The Old New Thing

Suggestion Box 3, short answers (part 2 of who knows how many)

11 Sep 2008 10:00 AM | **45**

Another round of short answers to questions in the suggestion box.

How does Windows choose the monitor a window maximizes to?

> The window maximizes to the monitor selected by `MonitorFromWindow`. The algorithm the `MonitorFromWindow` uses to select the monitor is documented in MSDN.

How do you make your Win32 application safe from keyloggers?

> You can't. In the battle between kernel mode and user mode, user mode loses. That's why we have a distinction between kernel mode and user mode in the first place.

Why should you ever call MessageBox from inside a message handler that is re-entrant? "I'm having a hard time explaining to people that it isn't a blocking call."

> I don't even understand the question. And it *is* a blocking call. The function doesn't return until the user responds to the message box.

What happens when you pass NULL to ShellExecute, like you did in one of your articles? "Is UI suppressed if I pass NULL?"

> The window handle is used as the parent for any dialogs or messages that the `ShellExecute` function may need to display. If you pass `NULL`, then you get the same effect as if you had passed `NULL` to, say, `MessageBox`. In the article, the thread has no UI active, so any error messages displayed by the `ShellExecute` function will appear as top-level unowned windows. I discussed this issue in detail a few years ago. If you want to suppress UI, then pass the `SEE_MASK_FLAG_NO_UI` flag to `ShellExecuteEx`.

Is there some way to use Explorer's filename sort function?

> It's called `StrCmpLogical`. Michael Kaplan discussed this function two years ago with a follow-up last year. Of course, if you want to mimic Explorer exactly, you also need to respect the `REST_NOSTRCMPLOGICAL` policy.

Is there a way to sleep for less than one millisecond?

> I don't know either. I've never needed to do that. (I try to get eight hours each night.)

Why are notification icons limited to 16 colors in Windows 95 and 2000?

> They were limited to 16 colors on Windows 95 to conserve memory. Windows 2000 inherited that code and by default, each version of Windows works the same as the previous one. There are millions of lines of code in the shell. It's not like somebody goes through every single one of them and says, "Gosh, should we change this line in the next version of Windows?"

Does Raymond use _alloca?

> Nope. It wasn't in K&R C, so it's not part of my world view. And incautious use of

`_alloca` can result in security vulnerabilities.

Windows can only handle 64 threads.

This statement is patently false. We've seen that even without taking any special precautions, we were able to pack about 2000 threads into a process before running out of address space. What the person probably meant to write is that 32-bit Windows supports a maximum of 32 *processors*. The reason is that the functions that manage sets of processors (such as `SetThreadAffinity`) use a 32-bit value to represent processor masks. Note that for 64-bit Windows, these masks were expanded to 64-bit integers, so 64-bit Windows supports up to 64 processors in principle.

How does that Temporary Internet Files thing work?

It's a shell namespace extension junction point.

Why are some count parameters declared as signed integers?

I don't think there's a profound reason for it. Each API designer is empowered to decide how their functions will work. After all, the original `strlen` function returned a signed integer, too. You might want to ask Brian Kernighan; he was doing it before Windows. (Signed integers do have the slight advantage of being resilient to integer underflow. If `a` and `b` are both non-negative integers, then `a - b` will never underflow.)

Why does the desktop lose track of icons, so it has to refresh them?

I'm not sure what you mean by "lose track of icons". Maybe you're asking about lost change notifications (in which case redirected folders can cause problems with lost network notifications). Or maybe you're talking about icon repositioning (maybe the previous icon locations weren't saved).

Questions about DLL imports, exports, and cross-module memory

I inadvertently answered this question in a rather long series of articles on DLL imports and exports, and a discussion of cross-module memory allocation.

I want to see your blog stats.

I thought that too, until I saw them. When you get over a million hits a month, a list of all the referrals is just a giant pile of noise. I haven't bothered analyzing the referrals because I have other things to do in my spare time that are more interesting.

I'd love to see a series of things that are obvious to you but not obvious to everyone.

How do I know what is obvious to me and not obvious to everyone?

The next category is, of course, the people who ask questions on things that I listed as topics I am not inclined to cover.

I have a problem (describes problem).

I don't think your problem really is of general interest. But it's clear that you're not respecting the modality of the window.

I'm trying to improve the performance of my specific scenario.

This doesn't strike me as a topic of general interest.

What are your thoughts on this research project? What are your thoughts on this Microsoft product?

I think you confused me with Robert Scoble.

I have a problem (describes problem).

This doesn't strike me as a topic of general interest.

Why does Internet Explorer do X?

Internet Explorer is explicitly on the list of things I am unlikely to cover.

Or the people who ask questions I've already answered or questions I've chosen to answer elsewhere.

Help me modify files that I didn't write.

Doing this breaks servicing.

What is the long pause after listing a directory the first time?
Answered in A brief and incomplete history of FAT32.

And then there are the questions I can't or choose not to answer.

Why does the window manager force windows fully on-screen when the workstation is unlocked?

I don't know either. It makes sense to force windows on-screen after a change in resolution, but if the resolution didn't change between the lock and the unlock, there's really no need to go around "fixing up" window positions.

What did Apple copy from Microsoft and vice versa?

I'm not going to touch this one. I don't see any upside to my answering it, only downside, and I don't welcome the flamewar that will doubtless ensue.

Do you know anything about this?

No.

Something about defragmenting.

I have no special knowledge about defragmentation. Try The Filing Cabinet. They answered a few questions in one blog entry and even have a Defrag FAQ. Personally, I've been happy with Dave Whitney's defragmenter to defragment specifically-targeted files. (I don't defragment my entire drive because it seems like a waste of time.)

A question about Aero glass

I have no special knowledge about Aero glass.

How about an under-the-hood look at the Windows Vista Start menu?

I didn't work on the Windows Vista Start menu, so I don't know how it works.

Do you have any insights into the evolution of WinHelp?

Sorry, I'm an outsider just like you when it comes to help technologies.

Long rambling question about ACCESS_DENIED

I quickly lost interest.

Please explain the subtleties of the ScrollWindowEx function when scrolling child windows.

I don't know the answer and I don't feel like doing the research necessary to find out. Sorry. The fact that I left it unanswered from the previous suggestion box should have been a clue.

That's all for this year.

Blog - Comment List MSDN TechNet

Comments

Name required
11 Sep 2008 10:13 AM
#

> Is there a way to sleep for less than one millisecond?

>

> I don't know either. I've never needed to do that. (I try to get eight hours each night.)

Snorts with laughter

Rick C
11 Sep 2008 10:55 AM
#

"A question about Aero glass"

A quick search for "aero glass client area" would've found the answer. (This comment was aimed at the original questioner, not at Raymond.)

peterchen
11 Sep 2008 11:04 AM
#

>> Why should you ever call MessageBox from inside[...]

I assume that's supposed to say *never*, because then it makes sense.

The call to the message box is blocking. However, messages are still processed, so the message handler is reentrant.

Sample:

...

SetTimer(wnd, ID_Delay, 1000); // process this later!

...

WM_TIMER:

```
if (id == ID_Delay)

{

  ProcessPendingThing();

  KillTimer(ID_Delay); // avoid firing twice

}
```

all nice, right?

Now, if ProcessPendingThings contains:

```
if (!DoProcess(thing))

 MessageBox(wnd, GetThingError()); // oops?

}
```

timer events will be processed while the message box is still shown, you get a new message box each second.

This is especially fun with a 10 ms interval.

Adrian
11 Sep 2008 11:15 AM
#

Regarding: "Why should you ever call MessageBox from inside a message handler that is re-entrant? 'I'm having a hard time explaining to people that it isn't a blocking call.'"

I suspect the confusion stems from the fact that message boxes (and other modal dialog boxes) pump messages. If messages can be generated for other windows, then the dialog's modal loop will dispatch them, possibly leading to recursion.

Generally this isn't a problem, because the parent window of the dialog is disabled. However, if you get the parenting wrong, or if you have other "aunt or uncle" windows that you're not disabling, unexpected messages could be dispatched. If you don't completely understand what's happening, it's easy to mistakenly conclude that MessageBox isn't a blocking call.

I recall learning about this the first time when an assertion dialog didn't seem to stop the program execution in time to see what had happened.

Adrian
11 Sep 2008 11:39 AM
#

Just yesterday somebody asked me how to sleep less than a millisecond.

If you sleep at all, then you're thread forfeits the rest of its quantum. If there are any other threads that are ready to run, it'll be at least another quantum before your thread runs again. I'm not an expert in this area, and I know it varies by machine and OS version, but I believe a quantum can be on the order of 10 ms.

This is one of those cases where you have to ask, "Why do you want to sleep for less than a millisecond?"

In my colleague's case, he has code that's creating a set of files in rapid succession. He'd like each of them to have a unique timestamp. His current process creates them so fast, that clusters of them have identical timestamps because of the limited resolution of the file system's timestamps.

I suggested a simple while loop before creating each file that simply waits for the current time to increment enough that it would make a distinct timestamp. Something like:

while (now / resolution == last_timestamp / resolution) { }

xix
11 Sep 2008 11:58 AM
#

Regarding "Why does the desktop lose track of icons, so it has to refresh them?", I think the writer might have been talking about the phenomenon of your icons (for me, usually the desktop) getting all messed up. Not usually corrupted, but with a PDF file having the icon of IE, or an PSD mini-preview icon suddenly being outlook. If you click on it and hit F5, it will correct itself.

Josh
11 Sep 2008 12:14 PM
#

@Adrian - Why even go that far? On any file system that's not write-once, timestamps can be set programmatically (on Windows, see SetFileTime: http://msdn.microsoft.com/en-us/library/ms724933(VS.85).aspx). Create all the files you need, go back and give them sequential timestamps, and don't misuse sleeps or loops to make your program run artificially slowly.

Similar mechanisms exist under other OSes (touch on *NIX machines obviously needs such an API), so Windows isn't special there.

Josh
11 Sep 2008 12:18 PM
#

Addendum: Of course, if he really needs the files to be uniquely identifiable or have a unique ordering he should probably find a better mechanism than timestamps. I don't know the parameters, so this may not be an option. For all I know he's trying to apply a custom sort to video files in Windows Media Center and using "order by modification date" for that purpose, then he doesn't have a lot of options.

Vijay
11 Sep 2008 2:22 PM

#

I think you could sleep for 0 ms. I think that means the thread gives up the remaining part of the time slice for the scheduler to schedule another thread (it might well be the same thread that get scheduled right away though)

Neil (SM)
11 Sep 2008 2:29 PM
#

Hmmm, I thought the icon question simply had to do with the desktop icons occasionally refresh after closing a program -- oftentimes you'll see the blank white placeholder icons appear in their place while the actual icons are loading. Perhaps the questioner was assuming that the desktop was somehow "losing track" of the icons and needed to reload them whenever this happened?

Skip
11 Sep 2008 2:46 PM
#

My assumption on the icons question was that he was wondering why the icons would blink out and have to be redrawn occasionally. And I don't know for sure because I've never tested it, but wouldn't a call to InvalidateRect(NULL,...) do it? I always figured that, or something like it, was what was going on.

Tony Schreiner [MSFT]
11 Sep 2008 3:28 PM
#

Skip, regarding icons blinking, LockWindowUpdate() can cause this even if you're only attempting to lock a child window in your application.

Ron Parker
11 Sep 2008 3:33 PM
#

Ignoring the obvious problem with preemption and hardware clock resolution and all that, which of course make the question meaningless and the answer moot, SetWaitableTimer does let you specify the time in 100ns intervals.

If you really need an accurate sub-millisecond delay, I expect your only option is to time the processor you're running on somehow and busy-wait the time away. (And you can still be preempted while you busy-wait, and some processors adjust their speed dynamically based on various environmental and other factors, etc.)

But, of course, the question itself reeks of being the question you think you want the answer to, rather than the question you actually want the answer to.

Trevel
11 Sep 2008 3:34 PM
#

Darnit, it sounds like my new 128 processor computer isn't going to be useful for some time, then.

Neil
11 Sep 2008 4:09 PM
#

The most noticeable problem with the colour depth of notification icons in Windows 2000 is that Automatic Updates was designed for Windows XP, so unless you're in a 256-colour video mode (e.g. Terminal Services), it tries to load a full-colour icon, which then gets dithered down to 16 colours. Ugly.

Old Salt
11 Sep 2008 5:46 PM
#

[It wasn't in K&R C, so it's not part of my world view]

It must be pretty painful coding for L"Unicode" without wide characters.

(It's a joke - not a nitpick)

mikeb
11 Sep 2008 6:00 PM
#

>> Do you know anything about this? (http://blogs.msdn.com/oldnewthing/pages/407234.aspx#480923)

No.

<<

But you didn't address David Candy's immediate follow-up question:

"What do the stars mean?"

Dean Harding
11 Sep 2008 7:22 PM
#

"What do the stars mean?"

The stars are pinpricks in the blanket that god puts over the world at night time to indicate that it's time for little boys and little girls to go to bed.

On another topic, the question about the "under-the-hood look at the Windows Vista Start menu" seemed to be rather leading... of course the answer was going to be "it's still shortcut file/folder based" at which point Randolpho would be able to say "ah ha! Windows is crappy!"

Chris Charabaruk
11 Sep 2008 7:55 PM
#

Some days, Raymond, I wonder if you're as grumpy as me. Today is one of those days. :D

余啊雷
11 Sep 2008 8:18 PM
#

Lately seen on The Old New Thing: I want to see your blog stats.

Simon Cooke
11 Sep 2008 9:23 PM
#

Damnit :) I guess the only way to figure out the scrolling child windows thing is going to be to crack out the disassembler. Boy am I rusty though...

Matt Lee
11 Sep 2008 11:48 PM
#

"I'd love to see a series of things that are obvious to you but not obvious to everyone.

How do I know what is obvious to me and not obvious to everyone?"

I guess you'll have to add that to the list of things that are obvious to everyone but not obvious to you.

Worf
12 Sep 2008 1:37 AM
#

Sub-millisecond sleeps are possible. ZYou just need to find an API that'll let you do it. A common one on Unix(-like) platforms is select(), which takes in a timeout value. Alas, we

know Windows is not Unix, so...

(Hint: Look up WinSock)

Dean Harding
12 Sep 2008 2:41 AM
#

Worf: select(), even on Unix, will timeout with a granularity of a jiffy (I believe 10ms on Linux? It's been a while...)

SuperKoko
12 Sep 2008 5:55 AM
#

From Vijay:

"I think you could sleep for 0 ms. I think that means the thread gives up the remaining part of the time slice for the scheduler to schedule another thread (it might well be the same thread that get scheduled right away though)"

If there're lower-priority threads, they will never get CPU time.

If your thread has the same priority than other threads, then, another thread will take its time slice, and so, will probably not return before 10 or 20 milliseconds.

To get better accuracy (up to 1 millisecond), timeBeginTime is a solution.

But, I don't think Windows NT/2000/XP provide any kind mechanism to wait for less than 1 millisecond (e.g. 100 microsecond).

I guess Windows NT can reach a 100 microsecond task switch latency on old computers (e.g. Pentium 75 Mhz)!

So, you've to use a busy loop for those very short delays.

Warning: Windows NT/2000/XP aren't real-time operating systems.

From Dean Harding:

"(I believe 10ms on Linux? It's been a while...)"

The time granularity depends on a compile-time kernel setting.

With menuconfig, you can set it to 1/100th, 1/250th, 1/300th or 1/1000th of second.

IIRC, by default it's 1/250th of second (4 milliseconds), but I set it to 1 millisecond.

For more info "man 7 time".

Using the RTC, since it can generate periodic IRQ, it might be possible to wait for 1/8192 second if your task is the only active one (i.e. there're no CPU-intensive background task), but I didn't try.

Sebastian Redl
12 Sep 2008 6:04 AM
#

> Why does the desktop lose track of icons, so it has to refresh them?

I think the question is about the case where, after a long-running, memory-hungry full-screen window closes (especially games), the desktop needs to reload all icons. You see the default file icon for everything and then one after the other reverts to their intended look.

My guess (and that's a pure guess) is that the desktop simply chooses to forget the icons when any window goes to full-screen because the icons are obviously not needed and thus a waste of memory.

The reason that it takes so long (up to two seconds) to reload them is that each icon is in its own executable file, all of which have to be looked up and loaded, since they're most definitely not in any cache anymore (memory-hungry app was running).

Mike Dimmick
12 Sep 2008 6:15 AM
#

On SetWaitableTimer and select():

Just because you can specify your timeout with a tiny precision (1 microsecond in the case of select, 100ns for SetWaitableTimer) doesn't mean that the system can do that with the same accuracy. In practice the accuracy is the system clock interrupt period, typically around 15ms, but varies with the version of the OS and the HAL used, and whether anyone's used a multimedia timer (which can change the clock interrupt period). Sleep() rounds up your request to the next system clock interrupt.

That is, Windows notes how long you wanted to sleep for in terms of ticks, and on each clock interrupt checks to see if the thread should no longer be sleeping and marks it ready if so - but you still might not actually be scheduled, depending on whether something with higher priority is still executing. The same scheme is used for wait timeouts (e.g. WaitForSingleObject).

RUF
12 Sep 2008 11:03 AM
#

>Is there a way to sleep for less than one millisecond?

> I don't know either. I've never needed to do that. (I try to get eight hours each night.)

This one is the best.

alegr

12 Sep 2008 1:11 PM
#

"Nope. It wasn't in K&R C, so it's not part of my world view. And incautious use of _alloca can result in security vulnerabilities."

In fact, incautious use of pretty much everything in C can result in security vulnerabilities. But _alloca is a big kludge.

edgar
12 Sep 2008 6:07 PM
#

>>"Nope. It wasn't in K&R C, so it's not part of >>my world view. And incautious use of _alloca >>can >result in security vulnerabilities."

>In fact, incautious use of pretty much >everything in C can result in security >vulnerabilities. But _alloca is a big kludge.

Yes, let us reduce the thread stack size to zero to have no security vulnerabilities. :)

alegr
12 Sep 2008 7:07 PM
#

"I suggested a simple while loop before creating each file that simply waits for the current time to increment enough that it would make a distinct timestamp. Something like:

while (now / resolution == last_timestamp / resolution) { }"

Never mind that filesystem timestamp resolution can be worse that system time resolution...

David Bowman
12 Sep 2008 8:31 PM
#

"What do the stars mean?"

I don't know, but the thing's hollow... it goes on forever... and... oh my God! - It's full of stars!

I shall sleep now, whether for a millisecond or a millennium, I do not know.

Mark
13 Sep 2008 3:10 PM
#

> How do you make your Win32 application safe from keyloggers?

> You can't. In the battle between kernel mode and user mode, user mode loses. That's why we have a distinction between kernel mode and user mode in the first place.

Sure you can- bring up an image of a keyboard and have the users type with the mouse. Or randomly remaps the keys (okay, so the keylogger would still capture the keys, but they wouldn't make any sense)

[*And the keylogger intercepts the mouse clicks, correlates them with the pixels on the screen, uses OCR to recognize the character on the keycap, and steals your password anyway. If you randomly remap the key, the keylogger can just read the decoded keys out of your program's memory space. -Raymond*]

Christian
13 Sep 2008 5:43 PM
#

>Sure you can- bring up an image of a keyboard and have the users type with the mouse. Or randomly remaps the keys (okay, so the keylogger would still capture the keys, but they wouldn't make any sense)

1.Yes, you can make things to make your application safe from specific keyloggers.

2.But NO, you cannot make your user-mode-application safe from kernel-mode-keyloggers:

They can make screenshots and record mouseclicks.

The person who asked the original question probably had 1. in mind, not knowing that you always have to approach these problems like in 2.

(As if deleting your cookies and disabling form/autocomplete from the webmaster would make it magically safe to use online banking in an internet cafe....)

Christopher
14 Sep 2008 5:40 AM
#

>> The reason that it takes so long (up to two seconds) to reload them is that each icon is in its own executable file, all of which have to be looked up and loaded, since they're most definitely not in any cache anymore (memory-hungry app was running).

And antivirus suites with realtime protection will also rescan each EXE file for viruses before allowing the shell to reload their icons.

AndyB
14 Sep 2008 7:55 AM
#

I'd like to see your blog stats too - preferably piped through awstats first, of course.

As for defragmentation, I agree. That's probably why Vista defrags everything in the background continually, and worst of all, without a nice calming GUI to show blocks being moved! (I mean, what's the point of a defragger if you can't see it working)

Alexander Grigoriev
14 Sep 2008 10:46 PM
#

AndyB:

"That's probably why Vista defrags everything in the background continually, and worst of all, without a nice calming GUI to show blocks being moved! (I mean, what's the point of a defragger if you can't see it working)"

There is actually no point in running the defrag, at all. I'm not sure Vista does it, either.

scattergather
15 Sep 2008 12:08 PM
#

NTFS partitions can't be fragmented as fat32 could. No need for a defrag program at all. It's a clear placebo effect.

alegr
15 Sep 2008 12:28 PM
#

[scattergather] "NTFS partitions can't be fragmented as fat32 could".

NTFS partitions can be as fragmented as FAT32. The only difference could be made by allocation strategies, which depends on the filesystem driver, not on the filesystem type by itself. That said, NTFS doesn't suffer from directory fragmentation, because it keeps file records in MFT.

anony.muos
15 Sep 2008 3:53 PM
#

Why does Windows Me require half of amount of disk space to hibernate? If it has 256 MB of RAM, why does it require only 128 MB? Why not the NT family? Why was hibernation only kept in Windows Me OEM versions?

Igor Levicki
15 Sep 2008 6:31 PM
#

As for keyloggers:

1. Open notepad

2. Open word

3. click in password field enter one character

4. click on notepad and type a random number of random characters

5. click in password field enter one character

6. click on word and type a random number of random characters

7. goto 3 until pass entered

That will still confuse most keyloggers.

xenon155
15 Sep 2008 8:05 PM
#

@Igor

There is an easier way:

If your password is 1234

type 1abc2qwe3uyt4cxz into the password box

then go back and delete the chars using cut instead of delete.

Dean Harding
15 Sep 2008 11:21 PM
#

xenon155: Except your method requires you to remember the number of "dummy" characters you entered, because they'll all be displayed as "*".

Igor's method also works for Unix-style password prompts which don't echo back "*"s.

Of course, anybody that paranoid about keyloggers probably shouldn't be typing *anything* in...

Michiel
16 Sep 2008 10:07 AM
#

Since Raymond is skipping the ACCESS_DENIED question, let me answer it. My crystal ball tells me the users problem is that he doesn't know which ACL was tested for which right. This information can generally be found in the Audit log.

SuperKoko
16 Sep 2008 6:12 PM
#

"NTFS partitions can't be fragmented as fat32 could. No need for a defrag program at all. It's a clear placebo effect."

Actually, the Windows XP NTFS driver fragments files a lot. Writing two big files at once, create lots of small extents (e.g. downloading several files at once).

The disk of my brother's computer currently has 81% file fragmentation.