Sinfo

1.04

Juan M. Aguirregabiria

Fisika Teorikoa. Zientzi Fakultatea
Euskal Herriko Unibertsitatea
P.K. 644, 48080 Bilbo (Spain)
Internet: `wtpagagj@lg.ehu.es`

August 1994

Copyright

The program **SINFO.EXE**, the help file **SINFO.HLP**, the documentation file **SINFO.WRI** and all the remaining files in this package are copyrighted:

Sinfo is FreeWare; there is no charge for using it and it may be distributed freely so long as the files are kept together and unaltered. You may neither sell nor profit from distribution of *Sinfo* in any way.

Disclaimer

In no event will the Author be liable to users for any damages, including but not limited to any lost profits, lost savings or other incidental or consequential damages arising out of the use or the inability to use this program, even if the Author has been advised of the possibility of such damages, or for any claim by other party.

Hardware and software requirements

To run *Sinfo* you need Microsoft Windows 3.1 and the hardware necessary to run it.

New in this version

New in version 1.04

New features:

When capturing a screen piece, the display of the box dimensions is now optional.

New in version 1.03

New features:

Using **File/Capture** one can now send a piece of screen to a new window where it can be seen at different magnifications. From that window you can save the captured rectangle to the clipboard or to a disk file.

New in version 1.02

New features:

A new menu entry, **Manager/Move...,** and a new button, **Move...** in the **Windows** dialog, allow moving, resizing and changing the z-order of a window.
Changing the window size is now smother.
A more reliable method is used to identify child and owned windows.

New in version 1.01

Corrected bugs:

In graphics modes with more than 256 colors it did not capture the DIB bitmap. (This bug was due to a bug in an example from Microsoft!)

New features:

A new entry, **File/Show cursor**, allows recovering a cursor that was lost due to an application crash.
A new entry, **File/Exit Windows**, allows exiting Windows even if the shell (*Program Manager* or alternative porogram) has crashed.

Other programs from the same author

Toolbar:

Toolbar is intended to provide some features that I found lacking in the *Program Manager* and *File Manager* of the retail Windows 3.1. To start with, while working in *File Manager*, you can drag an item and drop it over some running programs. But if one of these programs is not running and you drop the item over its icon, the program is not started. Instead, the dropped item is added to the corresponding group of *Program manager*. Of course, sometimes this is just what you wanted. One could say that the groups in *Program manager* were meant more for organization of programs than for running them. On the other hand, most programs still lack the ability to accept dropped files and you cannot use with them this drag and drop feature.

Since my screen tends to be cluttered with windows, I also wanted the ability to place (in a permanent and compact way) at a corner of the screen the utilities I used most often. *Program Manager* groups are nice but big and I do not need to see the names of my favorite utilities: I can easily recognize them by their icons. The utilities should always be at hand - ready to be started by double-clicking them (as in *Program Manager*) or by dropping an item on them. For instance, this is a nice way to provide the most important facility missing in *File Manager*: a file viewer.

Toolbar is the answer to these wishes. It provides an alternative to the groups of *Program Manager* with the following distinctive characteristics:

It can be used either as a substitute for *Program Manager* (which, in turn, can be run from *Toolbar*), or as a normal application installed in *Program Manager* (or in the current shell program).

It is better suited for running programs: there are multiple ways of running programs and advanced settings can be used.

All programs in a toolbar are always visible (as long as the whole window is on the screen). They are held together in a very compact way, to be as unobtrusive as possible. Optionally, you can force a toolbar to be always visible on top of other windows. In order to save space, the title bar can be suppressed (or displayed to have useful information at hand).

Items from *File Manager* can be dropped on any program installed as a tool. It only needs to be able to accept file names on the command line. Most MS-DOS and Windows programs have this ability. A program can also be started by double-clicking (or single clicking the right button), or by using the enter key, or a user selected accelerator key.

You can choose between running an instance of the program for each dropped file, or running a single instance of the program for all of the dropped files.

Programs can be easily added, removed and reordered. The corresponding caption, command line, working directory, icon and other options can be easily changed.

You can implement a full tree of nested toolbars.

You can drag files to and from *Toolbar*. *Program Manager* does not allow dragging of its elements outside of its windows.

Utilities (small utility programs that use a toolbar button as its display window) make *Toolbar* more compact and extensible. A few are provided with *Toolbar*: trashcan, alarm clock, menu, memory, resources, information and blank icon. Additionally, new utilities can be added by the user (the programmatic interface is fully documented and sample source code is included).

Toolbar supports most of the same user interface as *Program Manager.*

Toolbar is able to transparently use (and to translate to its own format) *group* files from *Program Manager.*

And more!

Filter:

Many DOS utilities and programs share a common characteristic: they are command line programs. Once the command line is supplied, they can run unattended, without any further input from the user. Often they are “filters”: they receive their input from the standard input, which can be the keyboard, a file redirected with the < operator or another program’s output (through the | pipe operator or using a TEE or Y command). Their output is usually sent to the standard output, which can be the default screen or redirected (with >, >>, |, TEE or Y). For example, every DOS copy comes with SORT and FIND, and there lots of free or shareware programs that emulate well know Unix filters: AWK, CAT, CMP, CUT, DETAB, DIFF, ENTAB, GREP, LS, MERGE, TAIL,

UNIQ, WC, etc.
Of course, one can always open a DOS box to take advantage of these programs from Windows, but often it would be nice to use Windows features: multiple windows, clipboard etc. This is especially true when one is trying a new command and want to see how it works. Often rather than using files, it would be nice to be able to enter the input in a window, run the program and see the output in another window.
Filter is a Windows shell that makes easier entering and trying external DOS (or Windows) programs. The input and output (as well as error log and program file) can be external files (which can be selected from the common dialog box used by most Windows programs) or windows. It is easy to specify the starting directory and other options for the external program and it can be run minimized or even hidden. Different configuration files can be used to take advantage of Filter with different external programs: the settings more appropriate to each program will be remembered in the next session.

Index

Introduction

Sinfo is designed to help Windows programmers by providing information on many aspects of Windows. It can display full information on current classes, windows, modules and tasks. There are some advanced (though highly risky) commands to unload modules and libraries and to free leaked memory. You can also examine and save to a file the contents (or the hex dump) of any global or local heap block. It is possible to see (and terminate) hidden tasks.

It can also be used as an advanced substitute for the *Task Manager* (in this way it can be accesed very quickly). Hidden tasks can be optionally listed and you can move, resize, change the z-order, hide (and shown again) windows and make them a top most window (or delete this property).

For instance, some **DLL** libraries cannot be simultaneously used for more than one program. If the program crashes, the library may be left in memory and cannot be reused without restarting Windows. The knowledgeable user can use *Sinfo* to free it. This possibility is especially useful when developing programs, but it has also helped us with a very known commercial package which crashes repeatedly leaving a library in memory.

Using *Sinfo* instead of *Task Manager* has other advantages. When developping programs (or testing other people's applications), it is not difficult to get the cursor lost. *Sinfo* can recover it for you. It is not unseen that the shell (*Program Manager* or alternative program) crashes leaving you with a blank desktop and no way to exit Windows exccpt restarting the computer. *Sinfo* lets you exit Windows in the right way.

The way in which information is obtained is described in this document/help file. To understand fully this information, a fairly knowledge of Windows programming is required (the SDK or equivalent documentation will be helpful).

Procedures

Installing *Sinfo*
Using *Sinfo* instead of *Task Manager*

Installing *Sinfo*

To install *Sinfo*, copy to the desired directory the executable file, **`SINFO.EXE`**, the help file **`SINFO.HLP`** and, optionally, the documentation file: **`SINFO.WRI`**.

Using *Sinfo* instead of *Task Manager*

To use *Sinfo* instead of *Task Manager* (in this way you can access it by double-clicking at the desktop), edit with a text editor (such as *Notepad*) the **`system.ini`** file in your Windows directory and change or add a line in the form:

```
taskmanager.exe=c:\sinfo\sinfo.exe
```

(Include there the actual full file specification of your copy of *Sinfo*).
Then *Sinfo* will be accessible by double clicking at the desktop. See **Manager**.

Commands

Menu	Entry	Accelerator key
File	**Save...**	f2
	Capture...	ctrl+a
	Show cursor	
	Exit	
	Exit Windows	
New in this version		
Acknowledgments		
Classes	**by Name**	ctrl+c
	by Module	**by Module handle**
	Unsorted	
	Pick a window	
Windows	**by Title**	ctrl+w
	by Handle	
	by Task	
	by Module	
	by Parent	
	Unsorted	
	Pick a window	ctrl+p
	Show children	
	Show buttons	
	Tree	ctrl+r
Modules	**by Name**	ctrl+d
	by Handle	
	by Task	
	by Exe file	
	Unsorted	
	Pick a window	
Tasks	**by Handle**	ctrl+t
	by Module handle	
	by Module name	
	Unsorted	
	Pick a window	
Memory	**by Module**	ctrl+e
	by Size	
	by Non-discardable size	
	Unsorted	**Summary...**
Manager	**Task manager**	ctrl+n
	Show hidden	
	Cascade	
	Tile horizontal	**Tile vertical**
	Arrange icons	
	Close	
	Minimized	
	Hidden	
	On top	
	Move...	

Window...
Terminate
Help **Index** f1
Commands
Procedures
Messages
Help on help
About Sinfo...

File

This menu has the following entries:

Entry	Accelerator	Meaning
Save...	f2	Save the contents of the list box to a file
Capture...	ctrl+a	Open a **Capture screen** dialog
Show cursor		Recover the cursor lost due to an application crash.
Exit		Exit *Sinfo*
Exit Windows		Exit Windows, after confirmation.

Show cursor is one of the useful but risky features in *Sinfo*. Do not use it except if afert an application crash the cursor is not visible in any window. Since it sets to zero the cursor count, do not use it if you have no mouse. If the cursor is already visible, you will have to wait for a couple of seconds (or a bit longer in slow machines) to recover the cursor. If you use *Sinfo* as task manager, you can invoke it by double clicking at the desktop even if the cursor is not visible. You can also use the keyboard to run it from the **Execute** entry of *Program Manager* (or the shell you use).

Exit Windows may be helpful if the shell (*Program Manager* or alternative program) crashes leaving you with a blank desktop and no way to exit Windows exccpt restarting the computer. If *Sinfo* is used instead of *Task Manager*, simply double click at the desktop and this entry lets you exit Windows in the right way.

Classes

If one of the following entries in this menu is selected:

by Name (ctrl+c)
by Module
by Module handle
Unsorted

the currently registered classes are listed (by using the **`ClassFirst`** and **`ClassNext`** functions of **`TOOLHELP.DLL`**) in the corresponding order as lines in the form:

Module hMod Name

where:

Module is the name of the module owning the class.
*hMod*is the handle of the module owning the class.
Name is the name of the class.

If a line is selected by using enter or a mouse double click, a **Class info** dialog box will open to provide more information.

Pick a window If this entry is selected, the cursor changes and you can select a

window by moving the mouse and using its left button (esc will abort the selection). The currently selected window will have a distinctive frame. A **Class info** dialog box will open to provide information on the class of the selected window. Child windows (as determined by IsAChild) may be picked with the mouse only if **Show children** in the next menu is checked.

Windows

If one of the following entries in this menu is selected:

by Title (ctrl+w)
by Handle
by Task
by Module
by Parent
Unsorted

the current windows are listed in the corresponding order as lines in the form:

Hwnd Task Module Par. Title

where:

Hwnd is the window handle.
Task is the handle of the task owning the window.
Module is the name of the module that has registered the window.
Par. is the handle of the parent window (or 0000, if none).
Title is the window caption (if any).

If a line is selected by using enter or a mouse double click, a **Window info** dialog box will open to provide more information.
In the listing the desktop window (given by **GetDesktopWindow**) is always listed, as well as all the parent windows given by **EnumWindows**. There is an additional entry in this menu:

Pick a window If this entry is selected, the cursor changes and you can select a window by moving the mouse and using its left button (esc will abort the selection). The currently selected window will have a distinctive frame. A **Window info** dialog box will open to provide information on the selected window.

Show children If this entry is checked, the child windows (as determined by IsAChild) will be included in the listing and can be picked with the mouse.

Show buttons If this entry is checked, the **Task manager buttons** will be displayed in the bottom of the screen.

Tree (ctrl+w) If this entry is selected, the full tree of windows (including child windows even if is not selected) is traversed with **GetWindow(hwnd,GW_CHILD)** and **GetWindow(hwnd,GW_HWNDNEXT)** and the found windows are displayed as described above. The *Title* field will include and identation that show the level of each window and its relation with other windows. This is the initial display when *Sinfo* is not the current task manager.

Modules

If one of the following entries in this menu is selected:

by Name (ctrl+d)
by Handle
by Exe file
Unsorted

the currently loaded modules are listed (by using the **`ModuleFirst`** and **`ModuleNext`** functions of **`TOOLHELP.DLL`**) in the corresponding order as lines in the form:

Name hMod EXE file

where:

Name is the module name.
*hMod*is the module handle.
EXE file is the name of the executable file from which the module was loaded.

If a line is selected by using enter or a mouse double click, a **Module info** dialog box will open to provide more information.

Pick a window If this entry is selected, the cursor changes and you can select a window by moving the mouse and using its left button (esc will abort the selection). The currently selected window will have a distinctive frame. A **Module info** dialog box will open to provide information on the module owning the selected window. Child windows cannot be selected in this way because they all share parent's module.

Tasks

If one of the following entries in this menu is selected:

by Handle (ctrl+t)
by Module handle
by Module name
Unsorted

the currently running tasks are listed (by using the **`ModuleFirst`** and **`ModuleNext`** functions of **`TOOLHELP.DLL`**) in the corresponding order as lines in the form:

Task hMod Module

where:

Task is the task handle.
*hMod*is the handle of the currently executing module.
Module is the name of the currently executing module.

If a line is selected by using enter or a mouse double click, a **Task info** dialog box will open to provide more information.

Pick a window If this entry is selected, the cursor changes and you can select a window by moving the mouse and using its left button (esc will abort the selection). The currently selected window will have a distinctive frame. A **Task info** dialog box will open to provide information on the task owning the selected window. Child windows cannot be selected in this way because they all share parent's task.

Memory

If one of the following entries in this menu is selected:

by Module (ctrl+e)
by Size
by Non-discardable size
Unsorted

the blocks in the global heap (as returned by **GlobalInfo** and the **GLOBALENTRY ge** structure returned by **GlobalFirst** and **GlobalNext**) are listed, grouped by module, in the corresponding order as lines in the form:

Module Handle Total Size (Blocks) Non-Discardable (Blocks) (Handle type)

where:

Module is the name of the module that has allocated the blocks.
Handle the owner handle (or the number of owners between parenthesis).
Total size is the total size in bytes of the group of blocks.
Blocks is the number of block in the previous group.
Non-... is the size of non-discardable blocks.
Blocks is the number of block in the previous group.
Handle type the type (**Task** or **Module**) of *Handle* or blank if more than one owner.

If the **wType** entry of the block **GLOBALENTRY** structure is **GT_FREE** the corresponding *Module* will be declared **<Free>**. If it is **GT_INTERNAL**, **GT_SENTINEL** or **GT_BURGERMASTER**, it will be called **<System>**. In all other cases **TaskFindHandle** will be used to find the *Module*.
A block is non-discardable if **GlobalFlags** returns neither **GMEM_DISCARDABLE** nor **GMEM_DISCARDED**.
Handle is the owner handle in the **hOwner** entry of **ge**, or (if more than one) the number of different owner between parenthesis.

If a line corresponding to more than one owner (the *Handle* entry between parenthesis) is selected (by using enter or a mouse double click), the list box will be redisplayed to provide information for each individual task/module (or entry if one task/module exists). Selecting again a line (or if the previous line corresponds to a single module/task) will redisplay the list box with lines displaying the information in the **GLOBALENTRY ge** structure for each global handle in the form:

Address Size Handle Lock (Page) (LDEP) Type

where:

Address The block linear address in the **dwAddress** field.
Size The block size in bytes (**dwBlockSize** field).
Handle The handle of the block.
Lock The lock count in the **wcLock** field.
Page The page lock count in the **wcPageLock** field.
L Is **Y** if the block contains a local heap (**wHeapPresent** field).
D Is **Y** if the block is **GMEM_DISCARDABLE**, according to **GlobalFlags**.
E Is **Y** if the block is **GMEM_DISCARDED**, as returned by **GlobalFlags**.

P Is **Y** if the block has the attribute **GF_PDB_OWNER** in the **wFlags** field.

Type The type of block described by **wType** and **wData**.

If a line corresponding to a local heap is selected (by using enter or a mouse double click), the list box will be redisplayed to provide information for each individual local handle as returned in a **LOCALENTRY le** structure. Each line will look as follows:

Address Size Handle Lock (TDEFM) Type

where:

Address The block address in the **wAddress** field.

Size The block size in bytes (**wSize** field).

Handle The handle of the block.

Lock The lock count in the **wcLock** field.

T Is **U** if **wHeapType** is **USER_HEAP** and **G** if it is **GDI_HEAP**.

D Is **Y** if the block is **GMEM_DISCARDABLE**, according to **LocalFlags**.

E Is **Y** if the block is **GMEM_DISCARDED**, as returned by **LocalFlags**.

F Is **Y** if the block has the attribute **LF_FIXED** in the **wFlags** field.

M Is **Y** if the block has the attribute **LF_MOVEABLE** in the wFlags field.

Type The type of block described by **wType**.

If a line of a local heap list or of a global list (if it does not contain a local heap) is selected (by using enter or a mouse double click), a **Dump window** will open to show its contents.

There is an additional entry in this menu:

Summary... If this entry is selected a **Memory summary** will open.

Manager

Task manager (ctrl+n) If this entry is selected the titles of the main windows of the current tasks are listed. By double clicking at a line (or pressing the button entitled **Switch to**, see **buttons**) you can select it as the new active task.

Show hidden If this entry is checked, the list will include hidden windows.

The following four entries let you arrange the windows (and icons) on the screen:

Cascade
Tile horizontal
Tile vertical
Arrange icons

If the list box displays windows (because **Windows** or **Task manager** has been used or because *Sinfo* is being used instead of *Task Manager*) and a window has been seleced in the list box, the following entries are available (they are duplicated in **buttons** if **Show buttons** is enabled):

Close Close the application by sending a **WM_CLOSE** message to the

window.
Minimize If this entry is checked/unchecked the window will be iconized/restored.
Hidden If this entry is checked/unchecked the window will be hidden/shown.
On top Only if this entry is checked will have the window the top most flag set.
Move... The **Move window** dialog box will.
Window... The **Window info** dialog box will open.
Terminate After a warning, the dangerous **`TerminateApp`** function will be called.

Help

From this menu the Windows help system is accessed:

Menu entry	Help topic
Index	Help index (it can also be accessed by using `f1`)
Commands	Commands in the menu system
Procedures	How to install the program, etc.
Messages	Meaning of error and warning messages
Help on help	How to use the help system
About Sinfo...	Version and Copyright information in a dialog box.

Dialog boxes

The following dialog boxes may appear in *Sinfo*:

Class info
Window info
Module info
Task info
Memory summary
About Sinfo... version and Copyright information.
Save
Capture screen
Move window

Class info dialog box

This dialog box appears when the main list box contains **Classes** an one of its lines is selected (by using `enter` or a mouse double click) or a window is selected with the mouse after using **Class/Pick a window**.
Apart from the self-explaining **OK** and **Help** buttons, the following entries contain information (obtained from the **`WNDCLASS`** **`wc`** structure returned by **`GetClassInfo`**) on the selected class (if it has not been unregistered in the meanwhile):
Class name the class name in **`wc.lpszClassName`**
Module name the owner module name obtained from **`wc.hInstance`** and **`ModuleFindHandle`**.

Module instance the owner module handle obtained from **wc.hInstance**.
Menu name the class menu name (or number in the form **#n**) from **wc.lpszMenuName**.
Class extra bytes the number of extra bytes per class from **wc.cbClsExtra**.
Window proc the window procedure address in hex format ***hhhh:hhhh*** from **wc.lpfnWndProc**.
Window extra bytes the number of extra bytes per window from **wc.cbWndExtra**.
Icon (*hhhh*) the class icon handle (if **0000** if none). The icon is displayed next to its handle.
Brush (*hhhh*) the class background brush handle (if **0000** if none). It is used to display the class icon and cursor next to their handles.
Cursor (*hhhh*) the class cursor handle (if **0000** if none). The cursor is displayed next to its handle.
Style (*hhhh*) the class style in hexadecimal format. The different styles are displayed by means of the corresponding **#define**s from **WINDOWS.H**.

Window info dialog box

This dialog box appears when the main list box contains **Windows** an one of its lines is selected (by using enter or a mouse double click) or a window is selected with the mouse after using **Window/Pick a window**.
Apart from the self-explaining **OK** and **Help** buttons, the following entries contain information on the selected window (if it has not been destroyed in the meanwhile):
Title The window caption as returned by **GetWindowText**.
Window handle The window handle.
Parent window The parent window handle as returned by **GetParent**.
Class name The window class name as returned by **GetClassName.**
Module The owner module as obtained from **GetWindowInstance** and **GetInstanceModule**.
Instance handle The window instance handle as obtained from **GetWindowInstance**.
Task handle The window task handle as obtained from **GetWindowTask**.
Id / Menu The menu or child identifier as obtained from **GetWindowWord(hwnd,GWW_ID)**.
Procedure The window procedure address in hex format ***hhhh:hhhh*** from **GetWindowLong(hwnd,GWL_WNDPROC)**.
Width The client rectangle width from **GetClientRect**.
(left,top) The window left and top coordinates from **GetWindowRect**.
Height The client rectangle height from **GetClientRect**.
(rigth,bottom) The window right and bottom coordinates from **GetWindowRect**.
Style: *hhhhhhhh* (Extra: *hhhhhhhh*) the window style and extra style from **GetWindowStyle** and **GetWindowExStyle**. The most common individual styles are also listed by means of the corresponding **#define**s from **WINDOWS.H**.
Properties The list of property handles and strings in the form ***hhhh=string***

from **EnumProps**.

The following buttons are also available:

Switch to Make this the active window.
Close Close the window by sendign a **WM_CLOSE** message.
Minimize/Restore the window.
Hide/Show the window.
On top/Not on top Set/reset this flag.
Move... Open the **Move window** dialog box.
Class... Open the **Class info** dialog box.
Module... Open the **Module info** dialog box.
Task... Open the **Task info** dialog box.
Terminate After a warning, the **TerminateApp** function will be called.

Module info dialog box

This dialog box appears when the main list box contains **Modules** an one of its lines is selected (by using enter or a mouse double click) or a window is selected with the mouse after using **Module/Pick a window**.
Apart from the self-explaining **OK** and **Help** buttons, the following entries contain information (obtained from the **MODULEENTRY me** structure returned by **ModuleFindHandle**) on the selected module (if it has not been unloaded in the meanwhile):

Module name The module name from **me.szModule**.
Module handle The module handle.
Usage count The reference count from **me.wcUsage**.
EXE file name The corresponding executable file specification from **me.szExePath**.
Free module After a warning, this button calls **me.wcUsage** times the **FreeModule** function using the module handle. Very dangerous!!!
Free library After a warning, this button calls **me.wcUsage** times the **FreeLibrary** function using the module handle. Very dangerous!!!

Task info dialog box

This dialog box appears when the main list box contains **Tasks** an one of its lines is selected (by using enter or a mouse double click) or a window is selected with the mouse after using **Task/Pick a window**.
Apart from the self-explaining **OK** and **Help** buttons, the following entries contain information (obtained from the **TASKENTRY te** structure returned by **TaskFindHandle**) on the selected task (if it has not ended in the meanwhile):

Task handle The task handle.
Instance (DGROUP) The instance handle form **te.hInst**.
Module name The currently executing module form **te.hModule** and **ModuleFromHandle**.
Module instance The currently executing module handle form **te.hModule**.
Parent task The parent task handle (if any) from **te.hTaskParent**.
Parent name The parent task name from **te.hTaskParent** and **TaskFromHandle**.

Command line The task command line from offset **0x81** of the PSP segment in **te.wPSPOffset**.
Working directory The task current directory (as described at offset 66h of the segment given by the task handle).
PSP Offset The PSP segment in **te.wPSPOffset**.
Queue handle The message queue handle form **te.hQueue**.
CS:IP The execute address returned by **TaskGetCSIP**.
Pending events The number of events in queue from **te.wcEvents**.
SS:SP The stack top address from **te.wSS** and **te.wSP**.
Stack top The stack top address from **te.wStackTop**.
Stack minimum The stack lowest position from **te.wStackMinimum**.
Stack bottom The stack bottom address from **te.wStackBottom**.
Show module Open the corresponding **Module info** dialog box.
Terminate After a warning, this button calls the **TerminateApp** function using the task handle. Very dangerous!!!

Memory summary dialog box

This dialog box appears when the **Summary** entry in the **Memory** menu entry is selected.
Apart from the self-explaining **OK** and **Help** buttons, the following entries contain the following information:

Virtual memory information (as returned by **MemManInfo** in the **MEMMANINFO mi** structure):
Largest free block The number of items in the global heap (**dwLargestFreeBlock** entry).
Maximum available pages The number of free items in the global heap (**dwMaxPagesAvailable** entry).
Maximum lockable pages The number of items in the global heap (**dwMaxPagesLockable** entry).
Max pages in linear space The total linear space in pages (**dwTotalLinearSpace** entry).
Unlocked and free pages The unlocked and pages (**dwTotalUnlockedPages** entry).
Free pages The number of pages not in use (**dwFreePages** entry).
Total pages The total number of pages in the system (**dwTotalPages** entry).
Free pages in linear space The number of pages in linear space not in use (**dwFreeLinearSpace** entry).
Pages in swap file The number of pages in the swap file (**dwSwapFilePages** entry).
Page size The system page in bytes (**wPageSize** entry).

Global heap items information (as returned by **GlobalInfo** in the **GLOBALINFO gi** structure):
Total The total number of items (**wcItems** entry).

Free The total number of free items (**wcItemsFree** entry).
Least recently used The total number of *least recently used* items (**wcItemsLRU** entry).

Save dialog box

This dialog box appears when the **Save** or **Capture** entries of the **File** window or the **Save** or **Copy** entries of a **Dump window** are used. It lets you choose the file where the output will be placed. Since the dialog box is invoked by *Sinfo* but provided by Windows you should refer to Windows documentation for more information on this common dialog.

Capture screen dialog box

This dialog box appears when the **Capture** entry of the **File** window or ctrl+a are used.
Apart from the self-explaining **OK**, **Cancel** and **Help** buttons, the following entries contain the following information:

Screen area to capture This lets you specify a screen rectangle to be captured:
Select with mouse You will select the dimensions of the rectangle by using the mouse.
Rectangle You select the dimensions of the rectangle in **Dimensions** which has two entries:
Width The rectangle withd (in pixels).
Height The rectangle height (in pixels).
Icon A rectangle with the dimensions of a icon will be captured.
After selecting the rectangle dimensions by means of one of the previous entries, you must select with the mouse its exact position. esc abort the selection.
Client area of a window You select with the mouse the window whose client area is to be captured. Child windows can be selected only if **Show children** is enabled.
Entire window You select the window to be captured. Child windows can be selected only if **Show children** is enabled.The whole window (including caption, etc.) will be captured.
Full screen The whole screen will be captured.

Show dimensions If this entry is checked the starting and end points of the rectangle will be automatically displayed.

Copy to Select here where should be stored the captured window:
New window A new **Capture window** will open to show the window.
Clipboard The device dependent bitmap, the device independent bitmap and the palette will be sent to the clipborad.
File The device independent bitmap will be stored in the **.BMP** file that you select in a **Save dialog box**.

Move window dialog box

This dialog box appears when the **Move** entry of the **Manager** menu or the **Move** button of the **Window info** dialog box are used. The window whose handle appears in the caption will be moved, sized or ordered by means of a **SetWindowPos** call.

Apart from the self-explaining **OK**, **Cancel** and **Help** buttons, the following entries can be used:

Move If this box is checked, the window will be moved to the position (relative to the parent window, if any) entered in the following two entries:
Left The new position of the left of the window.
Top The new position of the top of the window.
Size If this box is checked, the window will be resized according to the following two entries:
Width The new width of the window.
Height The new height of the window.
Order If this box is checked, the window will be placed in the Z-order according to the following entry:
After The window that will preceed the current one. This parameter must be a window handle or one of the following values:
0 Places the window at the top of the Z-order.
1 Places the window at the bottom of the Z-order. If the window was a topmost window, it loses this status.
-1 (FFFF) Places the window above all windows and gains the topmost status.
-2 (FFFE) Repositions the window to the top of all non-topmost windows (that is, behind all topmost windows). This flag has no effect if the window is already a non-topmost window.
Redraw If this flag is not set, the window will not be redrawn and strange things may happen.
Activate The window is activated and moved to the top of either the topmost or non-topmost group.
Draw frame Draws a frame (defined in the window's class description) around the window.

Dump window

If a line of a local heap list or of a global list (if it does not contain a local heap) is selected (by using enter or a mouse double click), a window will open to show its contents. The caption will display the handle, address and size and the client window will display a dump in the form of lines with the following structure:
1. The hexadecimal offset from the block start of the first byte displayed in this line.
2. 16 bytes in hexadecimal two-digit format.
3. The same 16 bytes as ANSI characters (a blank is substituted for the null character) between vertical bars. With the **File/ASCII only** option enabled, only printable ASCII characters (32 to 126)are displayed.
You can move around with the scroll bars (or the equivalent keyboard commands).

A maximum of 32767 bytes will be dumped to the screen. But the full block can be dumped to a file by using **File/Save** (in which case better results are obtained by using the **File/ASCII only** option).
The window has a menu with the following entries:

Menu	Entry	Meaning
File	**ASCII only**	Dump only printable ASCII characters
	Save	Save the full dump to a file
	Copy	Copy the block to a file
	Close	Close the window
Free	**Free memory handle**	See below
Help	**Help**	Show this help screen

The **Free memory handle** entry is available only for global handles and let you free the corresponding handle by repeatedly calling the **`GlobalUnlock`** and **`GlobalFree`** functions. As a warning will remind you, this is a powerful but very dangerous feature. Use it at your own risk.

Capture window

This kind of window will open when choosing **New window** in the **Capture screen** dialog. It will show at different magnification scale the piece of screen just captured.
The window has a menu with the following entries:

Menu	Entry	Meaning
File	**Save**	See below
	Copy to the clipboard	Copy the displayed bitmap to the clipboard
	Close	Close the window
Scale	**x1**	Set the magnification
	x2	
	x4	
	x8	
	x16	
	x32	
Help	**Help**	Show this help screen

By using the **Save** command you can send the captured piece of screen to a **`.BMP`** file that you select in a **Save dialog box**.

Task manager buttons

If the **Show buttons** entry of the **Windows** menu is enabled, the following buttons will appear in the main window when :

Switch to	The selected window/task will be activated.
Close	The selected window/task will receive a **`WM_CLOSE`** message.
Minimize/Restore	the current window.
Hide/Show	the current window.
On top/Not on top	Set/reset this flag to the current window.
Window...	Open the corresponding **Window info** dialog box.

Class... Open the corresponding **Class info** dialog box.
Module... Open the corresponding **Module info** dialog box.
Task... Open the corresponding **Task info** dialog box.
Terminate After a warning, this button calls the **`TerminateApp`** function using the task handle. Very dangerous!!!
They will be enabled only if the list box displays windows (because **Windows** or **Task manager** has been used or because *Sinfo* is being used instead of *Task Manager*) and a window has been seleced in the list box.

Error and warning messages

The following messages are issued by *Sinfo* when necessary:

`Class no longer exists`
The class has been unregistered and is no longer available.

`Could not create a window`
This improbable error indicates that it was impossible to register or create a window.

`Could not create the file`
Maybe the disk was full.

`File already exists`
`Do you want to overwrite it?`
`(Answer NO to append to the end)`
If you answer **Yes** the file will be truncated and overwritten. If you answer **No**, the output will be append to the end of the file. Using **Cancel** aborts the output operation.

`Global handle is no longer valid`
The handle has been freed.

`Invalid global handle`
The block contents cannot be displayed. It may be void or free.

`Local handle is no longer valid`
The handle has been freed.

`Modifying a child window is very unsafe`
`Do you want to proceed?`
If you ignore this warning, unexpected consequences may arise.

`Modifying the desktop window could lock Windows`
`Do you want to proceed?`
If you ignore this warning, well, you'll see what happens.

`Module no longer exists`
The module has been unloaded and is no longer available.

Not enough memory
There is not enough memory to collect the information.

Task no longer exists
The task ended and is no longer available.

This is a very unsafe way
to free a handle
Do you want to proceed?
This warning lets you know that severe problems could arise if you try freeing the handle. Proceed under your exclusive responsibility.

This is a very unsafe way
to free a module/library
Do you want to proceed?
This warning lets you know that severe problems could arise if you try using the unload feature. Proceed under your exclusive responsibility.

This is not a safe way to end a task
Do you want to proceed?
This warning lets you know that severe problems could arise if you try using the terminate feature. Proceed under your exclusive responsibility.

ToolHelp has caused a GP fault
This message lets you know that a severe problem happened when using ToolHelp services to explore the global heap. Simply, try again.

Window no longer exists
The window has been destroyed and is no longer available.

Suggestions, bugs and help

The Author would highly appreciate receiving information about any bug or problem found in the program. Suggestions to improve *Sinfo* in future versions are also welcome. I would especially appreciate help to correct the English version of the help and documentation files.
Write to:

Juan M. Aguirregabiria
Fisika Teorikoa. Zientzi Fakultatea
Euskal Herriko Unibertsitatea
P.K. 644, 48080 Bilbo (Spain)
Internet address: **wtpagagj@lg.ehu.es**

Glossary

Handle

An hexadecimal number (from 0000 to FFFF) identifying a module, a task, a window, etc.

#n

An number preceded by **#** to indicate a resource loaded by using **MAKEINTRESOURCE**.

#IsAChild

To decide if a window ia a child or if it is owned, *Sinfo* now uses the following check:

```
BOOL IsAChild(HWND hwnd)
{
 return GetWindow(hwnd,GW_OWNER) ||
        GetParent(hwnd) ||
        (GetWindowStyle(hwnd) & WS_CHILD);
}
```