Resolución de Problemas y Algoritmos

Clase 8 (2^3)
Almacenamiento en Memoria.
Sistemas Operativos.
Archivos secuenciales en Pascal.


Joseph C.R. Licklider

Dr. Alejandro J. García
http://cs.uns.edu.ar/~ajg

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca - Argentina

Memoria principal y memoria secundaria

Memoria principal
RAM (Random Access Memory)
Contiene: programas en ejecución y datos

Memoria secundaria
Ejemplos: disco rígido, pen-drive, dvd, unidad de estado sólido, "la nube".
Contiene: Archivos de programas, textos, datos, fotos, videos, música, etc.

bus: canal de comunicación

Memoria secundaria: disco duro

Sistema de grabación magnética con uno o más platos o discos rígidos, unidos por un mismo eje que gira a gran velocidad

Memoria secundaria: DDS (Solid State Drive)

Una unidad de estado sólido o SSD (acrónimo de solid-state drive) es un dispositivo de almacenamiento de datos que usa una memoria no volátil (no es un disco).

Memoria secundaria (SSD - solid state drive)

Una unidad de estado sólido o SSD (acrónimo de solid-state drive) es un dispositivo de almacenamiento de datos que usa una memoria no volátil, como la memoria flash, o una memoria volátil como SDRAM, para almacenar datos, en lugar de los platos giratorios magnéticos encontrados en los discos duros convencionales.

En comparación con los discos duros tradicionales, las SSD son menos sensibles a los golpes, son prácticamente inaudibles, y son más rápidas.

A veces se traduce erróneamente en español la "D" de SSD como "disco" cuando, en realidad, representa la palabra "drive", que podría traducirse como unidad o dispositivo.

A partir de 2010, la mayoría de los SSDs utilizan memoria flash basada en compuertas NAND, que retiene los datos sin alimentación.

Almacenamiento en "la nube"

El almacenamiento en nube posee las mismas características que la computación en nube con respecto a agilidad, escalabilidad, elasticidad y multiposesión.
Se considera que el concepto se forjó en los años '60 por Joseph Carl Robnett Licklider.


Joseph C.R. Licklider

Almacenamiento en nube se define como un entorno de almacenamiento compuesto por muchos recursos distribuidos, pero actúa como un solo almacenamiento con gran tolerancia a fallos porque implementa redundancia y distribución de datos.

http://es.wikipedia.org/wiki/Almacenamiento_en_nube

Dropbox
Google Drive
iCloud
OneDrive (SkyDrive)

Almacenamiento en memoria

Memoria RAM: (Random Access Memory)
Es una memoria volátil de gran velocidad de acceso.

El procesador debe accederla lo más rápido posible, y por ello es de alto costo (en 2015, 4Gb a $600). ¿Cuánto costaría 1 TB de RAM?

Los valores de las variables de tipos integer, char, real y boolean se almacenan en RAM. Por su tecnología es volátil (los datos no perduran al cortar la energía del sistema).

http://es.wikipedia.org/wiki/Memoria_de_acceso_aleatorio

Memoria secundaria: Es una memoria de mucho menor costo y por ello de menor velocidad. Ejemplos en 2015: Disco rígido externo 1Tb $1000 ($1 el Gb) - Pendrive 16Gb $100 - SSD 250Gb $2500

Por su tecnología (magnética, óptica, flash, etc.) permite que los valores perduren aún cuando no tenga energía eléctrica.

Los valores de los archivos están en memoria secundaria.

Unidades de almacenamiento

Unidad	Cant. Bits	Equivale a
1 Bit		
1 Byte	2^3	8 bits
1 Kilobyte (KB)	2^{10}	1024 bytes
1 Megabyte (MB)	2^{20}	1024 KB = 1.048.576 bytes
1 Gigabyte (GB)	2^{30}	1024 MB = 1.073.741.824 bytes
1 Terabyte (TB)	2^{40}	1024 GB = 1.099.511.627.776 bytes
1 Petabyte (PB)	2^{50}	1024 TB = 1.125.899.906.842.624 bytes
1 Exabyte (EB)	2^{60}	1024 PB = 1.152.921.504.606.846.976 bytes
1 Zettabyte (ZB)	2^{70}	1024 EB = 1.180.591.620.717.411.303.424 bytes
1 Yottabyte (YB)	2^{80}	1024 ZB = 1.208.925.819.614.629.174.706.176 bytes

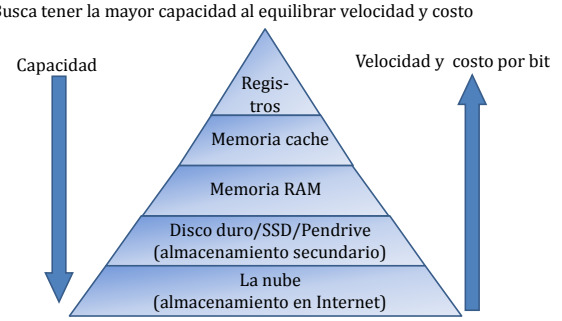
Ejemplos

Unidad
Bit
Byte
Kilobyte (KB)
Megabyte(MB)
Gigabyte (GB)
Terabyte (TB)
Petabyte (PB)
Exabyte (EB)
Zettabyte (ZB)
Yottabyte (YB)

- Un valor real ocupa generalmente 6 bytes.
- El código fuente en Pascal "veces.pas" de la clase pasada ocupa 670 bytes.
- El ejecutable de ese programa ocupa 69 Kb.
- El archivo ppt de esta clase ocupa 1 Mb.
- La memoria RAM de una PC en 2015 era de 4 GB.
- En 2015 un disco rígido típico era de 1 TB.
- El contenido de la web en 1995 se estimaba en 8 PB.
- El contenido de la web en 2009 se estimaba en 500EB.
- En un gramo de ADN contiene unos 455 EB de información.
- El contenido de la web en 2013 se estimaba en 4 ZB (¿unos 8 gramos de ADN?).

Jerarquía de Memorias

Busca tener la mayor capacidad al equilibrar velocidad y costo



¿Quién conoce todo el Hardware?

¿Qué es un Sistema Operativo?

Memoria principal
RAM (Random Access Memory)
Contiene: programas en ejecución y datos

Memoria secundaria
Ejemplos: disco rígido, pen-drive, dvd, unidad de estado sólido, "la nube".

Contiene: Archivos de programas, textos, datos, fotos, videos, música, etc.

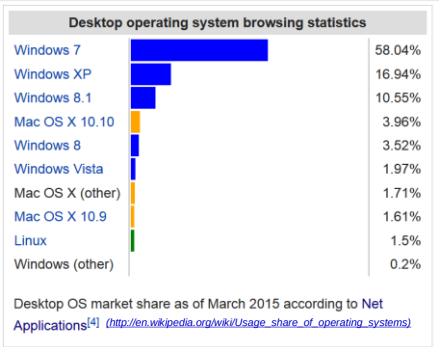
bus: canal de comunicación

Conceptos: Sistema Operativo (Operating System)

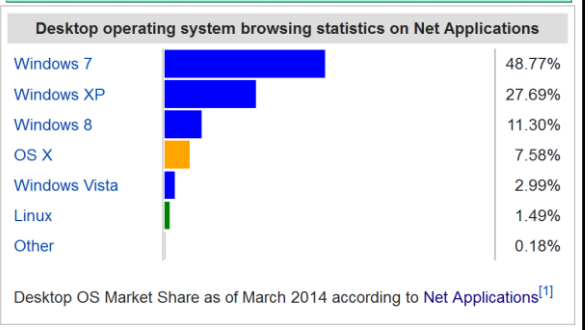
- Un **sistema operativo (SO)** es un programa que gestiona los recursos de hardware y provee servicios a los programas de aplicación. Se ejecuta en modo privilegiado respecto de los restantes programas.
- Es un programa que actúa como un intermediario entre un usuario y el hardware de la computadora.

Cada carrera tiene una materia para SO:
- Ing: "*Sistemas operativos*"
- Lic: "*Sistemas operativos y distribuidos*"

Sistemas Operativos – Estadísticas 2015 (de web servers)

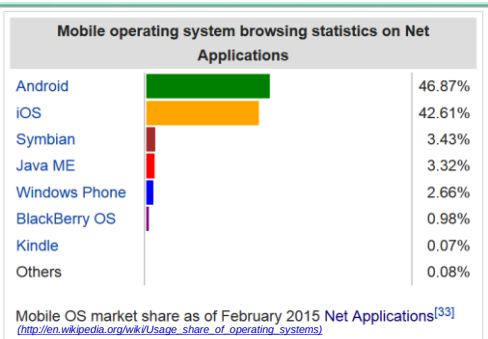


Sistemas Operativos – Estadísticas 2014 (de web servers)

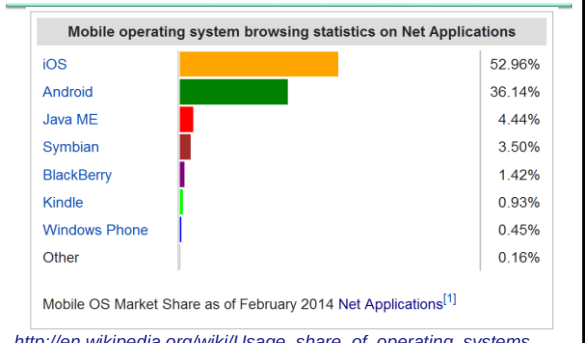


http://en.wikipedia.org/wiki/Usage_share_of_operating_systems

Sistemas Operativos (móviles)- Estadísticas 2015 (web servers)

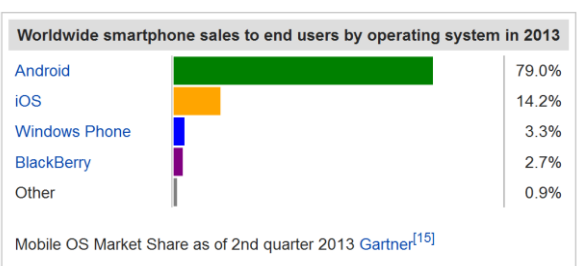


Sistemas Operativos (móviles)- Estadísticas 2014 (web servers)



http://en.wikipedia.org/wiki/Usage_share_of_operating_systems

Sistemas Operativos (smartphones) Estadísticas de ventas 2013



http:// en.wikipedia.org/wiki/Usage_share_of_operating_systems

Sistemas operativos

- Es importante el "concepto" y no el "producto" porque en su carrera y trabajo profesional usará muchos sistemas operativos.
- Estos son algunos de los que he usado hasta ahora:
  - RSTS (en PDP-11) y VMS (en VAX)
  - MS-DOS, Windows (3, NT, XP, Vista, 7, 8),
  - Unix, Linux, AIX, Symbian, Android,
- Puede mirar más sobre sistemas operativos en:
  http://es.wikipedia.org/wiki/Sistema_operativo
  http://en.wikipedia.org/wiki/Mobile_operating_system
- Un resumen de la cronología histórica de los sistemas operativos
  http://en.wikipedia.org/wiki/Timeline_of_operating_systems

Sistema Operativo (SO)

Por ejemplo:

- Un sistema operativo conoce los detalles del hardware del almacenamiento secundario (como un disco rígido) y provee servicios a los programas de aplicación para el **manejo de archivos**.
- Puede haber archivos de video, de música, de imágenes, de texto, de programas ejecutables, etc.
- Observe que el nombre del archivo le permite al sistema operativo asociarlo con una aplicación.

Nombres de archivos en un SO

- Un nombre válido para un archivo dependerá del SO.
- Por ejemplo, en las primeras versiones del sistema operativo MS-DOS (1981-1995), un nombre de archivo tenía el formato: nnnnnnnn.EEE (*i.e.*, 8 caracteres para el nombre y 3 para la extensión)
- La extensión es usada por el SO para identificar el tipo de archivo (ej: MP3, AVI, PAS, EXE, JPG).
- Windows actualmente limita a 260 caracteres, incluyendo el camino (path) y el nombre. No se pueden usar los símbolos **\ / ? : * " > < |**
- Ej: C:\usuarios\ale\RPA\clase-1(ale).pru.nums.pas

extensión

camino o ruta

nombre

CONCEPTOS: Valores de variables en Pascal

- Muchas veces es útil que determinados valores puedan perdurar aunque el programa termine, y que estos valores puedan ser utilizados en el futuro por otro programa.
- En Pascal existe un tipo de dato estructurado **FILE** (archivo/fichero) que permite almacenar valores que pueden perdurar aún cuando la ejecución del programa termine.
- De esta manera, un programa podrá "leer" elementos generados por otro programa; y además "escribir" datos que podrá leer otro programa (o él mismo pero en otra ejecución posterior).

Conceptos:

Tipo de Dato: define el conjunto de valores posibles que puede tomar una variable, y también define las operaciones que pueden usarse.

Los tipos de datos en Pascal se pueden dividir en:

- **Simples (Ejemplos que vimos en RPA: INTEGER, REAL, CHAR, BOOLEAN)**
- **Estructurados (Ejemplos que veremos en RPA: FILE (archivo) y TEXT (archivo de texto))**

Declaración de archivos secuenciales en Pascal

- **La palabra reservada FILE permite declarar un tipo o una variable de tipo archivo.**

```
Program ejemplo;

VAR  numeros,valores: FILE OF integer;
     caracteres: FILE OF char;
     temperaturas FILE OF real;
     datos: FILE OF boolean;
```


CONCEPTOS: Archivos (Files)

- Para que el contenido de un archivo (file) perdure más allá de la ejecución de un programa y aún cuando la computadora estuviera apagada por un tiempo, los archivos (files) residen en memoria secundaria (como un disco rígido por ejemplo).
- Es importante notar que el acceso a memoria secundaría depende del Sistema Operativo usado.
- El manejo de archivos en Pascal también puede tener diferencias de un compilador a otro.
- En esta materia se verán algunos conceptos de archivos secuenciales y algunos detalles estarán ligados al sistema operativo o al compilador.

CONCEPTOS: Archivos secuenciales

- Un Sistema Operativo (SO) puede manejar distintas clases de archivos (de texto, fotos, películas, música, ejecutables).
- Cada Lenguaje de Programación (LP) pueden manejar algunas de estas clases de archivos.
- En Pascal hay una clase de archivo que se denomina archivo secuencial (**FILE**) .
- Un archivo secuencial es una estructura compuesta por una secuencia de elementos (componentes) donde hay un orden lineal entre ellos.

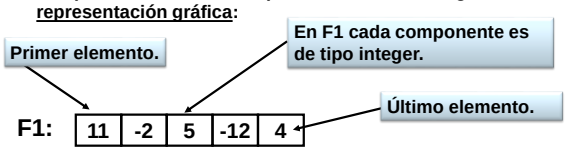
Archivos secuenciales en Pascal (FILE OF …)

- En Pascal, un archivo secuencial (FILE) es una sucesión finita de componentes que pueden accederse una a una, comenzando de la primera.
- Todos los componentes deben ser del mismo tipo de datos (ej. todos integer, o todos char).
- Por ser un archivo secuencial, una vez accedida la primera componente, el acceso a la componente de la posición P se logra luego de haber accedido previamente a la posición P-1.
- La cantidad de componentes es potencialmente infinita (su límite estará dado por la cantidad de espacio en la computadora donde está el archivo).
- Son almacenados en Memoria Secundaria.

Representación gráfica

```
Program ejemplo;
VAR F1: FILE OF integer;
```

- **Dado que una variable de tipo FILE es una sucesión de componentes del mismo tipo, es usual usar la siguiente representación gráfica:**



- **Aunque la capacidad de un archivo es potencialmente infinita, en cualquier momento dado, el archivo tendrá un número finito de componentes.**

Nombres de archivos secuenciales en Pascal

- El identificador de una variable de tipo **FILE** es un nombre interno usado por el programa en Pascal para referirse a un archivo secuencial.
- Como los valores almacenados en variables de tipo **FILE** van a estar en memoria secundaria (por ejemplo: disco rígido), el sistema operativo necesita asignarle un nombre válido en ese sistema.
- Este archivo puede ser usado en el futuro por otro programa usando ese nombre dado en el SO.
- La primitiva **ASSIGN** permite vincular el nombre interno (identificador de variable) y el nombre del archivo en el SO.

Primitiva ASSIGN

```
Program ejemplo;
VAR numeros: FILE OF integer;
Begin
ASSIGN(numeros, 'mis-numeros.datos');
…
```

- La primitiva **ASSIGN(F, N)** tiene dos parámetros: **F** que es un identificador de variable de tipo FILE, y **N** que es una secuencia de caracteres que representa un nombre válido de archivo en el sistema operativo.
- El identificador **F** es llamado **manejador del archivo** de nombre **N** (*file handler*), y en el código fuente toda otra referencia al archivo se hace usando el manejador **F**.
- Una vez ejecutada **assign** vincula a **F** con **N** (el nombre real del archivo en memoria secundaria).

Primitivas de Pascal para archivos secuenciales

- assign(F,N): vincula F con N (nombre del archivo en SO).
- rewrite(F): crea un archivo nuevo con el nombre N que está vinculado a F (si ya existe otro archivo con ese nombre N se sobre-escribe y se pierde el viejo archivo).
- write(F,e): en un archivo F creado con rewrite, escribe el valor de "e" a continuación del último elemento de F.
- close(F): cierra el archivo vinculado al *manejador* F.

(veamos un ejemplo antes de ver las tres restantes)

Problema: escriba un programa para crear un archivo llamado "mis-numeros.dat" y escribir en él enteros de 1 a un tope ingresado por el usuario.

```
Algoritmo
    Crear el archivo "mis-numeros.dat" para escribir en él.
    Solicitar un entero tope
    Escribir en el archivo "tope" enteros desde el 1 hasta tope
    Cerrar el archivo
fin.
```

Observación: En este algoritmo conozco de antemano cuanto elementos quiero escribir en el archivo, entonces en Pascal puedo usar FOR.

Problema: escriba un programa para crear un archivo llamado "mis-numeros.dat" y escribir en él enteros de 1 a un tope ingresado por el usuario.

```
Program crear;
VAR nuevo: FILE OF integer;
    valor,tope: integer;
begin
   assign(nuevo, 'mis-numeros.dat');
   rewrite(nuevo);  {crea archivo vacío y permite escribir en él}
   writeln('Cantidad de enteros a escribir en el archivo ');
   readln(tope);
   for valor:= 1 to tope do write(nuevo,valor);
   close(nuevo);
end.
```

Observación:
Si tope=100 generará un archivo de 400 bytes.

Escribe un entero al final del archivo

Primitivas de Pascal para archivos secuenciales

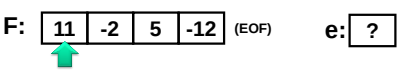
- assign(F,N): vincula F con N (nombre del archivo en SO).
- rewrite(F): crea un archivo nuevo con el nombre N que está vinculado a F (si ya existe otro archivo con ese nombre N se sobre-escribe y se pierde el viejo archivo).
- write(F,e): en un archivo F creado con rewrite, escribe el valor de "e" a continuación del último elemento de F.
- close(F): cierra el archivo vinculado al *manejador* F.
- reset(F): abre un archivo existente de nombre N para leer, y queda preparado para leer el primer elemento.
- read(F,e): lee un elemento del archivo F, copia el valor leído en "e" y queda preparado para leer el siguiente elemento (si existe) o queda en el fin del archivo.
- eof(F) (end of file – fin de archivo): retorna TRUE si se llegó al final de un archivo o FALSE en caso contrario.

Lectura de datos de un archivo

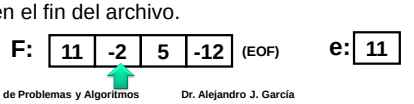
Considere las siguientes variables:

VAR F: FILE OF integer; e: integer;

reset(F): abre el archivo asociado al manejador F para leer, y queda preparado para leer el primer elemento.



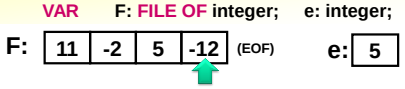
read(F,e): lee el elemento del archivo F que está listo para ser leído, copia el valor leído en la variable "e" y queda preparado para leer el siguiente elemento (si existe) o queda en el fin del archivo.



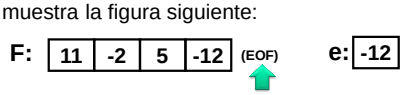
Lectura de datos de un archivo

Considere las siguientes variables:

VAR F: FILE OF integer; e: integer;



Si ya fueron leídos los tres primeros elementos, como se muestra en la figura de arriba, al hacer read(F,e) se lee el último elemento y se llega al final del archivo (end of file) como muestra la figura siguiente:

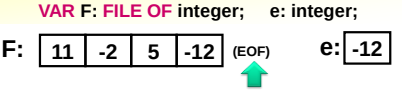


¿Qué ocurre si ejecuto read(F,e) ahora?

Lectura de datos de un archivo

Considere las siguientes variables:

VAR F: FILE OF integer; e: integer;



MUY IMPORTANTE: si la primitiva "**read(F,E)**" es usada sobre el fin de un archivo (o un archivo vacío) es considerado un error de programación ya que dará un error y el programa dejará de ejecutarse.

Por lo tanto antes de usar "**read(F,e)**" debe asegurarse no estar al final del archivo con la función **eof(F)**.

Lectura de datos de un archivo

Considere las siguientes variables:

VAR F: FILE OF integer; e: integer;

Al hacer "reset(F)" puede ocurrir que el archivo esté vacío y se tiene una situación como se muestra a continuación:

F: (EOF)

MUY IMPORTANTE: si la primitiva "**read(F,E)**" es usada sobre el fin de un archivo (o un archivo vacío) es considerado un error de programación ya que dará un error y el programa dejará de ejecutarse.

Por lo tanto antes de usar "**read(F,e)**" debe asegurarse no estar al final del archivo con la función **eof(F)**.

Lectura de datos de un archivo

```
…
assign(F, 'enteros');
reset(F);
read(F,e);
writeln(' primero:', e);
…
```

MAL

Error de programación: no controla si el archivo está o no vacío

MUY IMPORTANTE: si la primitiva "**read(F,E)**" es usada sobre el fin de un archivo (o un archivo vacío) es considerado un error de programación ya que dará un error y el programa dejará de ejecutarse.

Por lo tanto antes de usar "**read(F,e)**" debe asegurarse no estar al final del archivo con la función **eof(F)**.

Lectura de datos de un archivo

```
…
assign(F, 'enteros');
reset(F);
if not eof(F) then
begin
   read(F,e);
   writeln(' primero:', e);
end
…
```

OK

```
…
assign(F, 'enteros');
reset(F);
while not eof(F) do
begin
   read(F,e);
   writeln(' encontré:', e);
end
…
```

OK

IMPORTANTE: antes de usar "**read(F,e)**" debe asegurarse no estar al final del archivo con la función **eof(F)**.

Lectura de datos de un archivo

```
…
assign(F, 'enteros');
reset(F);
repeat
   read(F,e);
   writeln(' encontré:', e);
until eof(F)
…
```

MAL

Error de programación: no controla si el archivo está o no vacío

IMPORTANTE: antes de usar "**read(F,e)**" debe asegurarse no estar al final del archivo con la función **eof(F)**.

Problema: escriba un programa para abrir un archivo ya existente llamado "mis-numeros.dat", leer todos sus componentes, y mostrarlos por pantalla.

```
Algoritmo
   Abrir el archivo "mis-numeros.dat" para leer sus elementos
   Leer uno a uno los elementos y mostrarlos en pantalla
   Cerrar el archivo.
fin.
```

MUY IMPORTANTE: si la primitiva "**read(F,E)**" es usada sobre el fin de un archivo (o un archivo vacío) es considerado un error de programación ya que dará un error y el programa dejará de ejecutarse.

Por lo tanto antes de usar "**read(F,e)**" debe asegurarse no estar al final del archivo con **eof(F)**.

Problema: escriba un programa para abrir un archivo ya existente llamado "mis-numeros.dat", leer todos sus componentes, y mostrarlos por pantalla.

```
Program leer;
VAR arch_num: FILE OF integer;      elemento: integer;
begin
   assign(arch_num, 'mis-numeros.dat');
   reset(arch_num); {abre el archivo para leer de él}
   while not eof(arch_num) do {mientras no llegue al fin}
   begin
      read(arch_num,elemento); writeln(' fue leido:', elemento);
   end;
   close(arch_num);
end.
```

Lee un elemento del archivo y queda preparado para leer el siguiente.

Problemas propuestos

- Problema: escriba un programa que cuente cuantos elementos tiene el archivo "mis-numeros.dat" (ya creado y con números en él).
- Problema: escriba un programa que busque cuantas veces está el elemento E (ingresado por el usuario) en el archivo "mis-numeros.dat" (ya creado y con números en él).
- Problema: escriba un programa que vea si primer elemento es igual al último en el archivo "mis-numeros.dat" (ya creado y con números en él).
- Problema: escriba un programa que vea si los elementos del archivo "mis-numeros.dat" (ya creado y con números en él) están ordenados de menor a mayor.

Sistemas Operativos – Estadísticas (web servers)

Operating Systems 2012			Operating Systems 2013		
1	Windows 7	39.47%	1	Windows 7	43.69%
2	Windows XP	29.38%	2	Windows XP	22.59%
3	Apple OS X	8.79%	3	iOS	9.50%
4	Windows Vista	7.77%	4	Apple OS X	8.45%
5	Apple iOS	5.25%	5	Windows Vista	5.28%
6	Android	1.78%	6	Android	4.19%
7	Linux	1.76%	7	Windows 8	2.72%
8	BlackBerry	0.57%	8	Linux	1.89%
9	SymbianOS	0.18%	9	BlackBerry	0.57%
10	Windows 8	0.08%	10	SymbianOS	0.26%

http://www.w3counter.com/globalstats.php

Continuará