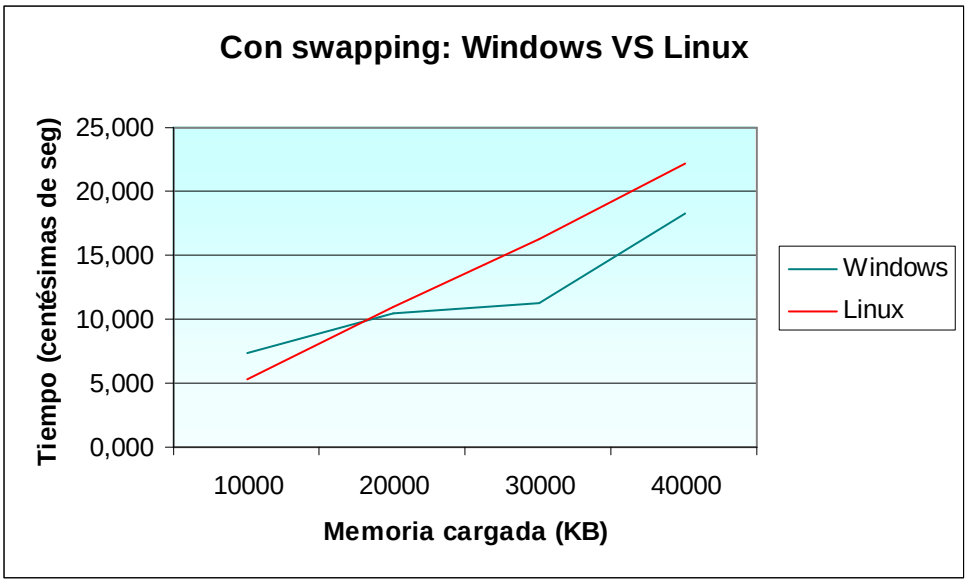

**Gráfica 1**



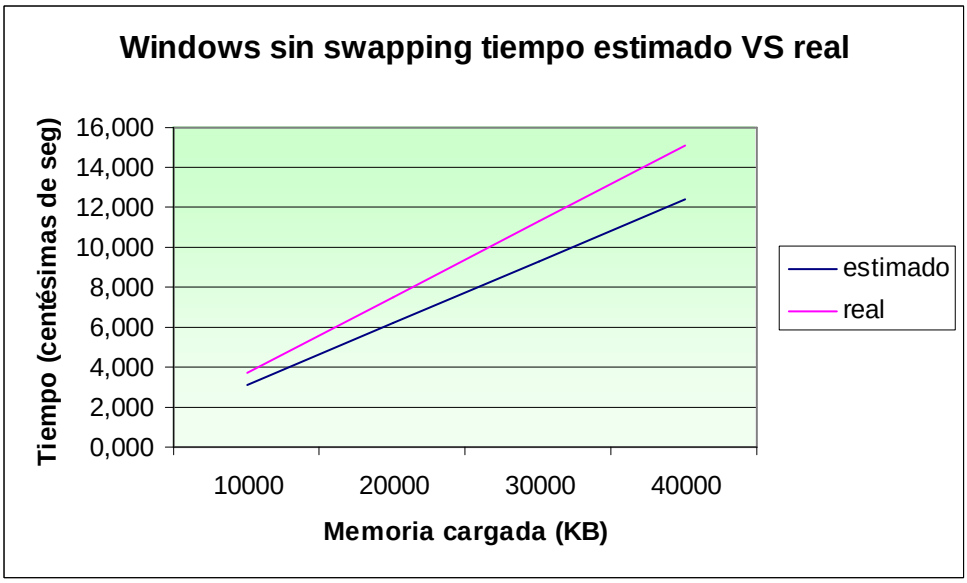

**Gráfica 2**



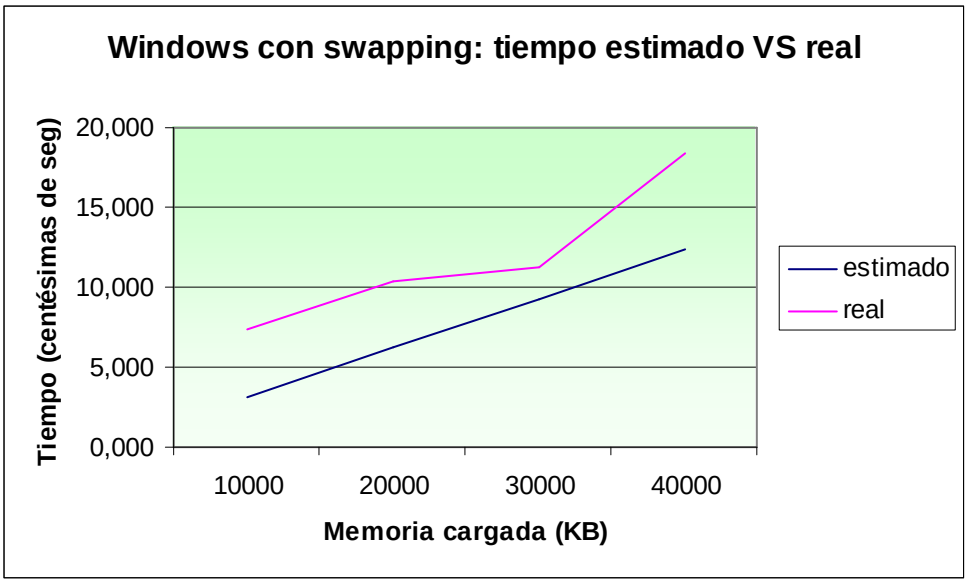

**Gráfica 3**



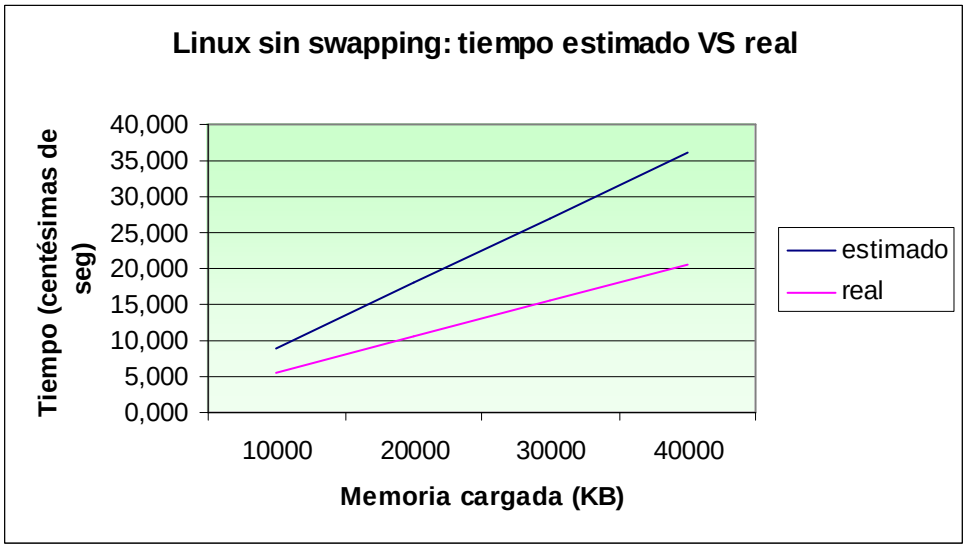

**Gráfica 4**



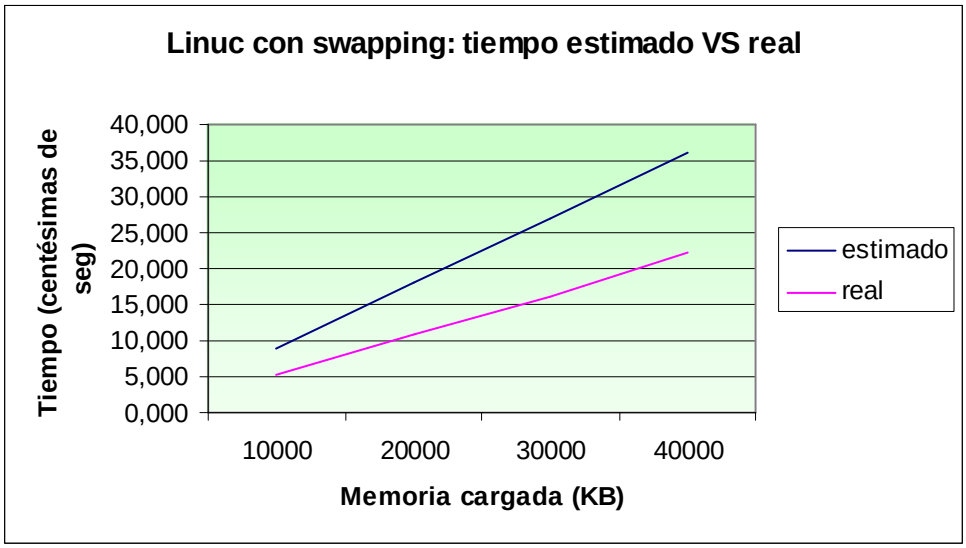

**Gráfica 5**




**Gráfica 6**




**Gráfica 7**




**Gráfica 8**

# 7 Conclusiones

Se pueden sacar las conclusiones enumeradas a continuación:

1. Windows tarda más en acceder a la memoria cuando requiere swapping y además su comportamiento se vuelve más impredecible, un argumento que se suma a la de los defensores de Linux que le atribuyen una mayor robustez contra caídas del sistema. *(Gráficas 1 y 2).*
2. Linux tiene aproximadamente el mismo tiempo de acceso a memoria cuando sus procesos requieren swapping o no. Además su comportamiento es siempre lineal. *(Gráficas 1 y 2).*
3. El acceso a memoria es siempre mucho más rápido con Windows que con Linux tanto cuando requieren swapping como cuando no salvo de procesos muy pequeños que requieren menos de 2 Megas de swap. Éste es el caso de los procesos básicos basales para el sistema. *(Gráficas 4 y 5).*
4. El comportamiento de Windows es más caótico cuanto más memoria exigen los procesos que ejecuta aunque estos sean siempre el mismo repetidas varias veces, mientras que la gestión más efectiva de la cache de Linux le permite mejorar más su tiempo cuanto más usado es un mismo proceso en periodos cercanos de tiempo. *(Gráficas 6, 7, 8 y 9).*

# 8 Bibliografía y fuentes


1. Outline of the Linux Memory Management System, de Joe Knapka: http://home.earthlink.net/~jknapka/linux-mm/vmoutline.html
2. Linux Memory Management Documentation: http://www.linux-mm.org/LinuxMMDocumentation
3. The Linux Documentation Project : http://www.tldp.org
4. Memoria en Win32: http://articulos.conclase.net/jm/prog/cpp/memoriavirtual.html
5. Administración de memoria en Win32: http://foro.elhacker.net/index.php/topic,64371.0.html
6. "Sistemas Operativos Modernos" Andrew S. Tanenbaum. 1993. Prentice Hall.
7. "Sistemas Operativos" William Stallings. 1997 Prentice Hall.
8. "Operating System Concepts" Abraham Silberscatz y Peter B. Galvin. 1994 Addison-Wesley.