The Old New Thing

When programs grovel into undocumented structures...

23 Dec 2003 10:42 AM | **58**

Three examples off the top of my head of the consequences of grovelling into and relying on undocumented structures.

Defragmenting things that can't be defragmented

In Windows 2000, there are several categories of things that cannot be defragmented. Directories, exclusively-opened files, the MFT, the pagefile... That didn't stop a certain software company from doing it anyway in their defragmenting software. They went into kernel mode, reverse-engineered NTFS's data structures, and modified them on the fly. Yee-haw cowboy! And then when the NTFS folks added support for defragmenting the MFT to Windows XP, these programs went in, modified NTFS's data structures (which changed in the meanwhile), and corrupted your disk.

Of course there was no mention of this illicit behavior in the documentation. So when the background defragmenter corrupted their disks, Microsoft got the blame.

Parsing the Explorer view data structures

A certain software company decided that they wanted to alter the behavior of the Explorer window from a shell extension. Since there is no way to do this (a shell extension is not supposed to mess with the view; the view belongs to the user), they decided to do it themselves anyway.

From the shell extension, they used an undocumented window message to get a pointer to one of the internal Explorer structures. Then they walked the structure until they found something they recognized. Then they knew, "The thing immediately after the thing that I recognize is the thing that I want."

Well, the thing that they recognize and the thing that they want happened to be base classes of a multiply-derived class. If you have a class with multiple base classes, there is no guarantee from the compiler which order the base classes will be ordered. It so happened that they appeared in the order X,Y,Z in all the versions of Windows this software company tested against.

Except Windows 2000.

In Windows 2000, the compiler decided that the order should be X,Z,Y. So now they grovelled in, saw the "X" and said "Aha, the next thing must be a Y" but instead they got a Z. And then they crashed your system some time later.

So I had to create a "fake X,Y" so when the program went looking for X (so it could grab Y), it found the fake one first.

This took the good part of a week to figure out.

Reaching up the stack

A certain software company decided that it was too hard to take the coordinates of the NM_DBLCLK notification and hit-test it against the treeview to see what was double-clicked. So instead, they take the address of the NMHDR structure passed to the notification, **add 60 to it**, and dereference a DWORD at that address. If it's zero, they do one thing, and if it's nonzero they do some other thing.

It so happens that the NMHDR is allocated on the stack, so this program is reaching up into the stack and grabbing the value of some local variable (which happens to be two frames up

the stack!) and using it to control their logic.

For Windows 2000, we upgraded the compiler to a version which did a better job of reordering and re-using local variables, and now the program couldn't find the local variable it wanted and stopped working.

I got tagged to investigate and fix this. I had to create a special NMHDR structure that "looked like" the stack the program wanted to see and pass that special "fake stack".

I think this one took me two days to figure out.

I hope you understand why I tend to go ballistic when people recommend relying on undocumented behavior. These weren't hobbyists in their garage seeing what they could do. These were major companies writing commercial software.

When you upgrade to the next version of Windows and you experience (a) disk corruption, (b) sporadic Explore crashes, or (c) sporadic loss of functionality in your favorite program, do you blame the program or do you blame Windows?

If you say, "I blame the program," the first problem is of course figuring out which program. In cases (a) and (b), the offending program isn't obvious.

Blog - Comment List MSDN TechNet

Comments

Mike Dunn
23 Dec 2003 11:13 AM
#

I bet some of those "special" fixes (like the NMHDR one) have some very "colorful" comments attached to them. ;)

Just curious, how do you know when to use that "fake stack" NMHDR?

Colen
23 Dec 2003 12:03 PM
#

But WHY?

WHY do you do these things? Surely that only encourages them? If I write software that doesn't work, isn't it my responsibility to fix it, not rely on someone else to work out what on earth is going wrong and do so?

Braden Simpson
23 Dec 2003 12:44 PM
#

As an aside, why is NTFS so hell-bent on fragmenting itself? Even if I have half the harddrive available as contiguous space it seems that a large file will invariably get written across every little chunk it can find before finally encroaching upon a decent sized block. Install several large applications and you automagically end up in fragmentation city complete with harddrive thrashing performance loss...

Raymond Chen
23 Dec 2003 12:54 PM
#

Colen: Look at the scenario from the customer's standpoint. You bought programs X, Y and Z. You then upgraded to Windows XP. Your computer now crashes randomly, and program Z doesn't work at all. You're going to tell your friends, "Don't upgrade to Windows XP. It crashes randomly, and it's not compatible with program Z." Are you going to debug your system to determine that program X is causing the crashes, and that program Z doesn't work because it is using undocumented window messags? Of course not. You're going to return the Windows XP box for a refund. (You bought programs X, Y, and Z some months ago. The 30-day return policy no longer applies to them. The only thing you can return is Windows XP.)

Braden: I don't work on NTFS; you'll have to ask somebody who does.

asdf
23 Dec 2003 2:20 PM
#

How come you guys can spot obscure things like case 3 but you can't spot blatantly obvious things like explorer drawing the treeview items 1 pixel off in the "Folders" explorer bar? Not only are there double pixels on the dotted line but the tooltips are offset.

Colen
23 Dec 2003 6:12 PM
#

Raymond: Isn't this the sort of thing that compatability mode was designed for? Or would telling it to run in Windows 2000 mode (for example) not help in such cases?

Regardless, I can see *why* you do it, it just seems like a horribly bad idea to me. Give them an inch, etc.

Raymond Chen
23 Dec 2003 6:29 PM
#

"Compatibility mode" changes the behavior of the operating system. Things like, "GetVersion() reports Windows 95 (even though it's really Windows XP)," or "If you read the registry to detect what version of IE is running, you will see 4.0 (even though it's really 5.1)," or "if you call RegDeleteKey, perform a recursive deletion because that's what Windows 95 did."

It does not mean "Rewrite the innards of the operating system so all undocumented data structures look just like they were in Windows 95."

And "compatibility mode" doesn't help for shell extensions, since Explorer itself does not run in compatibility mode. (Compatibility mode is set for an entire process.)

Christian
23 Dec 2003 11:52 PM
#

>So I had to create a "fake X,Y"

>I had to create a special NMHDR structure
>that "looked like" the stack the program
>wanted to see and pass that special "fake
>stack".

Does this two things mean that Windows 2000 contains this stuff always?

Did you really do that only that these two programs work? I can't believe that!

That compatibility mode enables things like this is fully aceptable, but that windows is full of stuff that is only there for just a few programs is scaring!

Who decided that these buggy software should work with XP? Your manager?
Is this a policy of MS that these kinds of "fixes" must be made?

Why don't you write an angry letter to the offending company, let them fix it and block the software via the machanismns in Windows that give you these message boxes that software is too old?

I think the best would be to have an embedded link "Details" and if the user clicks on it Windows should tell the story what idiotic thing that ISV thought of. All those stupid trolls and computer magazines would read this notice and blame the other company and not MS. The computer-illiterate wouldn't understand this text, but they would read that they have to go to the vendor's homepage and fix it.

asdf
24 Dec 2003 12:43 AM
#

I agree with Christian that the vendors should fix their stuff instead of MS but that "Details" feature is like the kettle calling the pot black. Some of the API is missing so much functionality, it's not even funny. Like the built in controls not being able to respond to some style changes (yeah ES_MULTILINE and LB_SORT are really hard for them to respond to in WM_STYLECHANGED) or getting scrollbars to only appear when text overflows a box, etc., etc., etc. The only way to combat for the lack of functionality is extreme stuff like that or rewriting your own stuff from scratch.

I think Microsoft is to blame for the first case because it should have added defragmenting from the start. It would be pretty niave to say that a company that makes defragmenting software should miss out on useful features because it has to use undocumented structures.

The problem here is that there is no (free?) way to communicate with Microsoft developers that you're using stuff in undocumented ways and they should test against it when the OS changes so they can harass you to fix your own software when things break during testing instead of after the OS ships. If there is, then I haven't heard of it.

AlisdairM
24 Dec 2003 1:53 AM
#

In order to get permission to use the Windows Logo selling your product, you need to pass Microsoft test process.

Can this process not be made more aggressive, using a special version of Windows that specifically does nasty things with undocumented behaviour? Eg modifying all 'reserved' parameters in some clear way and renaming all undocumented/unsupported APIs?

Anything abusing the documented system is more likely to fail early.

The thought of tying the OS of the masses to some stone age architecture to support a few abusive programmers scares me. I suspect Win32 is a lost cause, but hope something more aggressive, along the lines above, can make it into the Longhorn process.

Oh, and make sure MS apps (apart from those shipping with OS) pass this same test suite, or else legal will be crawling all over you!

Christian
24 Dec 2003 2:56 AM
#
Have you tried the checked build, AlisdairM?

David Reynolds
24 Dec 2003 3:14 AM
#
Why not publish documentation for all the code and structures? Then, companies and software writers can use the correct methods and interfaces to obtain what they need, rather than having to use reverse-engineering to get what they need. At least they will then know what they can and can't use, and why:- and it will sort out many of these messes.

Mike Dimmick
24 Dec 2003 3:26 AM
#
David: it's called MSDN Library. If it's not in MSDN Library, _don't use it_!

Actually, we all have to take a view on whether to do something unsupported. I've had to write Pocket PC programs that persist Remote Access phonebook entries across device cold boots, and set them up on installation. The RasSetEntryProperties API allows you to change the phone number, etc, but you can't set up the modem through that API. There are no APIs to configure the Unimodem TAPI driver as appropriate.

Luckily, the Unimodem TAPI driver is included in the CE 3.0 and CE.NET Shared Source, including the appropriate data structure. However, this structure is not documented - my code could still break in future.

I'd like there to be a public, published API for everything that the user can do through the UI, including changing locale settings, because our customers (we're a vertical-market ISV, producing custom data capture applications) need to be able to pick up a unit with a completely flat battery and wiped memory and be up and running again in minutes, with as little user intervention as possible. Forcing the user to enter an additional dial string for configuring a GPRS modem is not on the cards! Ensuring that the device is set to English (UK) locale and GMT London time zone is also required (haven't worked this one out yet).

Larry Osterman
24 Dec 2003 7:26 AM

#

David,
The reason we can't publish the structures is that it precludes our changing them FOREVER. I mean FOREVER. There are structures in NTVDM that have existed since 1984 (or earlier) because there are apps that depend on them.

And Raymond has already pointed it the answer to a number of other comments: We can't just simply mark the app as being "broken" because it's our job to make sure that as many apps as possible work. You can't say "Hey, <Major Software Vendor's application that has millions of copies in circulation> isn't supported on Windows XP because <Major Software Vendor> doesn't follow the rules" if you actually want to ship operating systems. The companies that use <Major Software Vendors'> products won't buy your new operating system upgrade. Companies won't ditch their word processor just to put in an OS upgrade, they'll wait on the upgrade to keep their apps working.

Raymond Chen
24 Dec 2003 8:53 AM

#

"I think Microsoft is to blame for the first case because it should have added defragmenting from the start."

So you're arguing that Microsoft should not ship an operating system until every imagineable OS feature is implemented fully? By this argument, linux shouldn't have shipped ext3 because there isn't a e3defrag yet.

"Why not publish documentation for all the code and structures?"

That wouldn't have helped program (b). Even if they had the documentation on the internal structure, the change to the structure in Windows 2000 was caused by the *compiler*. Or are you saying that Windows should never upgrade the compiler, too?

And, as Larry noted, this would prevent us from ever changing the structure. It turns out that in Windows XP, that very structure was changed to add new features to Explorer. Are you saying that we shouldn't have added those features? Should we just plain stop changing Explorer ever again?

"I'd like there to be a public, published API for everything that the user can do through the UI."

I discussed this previously. There are some settings which intentionally have no API because that would allow bad people to manipulate them.

http://blogs.gotdotnet.com/raymondc/commentview.aspx/441c4836-e2ef-4088-bc06-936d41886fbc

"Who decided that these buggy software should work with XP?"

You, the customer.

"Why don't you ... block the software...?"

I already discussed this.

http://weblogs.asp.net/oldnewthing/archive/2003/12/19/44644.aspx

"ES_MULTILINE and LB_SORT are really hard for them to respond to in WM_STYLECHANGED"

Actually, ES_MULTILINE is incredibly difficult to respond to WM_STYLECHANGED. You see, there are actually two different edit controls in the system, one for single-line and one for multiline. The WM_CREATE handler decides which one to create based on the style passed at creation. Changing it on the fly would mean having to replace one object with the other on the fly.

Changing LBS_SORT on the fly means that you can create inconsistent data structures: The listbox thinks it's sorted (and therefore can use binary search and other optimizations) when in reality it isn't sorted (because you started with an unsorted listbox and then flipped the LBS_SORT bit).

asdf
24 Dec 2003 2:50 PM
#

I know they're two different objects but I don't see how hard it is to handle ES_MULTILINE, just save the handle (whatever you get from EM_GETHANDLE), create a new one, set the handle again (but don't use SendMessage, just call the internal functions to do so), and if that succeeds destroy the old one or else keep the old stuff the same. You'll be doing everything the app developer has to do except this one will keep the HWND value the same.

Flipping the LBS_SORT bit on should assume everything is unsorted and resort everything from scratch.

I'm not arguing MS should ship defrag stuff from the start, I'm arguing that if MS doesn't add a feature, they shouldn't complain (well... they should complain, but at least they should accept it as a fact of life) that some company had to use something extreme to get it to work because they're not going to skip out on a cool feature if it's possible for it to work. Doing it your way without touching undocumented stuff will make MS the authority of what software can and cannot do. (Though I still stand by that the NMHDR case above is just plain dumb on their part).

Hennk Devos
27 Dec 2003 5:23 AM
#

Raymond, you know that i will not agree with you on this.
I do agree fully with the NMHDR problem: This is very bad behavior of the programmers.
But you are saying that a company writing a defragger SHOULD NOT BE ALLOWED to build CERTAIN FUNCTIONALITY into their software? This is never going to happen!
As long as you don't give us, developers, certain information, we will reverse-engineer and accidents will happen.
As i argued before, why not share information on undocumented things and make it very clear that it will not work in the next Windows version?
What the defrag programmers should have done in my opinion is not start up on a newer OS on which it was not tested. Instead they could display a message: This version of our software is not working on your opersting system. Please go *here* for a free update.".
Then instead of Microsoft patching the OS, they could have helped the developer patch the application.

But please please please, stop telling us which functionality we are allowed to implement for our software. This is completely unacceptable.

Raymond Chen
27 Dec 2003 8:20 AM
#

If only apps that relied on undocumented behavior were always so conscientious I could be swayed. But your very own program that relies on an undocumented (and changing, as you even noticed yourself) interface http://www.codeproject.com/shell/selecticon.asp doesn't follow your own advice.

Henk Devos
28 Dec 2003 1:53 PM
#

Yes i admit, i have sinned many times myself.
The article you refer to is 4 years old, and i did learn some things during that time. That's also why i want to have these discussions: Maybe we can figure out a good solution together. and a guideline that programs using undocumented functions should stop themselves from starting up on Windows versions they were not tested on might be exactly the solution we are looking for.
But you also have to admit, this function should have been documented from the start. It's a common dialog that many developers want to use.

Raymond Chen
28 Dec 2003 2:59 PM
#

(stupid viewstate error)

This article is much newer and doesn't do any version checks either.

http://www.codeproject.com/shell/foldertasks.asp

Documenting a function isn't just writing a web page and a header file. It implies a continuing support commitment, dediciation of testing resources, and a promise that the function will continue to operate as documented for the indefinite future, even if the underlying technology changes radically. Doing this for the "Pick Icon" dialog is pretty silly, since you can always write your own. (I've done it. It's not hard.) It's not like it's impossible to write one.

余啊雷
28 Dec 2003 5:23 PM
#

Henk Devos
29 Dec 2003 12:30 AM
#

It's easy to do yourself, true.
Once, on Windows NT 4, i mimicked the security dialogs (permissions etc) for my private secured objects. At that time there was no way to bring up the real security dialogs. I did this all using only documented functions. This took me lots of work to program. But then came Windows 2000 with completely changed security dialogs. This was a system specific for one company (Belgian railways) on computers dedicated to this job, so the computers never got upgraded. But if this would have been general-purpose software the result would

have been unacceptable on Windows 2000 (or not even have worked).
That is the purpose of common dialogs: when users want to perform the same task, they get the same dialog on the same OS.
If Microsoft suddenly decides to give the pick icon dialog a radical new look then do i have to program several dialogs and run a different one depending on the OS?

About the folder tasks article, it doesn't contain any sample code, let alone sample code with version checks in it. It just contains interface definitions. It mentions that it is Windows XP specific and in the comments i say it will not work on Longhorn. In the future i will be more explicit and include clear warnings that version checks are needed.

Mark Hurd
29 Dec 2003 5:21 PM
#
For the significant companies you can sick legal on them for reverse engineering.

Sooner or later there will be a test case on this so that we all know where we stand. With .NET this is even more important.

Mark Hurd
11 Jan 2004 11:39 AM
#
Good reasons here to not use Microsoft software.

Colen
11 Jan 2004 11:41 AM
#
Does all windows best weblog software suck this bad? <script>alert('help')</script>

Raymond Chen
11 Jan 2004 11:54 AM
#
Welcome all you slashdotters. Here let me insult Windows a bit, just to save you some time. "Micro$oft sux. Windoze is lame. Linux roolz. If Windows software sucks, it's Microsoft's fault."

acceleriter
11 Jan 2004 5:58 PM
#
Was that really necessary?

Raymond Chen
11 Jan 2004 6:53 PM
#
I figured I'd get all the insults out of the way, then people who wanted to have a meaningful discussion could do so.

Anonymous Coward
15 Jan 2004 3:56 AM
#

Actually, Mr. Checn, Windows software sucks because it's binary, closed-source, and written to satisfy endlessly diverse commercial interests that conflict with each other.

Raymond Chen
15 Jan 2004 7:32 AM
#

"written to satisfy endlessly diverse commercial interests that conflict with each other" - that's also why it's so successful. In the real world, sometimes you have to suck to succeed.

Anonymous Coward
15 Jan 2004 1:27 PM
#

You missed my point, which was that whether Windows sucks from a technical perspective (even though it seems inevitable given it's goal of satisfying those interests) is utterly irrelevant, because closed source software sucks period. This point is very much exemplified by the disgusting binary hacking you describe on this page. The idea of crawling up the stack to check variables is revolting, and would never happen in a non-proprietary system. I would never want to waste time on my system trying to deal with such nonsense when there are thousands of more productive, interesting things to do that don't involve playing with someone's regurgitated binary refuse.

In "the real world", there are better systems, that don't force you to waste time in such a manner. Unfortunately, the commercial market for software thinks differently at the moment, but I and many others like me hope it won't always be so because we think non-proprietary technology is better for everyone.

BTW, I don't see where other (possibly slashdot-reading) folk were flaming you that "Windows sucks, Linux rul3z" so it seems to me that you wanted to instigate such a flame war with your comment. Care to point out what you were retaliating to?

Raymond Chen
15 Jan 2004 1:32 PM
#

The insults started with "Does all windows best weblog software suck this bad" and "this sucks" so I thought I'd forestall them.

So suppose you're some app and you want to do something, but after checking the source code, you find that linux won't let you it. (Say you want some custom display behavior for the files in the directory that contains your program's data files.)

What do you do? Do you crawl up the stack looking for the information anyway? Do you grovel around KDE's internal data structures and patch yourself in? Or do you just shrug your shoulders and say, "Oh well, I guess I can't do that."

What if you really *really* want to do it?

hughk
15 Jan 2004 2:14 PM
#

I had the benefit of working with another large semi-closed source operating system. I say semi, because for a long time it was possible to get source listings of many interesting bits. You found all kinds of interesting things out from the listings and if you were careful *and built in a version check* you used them. Later the listings were dropped, but there were still sources for a lot of the info like the kernel and file system data structures. You accepted that they might change, but that was it.

I can understand your problems with people messing with undocumented stuff, but sometimes its the only option. The only thing that these guys did wrong was to forget the version sanity check.

However, the company that was reaching down the stack frames was really doing something a little dangerous.

Raymond Chen
15 Jan 2004 2:30 PM
#

Even if they had the version check, that doesn't help the customer. Customer upgrades to XP, program says "Sorry, this program doesn't work on XP", customer returns XP to store for refund and tells all her friends, "Don't upgrade to XP, its compatibility is awful."

Anonymous Coward
15 Jan 2004 6:55 PM
#

With GNU ld (guh-new el-dee, a user-space dynamic linker), you can override any library function at run-time by using the LD_PRELOAD environment. There are some libraries that do this to add transparent compression for example, but it's mostly considered a dirty hack. Despite that, it's much cleaner than the method described here.

There's nothing technical that stops a software writer from engaging in those kinds of dirty hacks on a GNU/Linux or other non-prorietary system -- the only effective way to stop people from doing it is social pressure. But it is very effective, because many people and professional distributiors would refuse to use or to distribute software that contains such dirty hacks.

Your KDE example is not a very good one, because it's likely you could override some part of a window like the company was trying to do much cleaner way (I'm no KDE expert though). A better example with which i'm more familiar is Linux kernel modules (device-drivers for Linux basically). There you may have no choice but to use some dirtily-coded commercial proprietary module or you may not use your hardware. The better alternative there is to use different hardware with visible source code, but very often people are stuck. The only effective method is again social pressure. This method has seen some effectiveness in getting companies to distribute source code for their drivers, so that if they do include dirty hacks they can be removed.

Ed
23 Jan 2004 6:00 PM
#

In the first example, the defrag program had to resort to undocumented behavior because

Microsoft had the attitude of "NTFS doesn't need disk defragmentation, it never gets fragmented".

So when you say, "And then when the NTFS folks added support for defragmenting...", why did they add this support??? Because the defrag programs *proved* that defragmentation was necessary.

MS forced the defrag companies to resort to undocumented behaviors because MS refused to believe that it was necessary.

Mark Hurd
27 Jan 2004 4:16 PM
#
FYI the 1/11/2004 comment (#57592) is NOT from me.

Mark Hurd B.Sc.(Ma.)(Hons.)

余啊雷
10 Feb 2004 4:01 AM
#
Another tour of the MSI file format.

D.J. Wiendels
16 Feb 2004 1:39 PM
#
is it posible to make (new) undocumented structures unavaialble for client software? This would stop the //only-there-for-downwards-compatibillity// stuff from piling up in the future...

Raymond Chen
16 Feb 2004 1:45 PM
#
Depends what you mean by "unavailable". Remember that once you let a DLL into your process it can access anything in the process (because it is now part of the process). If you meant something else by "unavailable", please elaborate.

余啊雷
17 Feb 2004 8:15 AM
#
First consequences of the Windows source code leak

余啊雷
17 Feb 2004 9:22 AM
#
The first one, is that there has already been found a security bug in IE 5. It didn't take long to find anything! :)

The second one is that people have had a look at the

JustMe2
17 Feb 2004 4:46 PM
#
I love the fact that Microsoft using hidden and/or undocumented APIs is OK but if other companies use them it is wrong...

BTW, what every happend to that 'chinese wall' that seperated the App side of Microsoft from the OS side?

Raymond Chen
17 Feb 2004 4:51 PM
#
I personally still believe in the Chinese Wall. And as an OS person, I personally disapprove of using undocumented APIs outside the OS itself regardless of who calls them.

JustMe2
17 Feb 2004 5:15 PM
#
Nice to see that Microsoft court testimony about the seperations boils down to words like believe. Why isn't it a policy?

Also, what happend to Microsoft statements during the antitrust trail that if the sourcecode was released it would harm the security of the USA. Then Microsoft released the code to China and Russia. Isn't that purjury?

Raymond Chen
17 Feb 2004 5:23 PM
#
I do not make policy or make court statements. I was just expressing my opinion.

Shaun
25 Feb 2004 4:39 PM
#
While I can understand your rationale, particularly given that you have to deal with the results of other programmer's mistakes, the simple fact is that users want novel functionality. If an undocumented interface or structure is required to make that functionality possible, then we have no option but to use it.

And the truth is that I don't see why I should apologise for that, I only resort to undocumented interfaces when there is absolutely no documented way of doing what I need to do. When I do use undocumented interfaces, I write code to sanity check the environment before using the interfaces in question and to disable their use if it isn't safe.

Obviously this is still a problem for you, since my application may stop functioning in the next release of windows, but there is simply no other option, no matter how annoying you may find it.

Manuel Alves
27 Feb 2004 4:06 AM
#

I liked the discussion until it got onto the Linux/windows thing. I don't have any preference for each; I just want to get the job done. As for the money thing, yes it’s true; you must pay Microsoft for Windows. In Linux case you must pay some guy that runs the system. Generally I prefer to depend on one company rather than on someone that can leave the next day.

Raymond Chen
16 Jun 2004 8:35 PM
#

Commenting on this article has been closed.

余啊雷
7 Apr 2005 12:12 PM
#

Hvis der er noget Microsoft kan, så er det at bevare bagudkompatibilitet (og derved, helt automatisk, fremadkompatibilitet). Microsoft bliver tit kritiseret for at bryde bagudkompatibilitet i nyere udgaver af deres programmer, men, helt ærligt, det er d

余啊雷
18 Jan 2007 12:16 AM
#

PingBack from http://blog.not-a-kernel-guy.com/2007/01/17/136

余啊雷
2 Apr 2007 10:44 PM
#

PingBack from http://alekseysanin.wordpress.com/2007/04/03/microsoft-and-backward-compatibility/

余啊雷
19 Mar 2008 2:26 PM
#

You’re about to see the mother of all flamewars on internet groups where web developers hang out. It

余啊雷
24 Mar 2008 10:56 AM
#

PingBack from http://sorokin88.wordpress.com/2008/03/24/%d0%9e-%d0%b1%d1%80%d0%b0%d1%83%d0%b7%d0%b5%d1%80%d0%b0%d1%85-%d0%b8-%d1%81%d1%82%d0%b0%d0%bd%d0%b4%d0%b0%d1%80%d1%82%d0%b0%d1%85-%d0%9c%d0%b0%d1%80%d1%81%d0%b8%d0%b0%d0%bd%d1%81%d0%ba%d0%b8%d0%b5/

余啊雷
2 Apr 2008 10:32 AM
#

PingBack from http://daarchitect.wordpress.com/2008/04/02/the-bbc-news-redesign-ramblings/

余啊雷
26 May 2008 10:17 AM
#

原帖：http://www.joelonsoftware.com/items/2008/03/17.html翻译：http://www.newsmth.net/bbstcon.php?board=J...

余啊雷
26 May 2008 10:21 AM
#

原帖：http://www.joelonsoftware.com/items/2008/03/17.html翻译：http://www.newsmth.net/bbstcon.php?board=...

余啊雷
14 Nov 2008 8:42 PM
#

The iPhone API is new. It has yet to mature. Yes, it does use a very mature 20-year-old foundation steeped in the NeXTSTEP heritage as its foundation, there's a lot that was removed or didn't make the translation from AppKit/Cocoa...