Contents

Introduction

Welcome to Scout, a programmers utility program.

The first thing you notice is an unassuming Main Window. It may not look like much, but it's the entrance to some very powerful features. Scout was designed to be as unobtrusive as possible.

Although Scout uses fault trapping, where appropriate, to protect you from crashes, it is not fool proof. There is little I can do to prevent System Crashes with the operating system if you use Scout carelessly. In other words, its up to you to avoid General Protection (GP) faults when using the of Scout's advance capabilities.

For context sensitive HELP, press the F1 key. Scout will bring up help for the dialog that has the focus (except the help interceptor dialog).

Scout can:

- Display free system resources
- List all currently loaded modules or running tasks
- List all exported functions in a module
- Show all modules that are implicitly linked to a dll
- Call any exported function in any module with arguments that you supply
- Show where some fat might be in a module (liposuction)
- Start a new task or free a currently running task
- Show which modules are taking up memory below 1 MB
- Free a module
- Monitor system resources
- Show a list of all windows and send any standard window message to any window or group of windows
- Display system metrics
- Display detailed information on a window
- Intercept calls to WinHelp

System Requirements

Because Scout uses API calls that were introduced in Windows 3.1, Scout requires Windows 3.1 or later. Scout also makes extensive use of the Toolhelp and Ctrl3d dlls that are included with Windows.

System Menu

Scout adds three items to the system menu of its Main Window.

Always On Top'

This entry allows Scout to Stay on top of all other windows. When this feature is selected, a check mark will appear next to the menu item.

Options...

Brings up the Options dialog box to allow you to configure some of Scout's options.

Help Contents

Brings up Scout's Help file set to the Contents page.

For context sensitive HELP, press the F1 key. Scout will bring up help for the dialog that has the focus (except the help interceptor dialog).

About Scout:

From here you can display the Help about box which you can find Scout's version number. This way you can be assured of always having the latest version.

The Main Window

Scout's Main Window display shows the percentage of free system resources (User and GDI) and potential free memory below 1MB.

On the Main window are 6 very powerful buttons. It's from here that you will accessed the best features of Scout. The most powerful of these buttons is labeled exit. This button will close scout and all of it's child windows. But beware, exiting scout will leave you helpless to accessed it's features (unless of course you start it up again). Equally powerful is the Options button. This will allow you to customize several of Scout's features. The other four buttons (and much more interesting), allow you to directly access the Module List, Task List, Memory below 1MB and Win Walker dialogs.

See Introduction for a more detailed list of Scouts capabilities.

Free Memory below 1MB

Windows requires a certain amount of memory below 1MB for every application that you start. When this value is too low, Windows will display an error message saying that there is insufficient memory when you try to start an application. Many applications take up some of this precious memory. This is may not be a problem unless the memory is fixed (Windows can move memory above 1MB to make room if it was allocated with the moveable attribute). Scout will show you a listing of modules that are using fixed memory below 1MB.

From here you can not run these applications, rewrite them if they are yours, or get the Fix1MB program from the May 95 issue of Microsoft System Journal (which was the basis for this function in Scout).

Options

Certain features of Scout are user configurable and are saved in Scout.ini when you exit scout. Settings take effect when you exit the Options dialog by pressing the OK button. Your current settings will remain unchanged if you exit the Options dialog by pressing the Cancel button.

The options dialog box can be reached from the Options button on Scout's main window or System Menu.

Stay On Top - Makes Scout stay above all other windows even when they have the focus. This option can also be set from the system menu of Scout's Main Window

Save Position - Scout will start in the same position it you last exited. If this option is not checked, Scout will start in the position that was last saved.

Filter System - When showing memory in use below 1MB, Scout will filter out the standard window dlls from the listing (since you can't do much about them anyhow). To see what memory windows is using, uncheck this option.'

Sort by Path - When freeing multiple modules, it is often easier if they were sorted by directory rather than name. This option can also be set from the system menu of the Module List.

Intercept - Scout will intercept all WinHelp calls and popup a dialog box with information on the call.

Show Help Quit - When checked, Scout will pop up the help intercept dialog for WinHelp calls with the HELP_QUIT parameter. Normally, these calls are not of much interest when tracking down help ids. If this box is not checked, Scout will automatically pass these calls on.

This option has no effect unless the Help Calls **Intercept** option is selected

Auto Update - You can choose to have the Module List, Task List, DLL Link List or the list of memory in use below 1MB automatically updated when modules are loaded or unloaded.

Auto Minimize - This options allows the Win Walker Dialog to minimize automatically when you select the Mouser function.

Start w/ Mouser - The Mouser function will popup automatically when you select the Win button from Scout's Main Window.

Mouser Only - The Mouser function will popup automatically when you select the Win button from Scout's Main Window and will close the Win Walker dialog when you exit Mouser. The Win Walker dialog is automatically minimized when you select this option.

Links to a Module

Links works two different way depending on whether it is called from the Task List or the Module List.

From the Module List, Links will show all the modules that implicitly use the selected module. If the selected module was not implicitly loaded, Scout will report the number of load library calls.

From the Task List, Links will show all the modules that the selected task uses implicitly.

Freeing Modules

Free works by decrementing the usage count to zero. Free does not free any resources that the module may be using or memory that it may have allocated.

Free comes in handy when the program you are testing closes and does not free all of the modules it loaded, either as the result of a General Protection Fault or on it own doing. Multiple modules may be selected by holding down the shift key (to select a range) or the ctrl key (to select individual modules).

Often it is more convenient to sort the list by path rather than by name (especially when freeing multiple modules). This options can be accessed from either the system menu or from the Options dialog.

You are given the option to confirm the freeing of each or all of the selected modules. You can skip the freeing of any selected module or cancel freeing the selected modules remaining.

Although you can use the module dialog box to free a running program, the Task dialog is more appropriate for freeing a running program that you can not close.

See Freeing a Task

Module List

The Module List show the name of every module current running, it module handle (hModule), its usage count, and the file path from which the module is running.

Update will force a screen update of all running modules. Unlike the resource monitor, the module list is not automatically updated unless a module is freed or loaded through Scout.

Load will bring up a Common dialog box allowing you load a new module. Loading a module that is already running will increment its usage count by one. Similarly, loading a module that is already running, even if it had a different path and a different file name will only increment the usage count of the version that is running.

Free will completely free a module regardless of it's usage count. See Freeing Modules.

Links will show all the modules that implicitly loaded the selected module.

Liposuction, which shows you where the fat is in your program.

F(x) is undoubtedly the gateway to the most powerful feature (and potentially the most dangerous) in Scout. F(x) will bring up a list of all exported function in a module and allows you to call any one of them. See Exported Function List and Calling a function for more information.

The Cancel button will not only close the Module list dialog, but will also close all sub dialogs (Liposuction, Exported Function List, DLL Link List, Function Call and Memory display) opened through the Module List box.

Liposuction

Liposuction, based on code that appeared in the July 1993 issue of *Microsoft Systems Journal*, shows where the fat is in your program. The two largest sources of fat are an unnecessary large value for the segment alignment and leaving debug information in your executable. Liposuction will show where you may be able to squeeze a few bytes out of your Windows based EXE or DLL.

Liposuction can be accessed from either the Module List or Task List Windows.

Exported Function List

To see a list of all exported function in module, first select a module in the module list and then click the F(x) button. The list of exported functions will be sorted by ordinal number. To sort the list by the function name, click on the Hide Ordinal button.

To call a function, first select a function and then click on the CALL Function Button. This will bring up the Function Call dialog.

To close the Function List dialog box, select the cancel button. Clicking on the cancel button on the Module list will close both the Module List and the Exported Function List.

Calling a function

Function Call was based on the sample application CallFunct in the book *Undocumented Windows*. Scout provides a easy to use interface for this powerful ability.

Calling a function with the parameters you specify can be a way to test and validate a routine in your program. The Function Call feature in Scout provides a convenient method of calling exported functions without the need of writing a specific test application. To call non-exported functions or to set up a special condition in your program for testing, you could write an exported function that is only compiled during a debug build of your program. Scout could then be used to call this special exported function which in turns calls other functions within your program.

Scout gives you a unprecedented level of control in calling a function. It is your responsibility to use the correct Calling Convention and the correct number and type of arguments.

WARNING:

Calling a function with the wrong calling convention or the wrong type of arguments or order of arguments will corrupt the stack and eventually lead to a General Protection (GP) Fault (hopefully a relatively benign one). A GP fault can cause system instability and lead to a lost of data, particularly if you reboot, or in the case that the fault is so bad, your machine reboots by itself.

Scout does trap for General Protection (GP) Faults (similar to the way Microsoft's Dr. Watson Utility does) and will easily catch such errors as divide by zero. However, certain GP faults will corrupt the system beyond Scout's ability to trap for them, particularly a trap occurring in one of the dlls used by Windows itself.

Scout can also be used to explore programs other than your own to discover hidden or undocumented features. This assumes that you know the correct calling convention as well as the correct number and type of arguments or are willing to suffer the consequences of guessing wrong.

See Basic steps of calling a function

Basic steps of calling a function

WARNING:

Calling a function, even with the correct arguments can cause unpredictable results and may eventually lead to a GP fault. Further, calling a function through Scout may bypass the program's built in safeguards. This may cause the program to do something it would not normally do such as write to your hard disk or even reformat it.

The basic steps of calling a function are:

Select Calling Convention

Add Arguments by selecting Argument type and entering in it's value

Enter the (printf style) Output Mask

Click on the Call button.

Examine the return value and/or Memory buffer

Calling Conventions

Scout supports three types of calling conventions: CDECL, PASCAL, and Register.

The calling-convention determines which way arguments are passed to functions, are pushed on the stack, and whether the calling or called function removes the arguments from the stack. C functions can have a variable number of arguments, and in the C/C++ calling convention (CDECL), arguments are pushed on the stack from right to left (so that the first argument in the list is the last one pushed on the stack). Pascal and FORTRAN programs use the Pascal calling convention (PASCAL), where the number of arguments is fixed and arguments are pushed on the stack from left to right.

Also in C, the calling function must remove the arguments, whereas in Pascal and FORTRAN, the called function does this. The Pascal calling convention optimizes size somewhat, while the C/C++ calling convention allows the flexibility of using variable-number parameter lists.

Most window API calls use the Pascal calling convention since it is more efficient than the C calling convention. The C calling convention, however, allows for a variable number of arguments in a function call where in the Pascal calling convention, the number of arguments is fixed. For this reason, a few API calls, such as wsprintf, use the C calling convention.

Other functions (such as the Window API DOS3Call) are designed to be called from Assembly language routines. In these functions, the arguments are placed directly into the CPU's registers.

Register Calling Convention

Scout lets you call functions that expect its arguments to be placed in the CPU's registers before the call is made. When you select the Register Calling Convention, Scout automatically replaces the arguments section with a register section. Each of the 4 main registers AX, BX, CX, and DX are represented as well as their 8 bit counterparts AH, AL, BH, BL, CH, CL, DH, DL.

When a change is made in the full 16 bit register, its 8 bit counterparts will automatically be updated to show the change. The 16 bit registers are also automatically updated if a change occurs in its 8 bit counterpart.

Scout ignores extraneous digits and displays the registers in hexadecimal. However, a decimal value can be entered into the registers by omitting the 0x prefix, but will be converted to hex when it's value is updated. During the actual call, Scout uses the values displayed in the 16 bit registers.

To view the current contents of the registers or to view them after a call, select Memory Dump.

Adding and Deleting Arguments

When you have selected either the PASCAL or CDECL calling convention, you have the option to pass one or more arguments. Arguments can be one of the following basic types:

WORD

Char

Float

DWORD

2Kb static buffer

Scout does not support more complicated argument types such as structures, or pointers (other than the pointer to a string).

To add an argument, first select the argument type. Once selected, the cursor is automatically placed in the edit box where you can enter the argument's value. The exception is the 2 kb static buffer which you can not enter a value. Click on the Add button to place the argument type and it's value into the list box. Arguments are always added to the end of the list even if an item is selected in the middle of the list.

To delete a function argument from the listbox, first select the item you wish to delete and then click the Delete Button. Arguments can be deleted from any position in the list.

Memory Dump

The Memory Dump button will bring up a display of the 2k static buffer used for function calls and the following CPU's registers: AX, BX, CX, DX, SI, DI, DS, ES. This is useful for examining the contents of the 2k static buffer after a function call which modifies it.

Clicking on the Clear Buffer button will zero out the buffer.

Output Mask

The Output Mask is where you specify how the return from the function call is displayed. The mask can be any valid printf() mask. If no mask is specified, the default "%i" is used.

Task List

The Task list shows the name of every task currently running, its task handle (hTask), parent handle (hParent), instance handle (hInst), stack size, and the file path from which the it is running.

Update will force a screen update of all running tasks. Unlike the resource monitor, the task list is not automatically updated unless a task is freed or loaded through Scout.

Load will bring up a Common dialog box allowing you start a new task. This is the same as starting a program from the File | Run menu option of Windows.

Free allows you to close down an application. When you click on the FREE button, a dialog box will pop up giving you a choice of two methods to free the task.

See Freeing a Task.

Links will show all the modules that the selected task links to. This is the opposite of what the links function does from the module window.

Liposuction shows you where the fat is in your program as discussed earlier. The Liposuction data is only for the main module. To display Liposuction data for other modules making up the task, select Liposuction for each module from the Module List.

Freeing a Task

When you choose FREE from the task list, you are given a choice of two methods to free a task: Close Task or Terminate Task.

Close Task sends a WM_CLOSE message to all of the windows in the task. This allows an application to close itself down and release any memory it may have allocated and resources it may have loaded.

Terminate Task ends the application as if the given task had produced a general-protection (GP) fault. Terminating a task may not free all objects owned by the ended application by will usually shut down a task that will not shut itself down.

Win Walker

Win Walker is a super spy utility that shows detailed information about selected windows. Unlike the spy utility that comes with the Windows' SDK, Win Walker can show information on any control in the window. While Microsoft's Spy is primarily a message spy utility, Win Walker primary task is gathering information that is useful for writing a test program.

You can walk the list of windows at the current level by using the arrow push buttons. Win Walker always starts at the first window in the chain. Trying to move past the end of the chain in either direction will result in a beep. To move up the chain, use the Parent button. To walk the chain of child windows for a given window, click on the Child button. Alternately, a window can be selected by using the Mouser which is described below.

From the Win Walker Dialog you can display various System Metrics or send a message (Send/Post message) to any window. Mail will also display a list of all windows currently running.

Mouser

The Mouser is a quick way to find information about a window or control. From the Win Walker Dialog, click on the mouser button. The Win Walker Dialog will minimize to an icon and the Mouser dialog will appear. (See Options to modify the starting of Mouser).

When you move the mouse, the window under the mouse cursor will have it's border change.

The Mouser dialog box will show some basic information about the window underneath the cursor including: its handle and title, Control Id Number, Module Name, Class Name and the screen coordinates and the RGB values of the color of the cursor position.

Clicking any one of the mouse buttons will return normal control to the mouse cursor. The Mouser dialog will continue to show the information for the window that was displayed when the mouse button was clicked. To find information on a new window, click on Mouser.

To return to the Win walker dialog, click on exit or double click on the Win Walker dialog. Double clicking on the Win Walker dialog has the same effect as clicking on the exit button. The Mouser dialog will disappear, and the Win Walker dialog will reappear with detailed information about the window that was selected with the Mouser.

Message

Scout allow you to easily send or post any standard Window message to any window or windows. When Mail is started from the Win Walker Dialog, the window whose information displayed in the Win Walker dialog is pre-selected in the Windows list box..

Next, select the window(s), from the Windows list box, that you wish to send a message to (if other than the preselected window). You can select more than one window by holding down the CTRL key while clicking and/or you can select a range of windows by holding down the shift key while clicking on a window in the list box. Undocumented Messages are displayed in mixed case.

While you can select multiple windows to send your message to, only one message can be selected at a time. Privately defined messages are not supported.

Once a message has been selected, Fill in the wParam and lParam parameters. You can enter either be a 32 bit value or hi and low words. Like the register edit boxes in the Call Function dialog, the 32 bit value will be updated if you modify one of the word values and the word values will be updated if you modify the 32 bit value.

Alternately, the lParam can be a pointer to a string buffer. The length of the string buffer will be the same length as the string in the string edit box (same edit box as used for the 32bit lParam) . If no string is present, a buffer of 10K bytes is allocated.

If needed, you can update the window list by clicking on the update button.

To send or post a message, click on the Send or Post buttons respectively. The selected message will be sent to all selected windows in the order they appear in the Windows list box.

Help Interceptor

When enabled (see Options), Scout will popup a dialog box showing which module made the WinHelp call, the help file being used, the type of help call (contents, context, key, etc.) and the help ID or key.

Scout determines the module making the call by the window handle that is being passed to the WinHelp API call. Note that it is possible that a program may pass a window handle that is associated with another module. In particular, a dll may pass a window handle asscociated with the exe that loaded the dll.

You have the option of letting the help file open or to ignore the call.

This feature can be useful in debugging context sensitive help.

System Metrics

Accessed from the Win Walker dialog, the Metrics dialog displays various system level parameters.

About the Author

Michael Bonincontri

I'm a Windows & OS/2 software developer at Wall Data. Prior to breaking into the software industry, I was an officer in the US Naval Submarine force.

I wrote Scout for many reasons: as a project for a university course, for use at work and to learn more about areas of Window's programming I'm not normally exposed to.

To comment on Scout, or to just boost my ego, feel to contact me on the Internet or AOL

MBONINCO@aol.com
or
MBONINCO@hq.walldata.com