Tanmyat AL-Rafidain 87(29) 2007
P.P [9 - 18]

Intercepting Windows Messages

Rawaa Putros Polos
Ass. Lecturer
Department of Computer Science
College of Computers and Mathematics Sciences
University of Mosul
Rawa_qasha@yahoo.com

Abstract

Windows messages are the heart of Windows operating system, since they have been used to communicate with other programs and the system. When Windows receives user events, the system sends them as messages to the programs, in order to respond to the user requests. So, the system controls other programs using these messages. When a program user attempts to intercept system messages, it gives the ability to manipulate and modify or even discard messages bound to other programs within the operating system, which offers a chance to change the behavior of these programs, and consequently control them.

This research presents software to intercept some types of Windows messages sent to other running programs to either modify or discard them. Additionally, software offers an ability to monitor the active program (that have been intercepted their messages by the program researcher) as well as all messages received from the operating system.

The researcher used Visual Basic 6.0 to develop the software

إعتراض رسائل الويندوز

رواء بطرس بولص
مدرس مساعد-قسم علوم الحاسبات
كلية علوم الحاسبات والرياضيات-جامعة الموصل

المستخلص

تعد رسائل النوافذ قلب نظام التشغيل ويندوز، لكونها تُستخدم للإتصال بين النظام وبقية البرامج. فعندما يستقبل نظام التشغيل أحداث من المستخدم يقوم بإرسالها كرسائل للبرامج لتقوم تلك البرامج بالإستجابة لطلبات المستخدم وبذلك فإن نظام التشغيل يسيطر على البرامج بإستخدام الرسائل. فمحاولة إعتراض هذه الرسائل من قبل برنامج لمستخدم تعطي إمكانية التعامل مع رسائل برامج تعمل ضمن النظام وتغيير أو إلغاء هذه الرسائل وهذا يوفر فرصة لتغيير سلوك تلك البرامج وبالتالي السيطرة عليها .

يقدم هذا البحث برنامج يقوم بإعتراض بعض أنواع رسائل الويندوز المرسلة لبرامج في حالة تنفيذ وإمكانية تغييرها أو إلغاءها فضلاً عن توفير إمكانية مراقبة البرنامج الفعال (الذي يتم اعتراض رسائله من قبل برنامج الباحث) وجميع الرسائل التي يستقبلها من قبل نظام التشغيل. استخدام الباحث لغة فيجوال بيسك ٦.٠ بغية تطوير البرنامج .

تاريخ التسلم ٢٠٠٦/٧/١٧ ______________________________ تاريخ قبول النشر ٢٠٠٦/١٠/١٨

1. Introduction

Microsoft Windows is a message - based system. This means that every action taken when using the system that creating one or more messages to carry out the action [Morries, 2002, 2].

Windows receives user input in form of "messages" to the window. It also uses messages to communicate other windows and all running programs.

Windows operating system sends a message to the program which means, it call a function within the program, a function that is an essential part of the program code. The parameters of this function describe the particular message that is being sent by Windows and then received by the program. This function is called Window Procedure.

The primary task of a window - based application is to process the messages received. This process involves routing of messages to the intended target Windows and the execution of expected responses to those messages according to the message type and parameters [Petzold, 1998, 35].

2. Basics of Windows Message:

Message is basically a structure that Windows generates each time to detect a certain event. It is used to package up and sends data between program and operating system. Message is a notification of some occurrence sent by Windows to an application [Yao, 2000, 15].

Messages are generated by both the system and applications. The system generates a message at each input event and respond to changes in the system brought by an application. An application can normally generate messages to direct its own windows to perform tasks or to communicate windows in other applications [Ezzell, 1998, 80].

Every time the user takes an action that affects window, such as resizing window, moving or clicking the mouse or pressing a key on the keyboard, the user's action triggers an event. Each time, an event is detected by the operating system and sends a message to the program, so that the program handles the message [Ezzell, 1998, 85]. So an event causes windows to send a message to an application and notifying it of what has been occurred.

Message structure contains information, such as what type of event occurred and additional information related with the message. The message structure of a mouse button click the message, for example, contains the mouse coordinates at the time that the button was pressed [Prosese, 1999, 5-8][Mueller, 2006].

This structure includes the following parameters:

- Windows Handle: The 32-bit window handle that identifies which window the message should be sent to.
- Message Identifier: It is a named constant that identifies the type of message sent.

- Lparam: It contains information used to process the message. These data varied by message type.
- Wparam: It contains information used to process the message. These data varied by message type.

Messages are passed between objects within the system. These messages carry information with them, and give the recipient more details on how to intercept and act upon the messages [Petzold, 1998, 42].

3. How the Windows Messaging System Works:

Under Windows, instead of applications, receiving information directly from the keyboard or the mouse driver, the operating system intercepts all input information, packaging this information using message structure and then forwarding the prepared messages to the appropriate recipients [Ezzell, 1998, 75].

A Window application's message system has three major components:

- Message Queue: Windows maintains a message queue for each application. Windows application must get messages from this queue and dispatch them to the proper window.
- Message Loop: This is the loop mechanism in a Windows program that fetches a message from the application queue and dispatches it to the appropriate window, fetches the next message, dispatches it to the appropriate window, and so on.
- Window Procedure: Each window in your application has a window procedure that receives each of the messages passed to it by the message loop. The window procedure's main job is to take each Windows message and respond to it accordingly.

So, a message flows from the OS to window as follows:

When a user clicks a button on a certain window, the mouse's device driver captures the click event. The device driver puts the event into the system message queue. The OS then pulls the event off the system message queue and converts it into a true Windows message. The OS then posts the message to the local message queue of the clicked window. The application owned that window constantly polls the local queue for messages [Balena, 1999, 122].

Getting a message creating, some event occurs in a window, the application responds to the message in five - step processes. Figure 1 shows these steps.

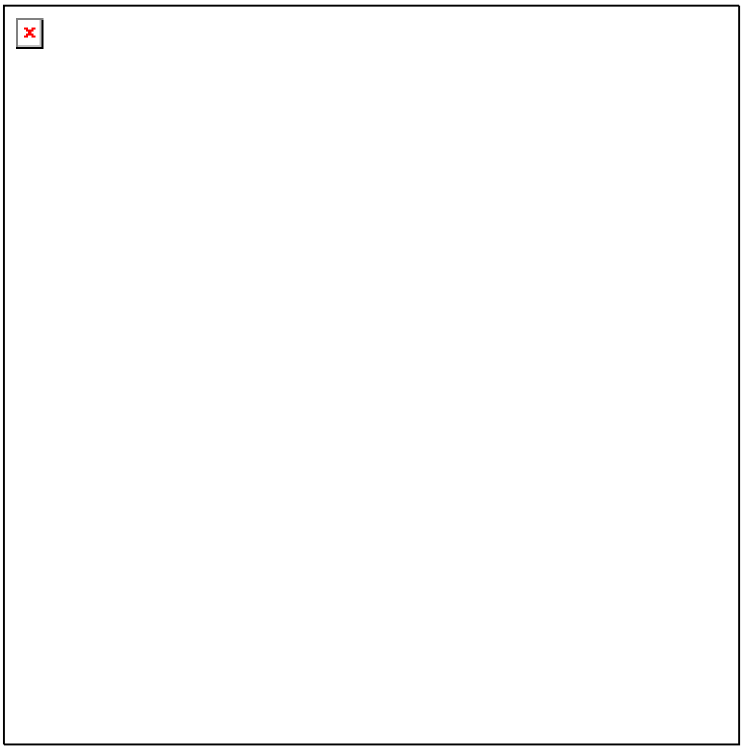


Figure 1
The Windows Messaging System

1. Some event occurs in the system.
2. Windows translate this event into a message and place it in the message queue for the application.
3. The application retrieves the message from the queue.
4. The application passes in the message to the window procedure of the application.
5. The window procedure performs some action in response to the message.

Steps 3 and 4 make up the application message loop. The message loop considered the heart of a window program, because the facility enables the program to respond to the external events. The message loop spends its whole life fetching messages from the application queue and passing them to the appropriate window procedure in the application [Morris, 2002, 10], as shown in figure 2.

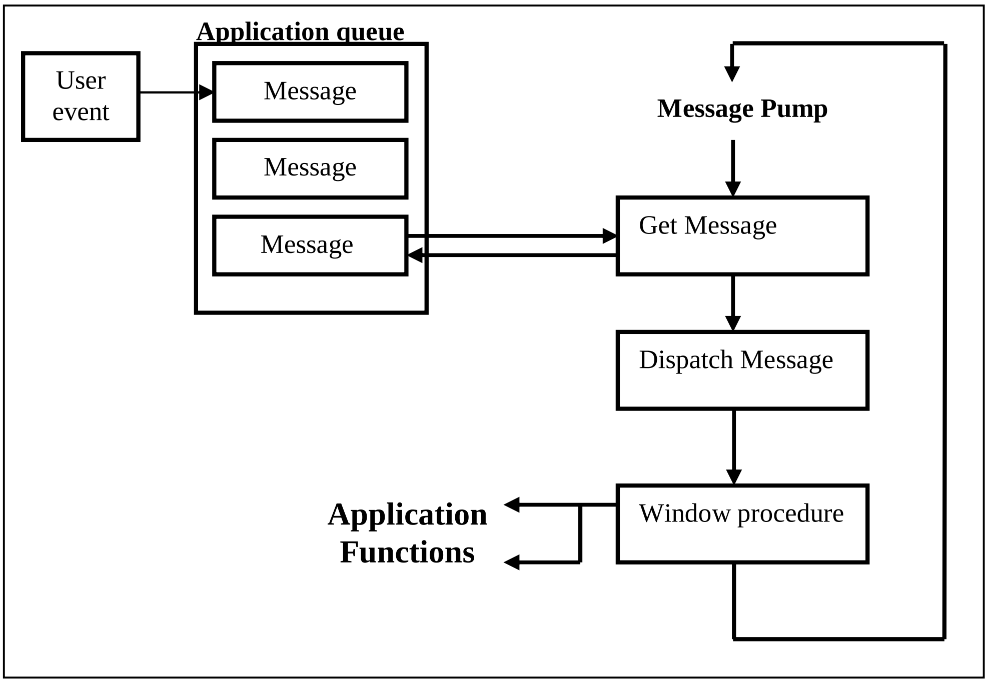


Figure 2
Windows Message Pump

4. Intercepting Windows Message:

Microsoft Windows controls applications through sending messages to window that are created by each application. The way that each application processes the message sent by Windows to determine the behavior [Mueller, 2006; Marsh, 2000].

Handling or processing a message means the application responds in some manner to a Windows message. In a standard Windows application, message handling is performed in each window procedure.

Message interception is a point in the system message -handling mechanism where an application can install a subroutine to monitor the message traffic in the system and process certain types of messages before they reach the target window procedure [Mcmahon, 2003].

So, intercepting messages must happen before the application’s window procedure. It has a chance to process them. After the interception operation, the message can be processed which make it possible to augmented, modify, or monitor the behavior of that application [Teilhet, 2001, 136-139].

The interception operation must be the main part of a program that process messages sent or posted to a particular application.

The interception program can take three actions upon receiving a message [Mcmahon, 2003]:

1. Modifying the message and pass it to its original window procedure.
2. Pass the message to the original window without modifying.
3. Discard the message.

5. Implementation Message Interception

The program that is responsible for intercepting messages sent to other application should determine the following:

1. The application has to be intercepted by messages.
2. Types of messages have to be intercepted.
3. Types of action that can be taken upon each type of these messages. Action type depends on the corresponding intercepted message.

The interception operation had achieved using Tegosoft spy ocx control. This control offers number of tools including subroutine, variable and methods to accomplish the job.

Message - Trap is an important subroutine that traps messages that are sent by Windows to all running application, it must contain code that receives operating system message, analysis and intercepts them, then changed them.

msgProgramWindowTitle is the control's variable contains the name of the active window that belongs to an application.

MsgMessage is one of control's variables that contains message ID be sent by the OS.

MsgWParam is the control's variable contains additional information related to the message.

The following steps can summarize the message intercepting operation as depicted by the flow chart (figure 3):

1. Obtaining the running program names in the system.
2. Determining name of application to be intercepted in the messages.
3. Determining message type and action to be applied on that message.
4. Calling message - Trap() subroutine.
5. Comparing the determining program with the content of msgProgramWindowTitle. If the match continue.
6. Comparing the determined message type with the content of msgMessage variable of the control. If they match then continue.
7. Applying the desired action upon that determined message.

The list of all running programs in the system can be obtained via using a number of Windows API functions that may detect these programs. These API functions are as follows:

ProcessFirst() and ProcessNext which contained in the kernal32 library in Windows operating system.

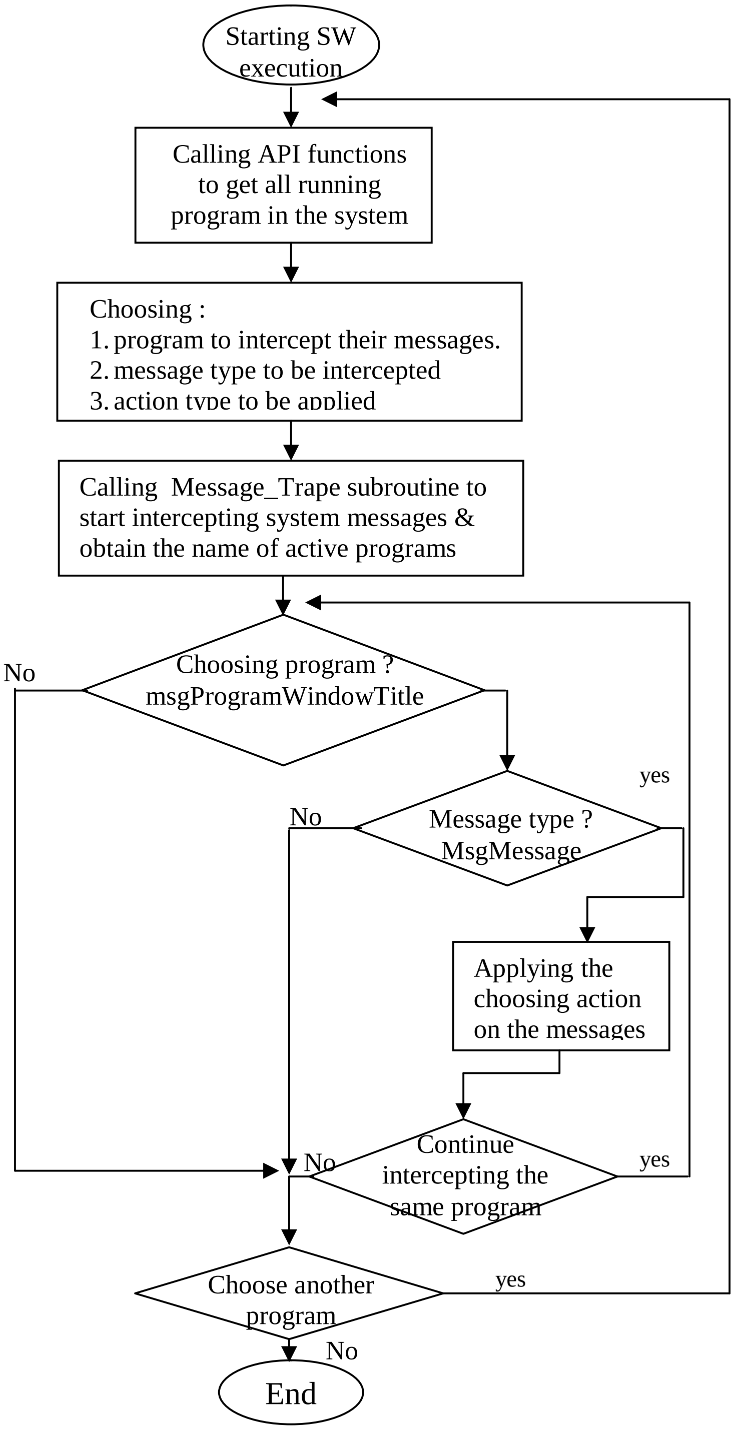

Starting SW execution
Calling API functions to get all running program in the system
Choosing :
1. program to intercept their messages.
2. message type to be intercepted
3. action type to be applied
Calling Message_Trape subroutine to start intercepting system messages & obtain the name of active programs
Choosing program ? msgProgramWindowTitle
No
yes
Message type ? MsgMessage
No
Applying the choosing action on the messages
Continue intercepting the same program
No
yes
Choose another program
yes
No
End

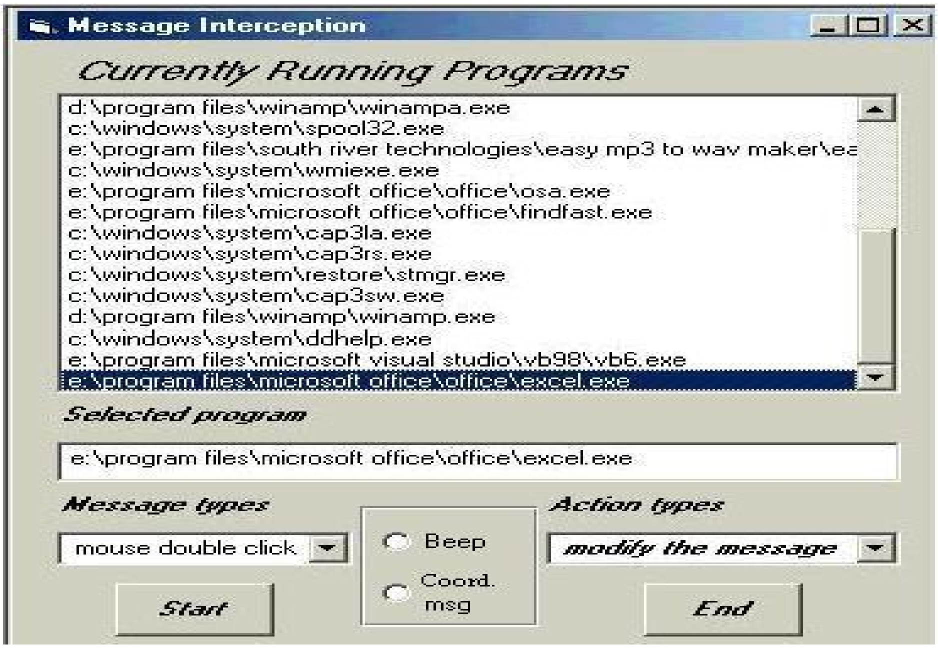
Figure 3
Software Flow chart

6. Software Preview:

Message Interception software has been developed by using Visual Basic (6.0). It contains two forms.

The first form represents software main interface and contains the following parts as shown in figure (4):

1. List of all running programs within the system. As mentioned in the previous section. These programs are obtained using number of Windows API's function.
2. Text box that display the path and name of the selected program to be intercepted from the above list.
3. Combo box1 contains types of messages that can be intercepted by the software and they are:
   - Keypressed message.
   - Mouse double clicks.
   - Mouse click.
4. Combo box2 contains type of actions that can be applied to the selected message and they are:
   - Discard message.
   - Modify message.
   - No change.



```
d:\program files\winamp\winampa.exe
c:\windows\system\spool32.exe
e:\program files\south river technologies\easy mp3 to wav maker\ea
c:\windows\system\wmiexe.exe
e:\program files\microsoft office\office\osa.exe
e:\program files\microsoft office\office\findfast.exe
c:\windows\system\cap3la.exe
c:\windows\system\cap3rs.exe
c:\windows\system\restore\stmgr.exe
c:\windows\system\cap3sw.exe
d:\program files\winamp\winamp.exe
c:\windows\system\ddhelp.exe
e:\program files\microsoft visual studio\vb98\vb6.exe
e:\program files\microsoft office\office\excel.exe
```

Selected program: e:\program files\microsoft office\office\excel.exe

Message types: mouse double click

Action types: modify the message

Figure 4
Software Main Interface

For the first action type, the application program will not receive the message at all.

When “modify message” action has been selected by the user, type of modification differs according to the message type. If the user select keypressed message, an input box appears as shown in figure (5) asking the user to type number of letters (3 letters as a maximum).

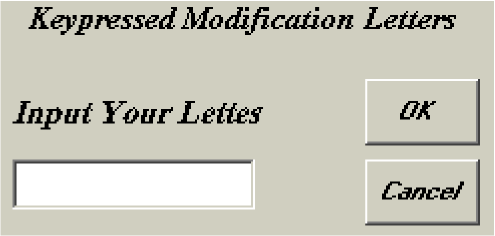


Figure 5
Letters Input Box

If the user input more than 3 letters, an error message will be appeared. These letters will be appeared instead of the letters that had been typed in the selected application. For example, if “Painbrush” is selected as an application to be intercepted its message. The user selected keypressed message as a message type and “msg” as the replacement letters, then Paintbrush always will type the letters “msg”, whenever the user types anything in the Paintbrush.

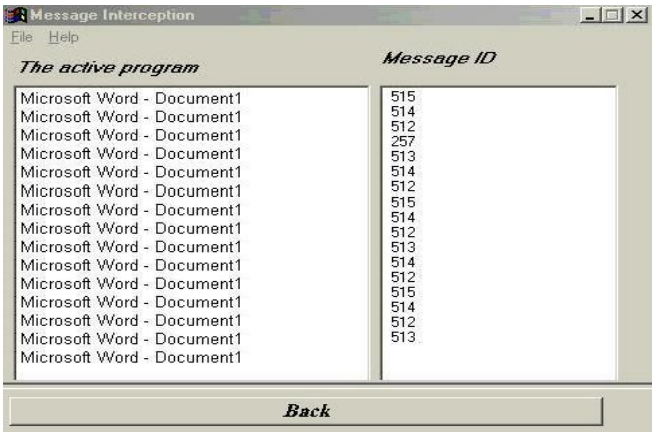
While in the case of selection mouse double clicks message type, two option buttons will be activated on the form to be chosen; one of them as the modified action type. The first option produces a beep sound, and the other display message box that views coordinate of mouse position at the time of occurred event. Mouse click message is handled as the same way as double clicks message.

At the end of selection operation, “start” button must be pressed to begin the interception operation on the selected program and the next form will be appeared as depicted in figure (6).

The second form displays two lists:

The first list display the current active program, while the second list displays all messages that are sent to the active program.

The software offers the ability to back to the first form and choose anther program and message type then start a new interception operation.



The active program

```
Microsoft Word - Document1
Microsoft Word - Document1
Microsoft Word - Document1
Microsoft Word - Document1
Microsoft Word - Document1
Microsoft Word - Document1
Microsoft Word - Document1
Microsoft Word - Document1
Microsoft Word - Document1
Microsoft Word - Document1
Microsoft Word - Document1
Microsoft Word - Document1
Microsoft Word - Document1
Microsoft Word - Document1
Microsoft Word - Document1
```

Message ID

```
515
514
512
257
513
514
512
515
514
512
513
514
512
515
514
512
513
```

Figure 6
Software Second Interface

7. Conclusion

Since, Windows message is considered to be the heart of the operating system. As a result, the messaging system is very complex, and handling with the messaging system, it is however critical to master the interception operation.

By this technique, we have a powerful influence of how the program will react to the message it receives, and monitoring messages that can help to better understanding on how the Windows applications really work.

The interception technique tends to slowdown the system, because they increase the amount of processing the system and must perform each message.

References:

1. Ben Ezzell & Jim Blaney, “Windows 98 Developer’s Handbook”, 1998, Gary Masters.
2. Charles Petzold, “Programming Windows”, 1998, Microsoft Press.
3. Francesco Balena, “Programming Microsoft Visual Basic 6.0”, 1999, Microsoft Press.
4. Jeff Prosise, “Programming Windows With MFC”, 1999, Microsoft Press, 2nd Edition.
5. John Mueller, “Hooking Windows Messages in .NET”, 2006, http://www.devsource.com/article2/0,1895,196 9408,00.asp.
6. John Mueller, “Working with Windows Messages”, 2006, http://www.devsource.com/article2/0,1895,1961574,00.asp.
7. Kyle Marsh, “Safe Subclassing in Win32”, 2000, Microsoft Developer Network Technology Group (MSDN).
8. Paul Yao & Richard C., “Visual C++ 5 Bible”, 2000, International Data group company.
9. Peter J. Morris, “How Windows Work”, 2002, The Mandelbrot set limited.
10. Stephen Teilhet, “Subclassing and Hooking with Visual Basic”, 2001, O'Reilly.
11. Steve Mcmahon, “Win32 Hooks in VB”, 2003, http://www.vbaccelerator.com/home/vb/code/libraries/Hooks/vbAccelerator_Hook_Library/article.asp