Windows CE vs Embedded Linux [closed]

Asked 16 years ago Modified 8 years, 9 months ago Viewed 28k times

32

Closed. This question needs to be more focused. It is not currently accepting answers.

Want to improve this question? Update the question so it focuses on one problem only by editing this post.

Closed 8 years ago.

Improve this question

Now I'm sure we're all well aware of the relative merits of Linux vs Windows Desktop. However I've heard much less about the world of embedded development. I'm mainly interested in solutions for industry and am therefore uninterested about the IPhone or Android and more interested in these two OSes.

What are the relative trade-offs between the two platforms in the embedded world? If you were considering building a box for a specific project with custom hardware, a partially customised OS and a custom app then which would you choose and why?

I would assume that Windows CE wins on tools and Linux wins on both cost and possibly performance. However this is just utter speculation. Does anyone have any facts or experience of the two?

linux embedded operating-system windows-ce

Share

Improve this question

Follow

edited Nov 28, 2008 at 19:53

ctacke

67.2k ● 20 ● 98 ● 155

asked Nov 28, 2008 at 19:41

Quibblesome

25.4k ● 10 ● 62 ● 104

1 You may want to specify more details on the type of hardware and purpose since they will make a difference. You will probably find Linux will run on a very wide range of hardware, real time support is different between the two, etc... – carson Nov 28, 2008 at 19:49

1 well cannot talk about linux. but: if you are 'a serious developer' then maybe winCE is a good idea. if you are 'a lone kiddie' with an interest in embedded development, you may find winCE something like a closed garden with high walls, with limited free shared knowledge. – n611x007 May 14, 2014 at 11:18

8 Answers

Sorted by: Highest score (default)

46

I worked for several years at a company that provided both CE and Linux for all of their hardware, so I'm fairly familiar with both sides of this equation.

- **Tools:** Windows CE tools certainly are better than those provided by Linux, though the linux tools are certainly getting better.
- **Performance:** Windows CE is real-time. Linux is not. The linux kernel is not designed for determinism at all. There are extensions that you can add to get sort-of real time, but CE beats it.
- **Cost:** This is an area of great misunderstanding. My general experience is that CE is lower cost out of the box ($1k for Platform Builder and as low as $3 per device for a shipping runtime. "What?" you ask? "Linux is free." Well, not really so much, especially in the embedded arena. Yes, there are free distributions like Debian. But there are plenty of pieces that you might need that aren't in that free category. UI frameworks like QT, Java runtimes and media codecs just as a start. Also, most Linux distributions with a commercially-backed support system (e.g. MontaVista) are far from free.
- **Source Availability:** Linux proponents may like to say that CE is a bad choice due to lack of source code. All I can say is that in over a decade of working with CE, half of which spent doing custom kernel and driver work for custom boards, I've only ever had need for source that didn't ship with CE (they ship a

vast majority of it) once. I like having source too, but Microsoft provides support, so in the rare case you might think you need that source, you can get them to fix the problem (the one time we needed source, Microsoft provided a fix, and for free - which is their model under CE.

Does this mean that CE wins every time? No. I wouldn't suggest that at all. If you are a Linux shop and you have lots of Linux experience and code assets, you'd be foolish to run out and go CE. However, if you're coming into it from scratch CE usually has a lower TCO. Developers with Win32/C# experience are more prevalent and consequently less expensive. You also get a lot more "in the box" with CE than most other distributions, meaning faster time to market if you don't already have these things done in-house already.

Share Improve this answer Follow

edited Nov 28, 2008 at 20:00

answered Nov 28, 2008 at 19:52

ctacke
67.2k ● 20 ● 98 ● 155

---

2 Mostly agree. Some comments: 1. According to Microsoft document - WinCE price can rise up to 16$ per device, depend on what components you choose to include. msdn.microsoft.com/en-us/windowsembedded/ce/dd630617.aspx 2. I didn't find Java run-time for WinCE. So the additional cost of license for Linux is not relevance. – landmn Nov 18, 2009 at 7:03

3 QT is free nowadays, at least if your license is compatible with LGPL. – wvdschel Jan 15, 2010 at 9:45

4 Yell about the benefis of Linux all you want. As I stated, I worked for years at a company where we delivered both Linux and CE for identical hardware. Your statements about CE show a lack of understanding about it. The license for many applications is in the range of $3 - hardly a deal breaker in a BOM. All task you can do in Platform Builder can be done from the command line (in fact is was command-line only for many years) and easily automated. CE ships about 95% of the source code - I've only had the need to see what wasn't shipped once in my life - over many, many products. – ctacke May 18, 2011 at 1:00

3 You can fix the kernel in CE if you need to (though I've never needed to). And to say no one needs "support" and then say that TCO is lower is a bit interesting. support isn't free, even if you're doing it yourself. As I said, CE doesn't win every time. Neither does Linux. There are solid business cases for both. Putting on blinders because you have some fallacy notion of Microsoft's "evil empire" is a disservice to people investigating what might actually get their product to market on time, under budget, and be maintainable for the future. If Linux does that for you, great. – ctacke May 18, 2011 at 1:01

11 I'll pass on the Linux-as-religion flame bait. Thanks. – ctacke May 24, 2011 at 12:50

26

I'll speak for the Linux side, at least for the category of software I'm familiar with (which is RF data collection equipment). Or industrial apps vs. consumer apps.

Windows CE (and its associated tools) IMH fairly recent E) is strongly biased to creating a "Windows Experience" on a small screen. The user input mode emphasizes mouse-like actions. Logons, application selection, etc. all try to be as similar to standard Windows as possible.

If a user is driving a lift truck, or filling a picking cart, or moving material from one place to another, there's a problem.

And it's a moving target - particularly on the .NET side. The Compact .NET runtime is seriously handicapped, and important libraries (like networking, data handling, and UI) are incomplete and versions too often deprecate the previous version. . CE seems to be the stepchild in the Windows family (possibly because there's not a lot of active competition selling to the hardware integrators.)

A nice stable rows-and-columns Linux console is a pretty handy context for many (in my experience most) high-use apps on a dinky screen.

Not much good for games on your cell-phone or Zune, though.

NOTE:

I think ctacke probably speaks accurately for the hardware integrator's side. I'm more aligned with the players further down the pipe - software integrators and users.

Share Improve this answer Follow

edited Nov 28, 2008 at 22:26

answered Nov 28, 2008 at 22:08

dkretz
37.6k ● 13 ● 83 ● 140

4 That's an interesting and useful view. I've done CE systems for headless as well as headed devices that look nothing like CE or Windows, but you're right that UI apps certainly tend to look "Windows-ish". I think that's simply due to developers using common controls instead of custom ones. – ctacke Nov 28, 2008 at 23:02

21

Choice is often made largely on perception and culture, rather than concrete data. And, making a choice based on concrete data is difficult when you consider the complexity of a modern OS, all the issues associated with porting it to custom hardware, and unknown future requirements. Even from an application perspective, things change over the life of a project. Requirements come and go. You find yourself doing things you never thought you would, especially if they are possible. The ubiquitous USB and network ports open a lot of possibilities -- for example adding Cell modem support or printer support. Flash based storage makes in-field software updates the standard mode of operation. And in the end, each solution has its strengths and weaknesses -- there is no magic bullet that is the best in all cases.

When considering Embedded Linux development, I often use the iceberg analogy; what you see going into a project is the part above the water. These are the pieces your application interacts with, drivers you need to customize, the part you understand. The other 90% is under water, and herein lies a great deal of variability. Quality issues with drivers or not being able to find a driver for something you may want to support in the future can easily swamp known parts of the project. There are very few people who have a lot of experience with both WinCE and Linux solutions, hence the tendency to go with what is comfortable (or what managers are comfortable with), or what we have experience with. Below are thoughts on a number of aspects to consider:

SYSTEM SOFTWARE DEVELOPMENT

Questions in this realm include CPU support, driver quality, in field software updates, filesystem support, driver availability, etc. One of the changes that has happened in the past two years, is CPU vendors are now porting Linux to their new chips as the first OS. Before, the OS porting was typically done by Linux software companies such as MontaVista, or community efforts. As a result, the Linux kernel now supports most mainstream embedded cpus with few additional patches. This is radically different than the situation 5 years ago. Because many people are using the same source code, issues get fixed, and often are contributed back to the mainstream source. With WinCE, the BSP/driver support tends to be more of a reference implementation, and then OEM/users take it, fix any issues, and that is where the fixes tend to stay.

From a system perspective, it is very important to consider flexibility for future needs. Just because it is not a requirement now does not mean it will not be a requirement in the future. Obtaining driver support for a peripheral may be nearly impossible, or be too large an effort to make it practical.

Most people give very little thought to the build system, or never look much beyond the thought that "if there is a nice gui wrapped around the tool, it must be easy". OpenEmbedded is very popular way to build embedded Linux products, and has recently been endorsed as the technology base of MontaVista's Linux 6 product, and is generally considered "hard to use" by new users. While WinCE build tools look simpler on the surface (the 10% above water), you still have the problem of what happens when I need to customize something, implement complex features such as software updates, etc. To build a production system with production grade features, you still need someone on your team who understands the OS and can work at the detail level of both the operating system, and the build system. With either WinCE or Embedded Linux, this generally means companies either need to have experienced developers in house, or hire experts to do portions of the system software development. System software development is not the same as application development, and is generally not something you want to take on with no experience unless you have a lot of time. It is quite common for companies to hire expert help for the first couple projects, and then do follow-on projects in-house. Another feature to consider is parallel build support. With quad core workstations becoming the standard, is it a big deal that a full build can be done in 1.2 hours versus 8? How flexible is the build system at pulling and building source code from various sources such as diverse revision control systems, etc.

Embedded processors are becoming increasingly complex. It is no longer good enough to just have the cpu running. If you consider the OMAP3 cpu family from TI, then you have to ask the following questions: are there libraries available for the 3D acceleration engine, and can I even get them without being committing to millions of units per year? Is there support for the DSP bridge? What is the cost of all this? On a recent project I was involved in, a basic WinCE BSP for the Atmel AT91SAM9260 cost $7000. In terms of developer time, this is not much, but you have to also consider the on-going costs of maintenance, upgrading to new versions of the operating system, etc.

APPLICATION DEVELOPMENT

Both Embedded Linux and WinCE support a range of application libraries and programming languages. C and C++ are well supported. Most business type applications are moving to C# in the WinCE world. Linux has Mono, which provides extensive support for .NET technologies and runs very well in embedded Linux systems. There are numerous Java development environments available for Embedded Linux. One area where you do run into differences is graphics libraries. Generally the Microsoft graphical APIs are not well supported on Linux, so if you have a large application team that are die-hard windows GUI programmers, then perhaps WinCE makes sense. However, there are many options for GUI toolkits that run on both Windows PCs and Embedded Linux devices. Some examples include GTK+, Qt, wxWidgets, etc. The Gimp is an example of a GTK+ application that runs on windows, plus there are many others. The are C# bindings to GTK+ and Qt. Another feature that seems to be coming on strong in the WinCE space is the Windows Communication Foundation (WCF). But again, there are projects to bring WCF to Mono, depending what portions you need. Embedded Linux support for scripting languages like Python is very good, and Python runs very well on 200MHz ARM processors.

There is often the perception that WinCE is realtime, and Linux is not. Linux realtime support is decent in the stock kernels with the PREEMPT option, and real-time support is excellent with the addition of a relatively small real-time patch. You can easily attain sub millisecond timing with Linux. This is something that has changed in the past couple years with the merging of real-time functionality into the stock kernel.

DEVELOPMENT FLOW

In a productive environment, most advanced embedded applications are developed and debugged on a PC, not the target hardware. Even in setups where remote debugging on a target system works well, debugging an application on workstation works better. So the fact that one solution has nice on-target debugging, where the other does not is not really relevant. For data centric systems, it is common to have simulation modes where the application can be tested without connection to real I/O. With both Linux and WinCE applications, application programing for an embedded device is similar to programming for a PC. Embedded Linux takes this a step further. Because embedded Linux technology is the same as desktop, and server Linux technology, almost everything developed for desktop/server (including system software) is available for embedded for free. This means very complete driver support (see USB cell modem and printer examples above), robust file system support, memory management, etc. The breadth of options for Linux is astounding, but some may consider this a negative point, and would prefer a more integrated solution like Windows CE where everything comes from one place. There is a loss of flexibility, but in some cases, the tradeoff might be worth it. For an example of the number of packages that can be build for Embedded Linux systems using Openembedded, see.

GUI TRENDS

It is important to consider trends for embedded devices with small displays being driven by Cell Phones (iPhone, Palm Pre, etc). Standard GUI widgets that are common in desktop systems (dialog boxes, check boxes, pull down lists, etc) do not cut it for modern embedded systems. So, it will be important to consider support for 3D effects, and widget libraries designed to be used by touch screen devices. The Clutter library is an example of this type of support.

REMOTE SUPPORT

Going back to the issue of debugging tools, most people stop at the scenario where the device is setting next to a workstation in the lab. But what about when you need to troubleshoot a device that is being beta-tested half-way around the world? That is where a command-line debugger like Gdb is an advantage, and not a disadvantage. And how do you connect to the device if you don't have support for cell modems in New Zealand, or an efficient connection mechanism like ssh for shell access and transferring files?

SUMMARY

Selecting any advanced technology is not a simple task, and is fairly difficult to do even with experience. So it is important to be asking the right questions, and looking at the decision from many angles. Hopefully this article can help in that.

Share Improve this answer Follow

edited Mar 24, 2016 at 3:34
Keshava GN
4,255 ● 2 ● 37 ● 49

answered Jun 20, 2009 at 19:20
user79117

13

I have worked in projects that involved customizing the software of an OEM board and I wouldn't say that Linux is cheaper. When buying a board you also need to buy the SDK. You still need to pay even for the Linux version. Some manufacturers offer both Windows CE and Linux solutions for their boards and there isn't a price difference. For Windows CE you also need the Platform Builder and pay for the licenses, but it is easier to go without support.

Another important issue is if you are building a User Interface or a headless device. For devices that require an LCD screen and human interaction is much easier to go with Windows CE. If on the other hand you are building a headless device, Linux may be a sounder option - especially if network protocols are involved. I believe that Linux implementations are more reliable and easier to tweak.

Share Improve this answer Follow

answered Nov 28, 2008 at 21:45

kgiannakakis
104k ● 28 ● 161 ● 197

> "When buying a board you also need to buy the SDK" Seriously ? Why dont you just configure a proper kernel / DT ? I never ever bought any "SDK" for Linux, and dont see why I ever should. The kernel/DT is board-specific anyways, so - unless you have a standard board - you'll have to customize it anyways. – Enrico 'nekrad' Weigelt Sep 22, 2016 at 14:34

8

With Linux you are **never on you own** and you are never dependent on one single entity to provide permissions. There are many support options and you have the freedom to choose your support options for any part of the system through many competing sources.

With Windows CE you must adhere to the license and restrictions as set forth in the complex license agreements that must be agreed to. Get a lawyer. With windows CE you have only one proprietary source for OS support and you will proceed only as they see fit to support and provide what you need. You may not agree with their position, but will not have any recourse but to bend to what they prescribe. The costs of incremental components, modules, development kits, licensing, and support tend to pile up with proprietary platforms. In the longer term, what happens when the vendor no longer desires to support the platform and you do not have the rights to support and distribute it yourself? What happens when the vendor moves to newer technology and wants you to move along with them even though you may not be ready to make the move? $$$

Our experience with Windows solutions in general is that they tend to become more expensive over time. What was originally considered lowest TCO gravitates quickly towards and solution that is encumbered and costly to maintain and support. Licenses have to be re-negotiated over time and the new technologies, often unneeded, are forced into the picture at the whim of the provider for the sake of THEIR business needs. On top of that, the license agreements are CONTINUALLY changing--get a lawyer.

With Linux you have the freedom to provide in-house support and expertise without being encumbered against distributing the solution as you need. You also have the freedom to continue to use and support technology that original providers no longer want to support. Having the source code and the RIGHTs to do with it what you want (GPL, LGPL) is a powerful attractor when it comes to business continuity and containing costs while providing access to the very latest technologies or technologies that fit your needs.

Share Improve this answer Follow

answered Jun 2, 2011 at 15:09

zman58

81 ● 1 ● 1

7

I have developed network drivers that work both on RT Linux (to be more specific, Linux preemptive kernel with RT patch) and Windows CE. My experience was windows CE was more stable in terms of real-time response. Frame timings also showed that windows CE had less jitter.

On RT Linux, we had all sorts of problems. For example, when user moved the mouse; our frames were being delayed. Guess what, certain variants of x-windows disable interrupts. You may also feel that you are safer on console screen only. If you have VGA frame buffers enabled, you are doomed again. We had only one problem with windows CE in terms of jitter again. The problem happened when the USB controller was set to an incorrect mode in the BIOS and windows CE was using lots of time for polling.

To be honest, windows CE had more support. On Linux, you are on your own. You have to read every possible mailing list to understand what problems you may hve.

Share Improve this answer Follow

answered Jun 20, 2009 at 22:01
user108570
129 ● 1 ● 3

5

> a partially customised OS

Is much easier to achieve if the OS is open source (and you have the expertise).

Share Improve this answer Follow

answered Nov 28, 2008 at 19:55
Ali Afshar
41.6k ● 12 ● 97 ● 110

2 CE ships with enough source (something like 95% last I checked) to modify or replace almost anything you want. – ctacke Nov 28, 2008 at 20:01

ctacke: That is interesting, how are modifications licensed? – Ali Afshar Nov 28, 2008 at 20:09

4 I beleive it's called the Microsoft Permissinve License (en.wikipedia.org/wiki/…). Basically you can make and use

modifications as you see fit without providing those changes to anyone and use the derivative in a commercial project if you like – ctacke Nov 28, 2008 at 20:21

2

Android is a good option for some embedded systems. (it's linux based)

- You have many experts that are able to develop on this system.
- You have access to many libraries in java or C.

but it uses lot of memory and energy.

What we often forget with paid / licenced software is that you have to deal with licenses. It takes time and energy! Then you have to track if you pay it correctly. It involves many different people with different skills and it costs in decision.

This cost is often not included in the studies that show that open-source/free is more expensive than paid software.

With "free software" it's way easier to deal with licenses and you spend less time on dealing with these issues. Personally I prefer to avoid unnecessary communications with your legal / financing team every time you change some pieces of the software.

Share Improve this answer Follow

answered Oct 9, 2013 at 8:45

Gabriel Klein
534 ● 4 ● 7

1 with "embedded for industry". Microsoft essentially give it away for free making them more competitive and its much less hassle. They also give you a lot of the sources. However they've pretty much been ignoring the market space for the past four years so they've probably lost a lot of their advantage in this space. – Quibblesome Oct 9, 2013 at 10:10