

Re: Cancel-safe without a thread?

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.development.device.drivers/2006-09/msg00148.html>

- *From:* "Calvin Guan" <hguan@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 4 Sep 2006 19:24:04 -0700
-

<BubbaGump> wrote in message
news:fu9pf2tb7raec7qs3o9l2aagkc17pk70i1@xxxxxxxxxx

I choose a combination of the alternatives: learning the current solution, doing the same or similar, and using the deeper understanding learned to gain insight that will help in debugging related weird issues. I don't reject the current solution. I just want to see what's inside of it, the motivations for it, before I use it. — I'm thinking that learning addition and subtraction gave me a better appreciation for multiplication and division. — If I couldn't see inside it because it was hidden inside object code then I'd move on. I don't mind if it's hidden inside KMDF.

I totally hear you. I used the exact approach when I learned windows driver programming. I insisted writing every single line of code in my driver. Then I compared my result to others'. When in doubt, I used debugger to figure out what the manual and/or others didn't say (or say it wrong.). I think many folks in this group have run into such situation that you likely won't get answer from anywhere else and hence are forced to use a kernel debugger stepping through the OS code to find out what really happened. The combined approaches give me better understanding the ins and outs of the operating systems and the driver model i have to deal with. I really learned when I screwed something up badly.

I can't do the same for linux b/c it doesn't and will not have a decent kernel debugger as Windows does and I don't want to read source code from others.

(This is actually one of the problems I have with Linux and open source in general. I can see all the source, and I have a hard time knowing at which depth to stop looking.)

Re: Cancel-safe without a thread?

No, I think you would stop at some point unless you have __unlimited__ time and resources. My experience in doing Windows drivers that being delivered to massive production, the OS internals/DDI is just a small part of the business (make no mistake, I've worked on H/W drivers that have *very* complex OS interface and interactions and requires to develop deep and broad driver stack in both um and km and interacts with multiple os device stacks). There are always tight schedules and back-to-back deadlines to meet, occasionally there are showstoppers reported from OEMs or Microsoft which require immediate attentions. I just have to stop poking around OS stuff unless it's directly related to the "business technical" problem I have to solve at the moment. I wouldn't mind throwing my own proven PNP/power/wmi/cancel framework away and use KMDF if I was to write a pnp driver today. But I still maintain the ability to figure out something myself if I really want to.

There are more interesting, challenge things in the emerging device and platform technologies which worth more of my time. Coming from my electrical engineering background and solely working for hardware and semiconductor companies, OS stuff is inherently less interesting compared to hardware technologies. YMMV, but my point is that given your tactics, I'm sure you will find something more interesting sooner or later. In the end, like someone had said, life is short — enjoy!

--

Calvin Guan (DDK MVP)
Sr. Staff Engineer
NetXtreme Vista/Longhorn Server Miniport
Broadcom Corporation
Connecting Everything(r)