Linux vs. Standards

by Alexandru Lazăr

Email: std@cwazy.co.uk
Web: http://donkey.50webs.com

September 8, 2005

1 Running by standards

Standards are one thing we can barely live without today. Most of our life is standardized, and we often impose ourselves the usage of a standard, even without knowing it. The coffee you dink in the morning is a self-imposed standard. The tea I drink in the morning is a standard I impose myself. My morning would seem strange if it didn't start with a mug of Royal Ceylon tea, in the exact same way my nights would seem dull without Irish Cream. Even the recipes of these brands are standard. I already know how the Royal Ceylon tea should taste, because I know what it contains. And you will find standards anywhere – in the car you drive, in the computer you use every day, and even the language I use to typeset this text is a standard. TeX.

Why do we need standards? There are more reasons, and many depend on the domain where the standards apply. For example, in Romania, where I live, all the plugs supply 220V. If they weren't standardized, you could end up needing a whole bunch of adaptors for your machines and, with enough bad luck, you may even fry a few. In England, all cars drive on the right to avoid traffic jams. But since I'm talking about Linux and standards, I'll try to keep the range a bit more narrow. I don't really care where the steering wheel is on my Linux box. Oh, wait, it doesn't have any, maybe another kernel module should be written for it ;-).

Without trying to make an encyclopedia of reasons, here is why standards are so needed when talking about computers:

- **Interoperability.** This is one of the most serious reasons, and the top example is the one featuring TCP/IP and endianness. Computer processors fall into two large categories: little endian and big endian. The difference is how the data is stored. When storing a sequence of bytes, some processors (like the Motorola 68000 family) store the most significant byte first. For example, on a 16−bit big−endian CPU the hexadecimal value A3FF2DE1 (which requires 32 bytes to store) would be stored: A3 FF 2D E1 at consecutive memory locations. However, others, like the x86 family of CPUs, store it the other way: E1 2D FF A3. It's not a problem on a single machine (in fact, some processors can switch modes, and become either big endian or little endian). But, in the early days of TCP/IP, this was a problem. A sequence of bytes sent by a little-endian-operated machine would end up reversed on a big-endian-operated machine. Anecdotically, his was discovered when trying to port UNIX from the PDP-11 (little-endian) to the x86, and when the PDP-11 sent the string UNIX, the x86 ended up with NUXI. The solution was easy: TCP/IP standardized a single method of encoding binary data in a packet. All data sent using TCP/IP is big-endian, and everyone receiving it can expect it to be like this and convert it if necessary.
- **Portability**. When the computer market consisted of 4 to 6 computers, this wasn't a problem. However, today things have changed radically. A computer no longer serves a single purpose, and you may expect a software package to be available on more architectures. It's obvious that you cannot expect all computer architectures to be similar. The first wide-used solution was the C language. Instead of writing all software using assembly-language, specific to every computer architecture, a minimum of it is written specifically for the machine, and the rest is written in a more high-level, universal language. The compiler would translate C code to machine-specific code and, if the code is cleanly written, in some cases a simple recompilation on a platform is enough to run software initially written for another one. The C language itself has been standardized a few times, so that different implementations of C could be unified, allowing for an even greater portability.

- **Mobility.** While the usage of laptops is becoming more and more of a trend, you can still expect one to be forced to use a computer he doesn't own. Of course, you cannot expect all computers to behave the same, but some things should remain the same. In the case of Unix for example, which happens to be a standard itself, it's safe to assume that you can get a list of the files in a directory with the ls command or that you can view a file on the standard output using the cat command. To a certain extent, you can expect all desktop environments to provide at least a simple text editor, an image viewer if the hardware underneath allows it and so on. Sure, diversity is one of the reasons why we all love computers (and a direct consequence of innovation), so it should not be discouraged, but at least it should not go as far as giving you a bottle beer when you asked for a cup of coffee.
- **Adaptability.** For many of us, adapting to a new working environment is not something too hard to do. For example, despite being a fan of emacs, I easilly got used to NEdit when I had to, and while being an astute fvwm user I didn't mind working on a Gnome environment. However, this is not the case for everyone. Again, diversity is not to be condamned, but an environment should not work against the one who is trying to use it. It is why many standard Unix distributions (Solaris, HP-UX, AUX, you name it) ship with CDE as a desktop environment. CDE has been a de facto standard for a long time. When you don't have others, de facto standards will suffice I guess. Some (like Solaris) even include other environments on the standard CDs you get when obtaining the operating system, and you're free to install them if you want, but you still get the choice of CDE. This way, if one is suddenly forced to use a HP-UX workstation instead of a Solaris workstation, he will still find the desktop environment he is used to. As a sidenote, some commercial Unix developers announced they would phase out CDE replacing it with Gnome. HP returned to CDE eventually.
- **Strong base for future development.** Standards provide a way of "freezing" technology to a certain point. When an environment has reached maturity and further improving it require a great deal of effort, or maybe a change of perspective, having a stable foundation to rely on is a must. Imagine the hell Unix desktops would have been now if, instead of having the X protocol and its various (bloated, slow, whatever) implementation, we would have had a few different protocols. While there were certain attempts at it (Sun's NeWS for example), they failed to make it to the "big outside".

These are the strongest arguments I can give. While I do have many others on my mind, the ones I listed above are the ones that cover the widest possible range and apply to most users.

2 The Unix standard

Earlier, I remotely mentioned Unix was a standard. Rise two fingers all of you who thought it was an operating system.

Well, historically speaking, you are half right. Sintactically speaking, you are only 33.33% wrong. Unix used to be an operating system. Now there are more Unices. A few dozens to be remotely precise. Unix is also a standard of the Open Group. And, Unix is also a trademark, which is why we don't have a few dozens of systems called Unix. We just have a few Unices (because the Unix name itself changed its owner several times) having the name of Unix, and a lot of other operating systems who comply Open Groups's standards. Some of them (like Solaris) are comply those standards well enough to be certified as Unices.

Is Linux a certified Unix? The short answer is no. The long answer is no, but there are standards for Linux as well. The even longer answer is: no, and despite the fact that there are standards for Linux as well, they have been developed by a small number of companies, so nobody seems to bother except for those who agreed upon them. These standards were published under the name of Linux Standard Base or LSB. Many laughed at it, asking if it took them long to decide upon what is the Least Significant Byte (also abreviated as LSB). The reason? It didn't solve too much.

3 So what's wrong with LSB?

The Linux Standard Base seemed like a very nice initiative at first. However, it ended up as something on the narrow margin between failure and acceptance. Many of the standards it describes were already long being implemented even without being standardized, simply because they happened to be the best solution. Other standards seemed to bluntly reflect the marketing policies of those who developed them, especially since the sugestions of those who weren't involved (Debian being the biggest name) were easily discarded. One example is the RPM format, which was standardized as the universal package format for software distribution. Many voices criticized this, since RPM doesn't allow for the general compatibility and platform-independence offered by other package managers and, at the same time, it lacks many features which are present in other package managers. Many voices rised against this choice. While many of those who revolted were not really getting the point (RPM isn't supposed to be the package manager of all LSB-compliant distributions, it's just the package format supposed to be used by 3rd party software vendors), the fact that an inferior solution was chosen remains.

The LSB also fails to address other problems, which seem to be even more bothering.

4 Where do we need standards?

Freedom of choice has always been an issue with open source. Linux is so special because it is open source. And most of the software you'd expect to run on a Linux platform is open source. In my case, the only closed-source piece of software I'm using is NVidia's binary driver for my graphics card.

One thing that has been said about standards is that they harm the freedom of choice. They don't. A standard doesn't meen you absolutely have to use a certain program, just that using it would guarantee its interoperability, its integrity and its coherence on all platforms that run it. You may use whatever alternative you want, but they won't provide these guarantees.

Standards in open source are rather hard to impose. When we're talking about standards at a small scale, it seems less troubling. For example, Free Desktop's standards are already being used by Gnome and KDE, the big players in Linux desktop environment. However, when talking about a scale as large as the one a complete Linux distribution implies, things do change radically.

4.1 The desktop environments

Linux (and Unix in general) offer a very wide range of choices in this area. This is not something that has to do with Linux in particular, but more with choices in X11's design.

X11 never tried to impose anything except for a protocol to those who were developing. This means that the X desktop environments offer a great deal of diversity. Desktop environments solve many problems, because they don't roll out just a workspace, they also bring some usual applications, hardware configuration utilities and so on. However, they do have some problems:

4.1.1 Coherence

The X11 environment doesn't even impose a widget set. While the Athena widgets and the twm (Tabbed Window Manager) ship with it nowadays, they are hardly what I call a support for a good desktop environment. Many alternatives were developed, but they are not similar in many places. The most eye-cutting of them all is aspect. You will instantly know a Motif application from a GTK application, and while Motif is a standard of Open Group, it is not widely used in Linux.

Some toolkits implement different approaches to how widgets look and perform. Using the scrollbars in applications with Athena widgets is different from using scroll bars in GTK-based applications. Some toolkits may refuse to work with the scroll wheel of your mouse. And all these, if you can live with the fact that you may have applications that look in totally different ways.

This is having a negative impact on the user. It's like you were driving a car, and the steering wheel's shape would suddenly change from round to square, the seats would be painted red while the floor would be painted blue and the acceleration pedal would look like a brake pedal. All these give a feeling of unreliability and inconsistence.

Changing this sittuation wouldn't be that difficult. In fact, my KDE/Qt-based apps look just like my GTK-based apps on my system. I easily achieved that by applying the same theme for these toolkits, since many modern widget sets have support for theming. However, this is left to be the packager's job. If major toolkits (at least Qt and GTK) could agree upon one common theme to use by default, the system would get a much more consistent look out-of-the-box.

4.1.2 Common applications

All desktop environments come shipped with at least a small number of basic applications. However, despite being called "common", they aren't so common.

Of course there's not much to worry about when you write a program to handle, say, network configuration. However, just making the applications themselves look and behave the same among the desktop environments would be a good thing. Closing the general gaps in terms of usability between major desktop environments would not harm the environments themselves (they can still stay unique without giving a new example of how a Set Wallpaper dialog should look like). Certain points really need addressing – for example, if you don't know how the text editor KDE just installed on your system is called, you may end up not being able to edit your files if you don't have a menu entry. Having a standard symlink named "text-editor" that is linked against whatever the default text editor is just one of the posibilities.

4.1.3 The environments themselves

Right now, there are two big players in this game: Gnome and KDE. While I'm not a big fan of either (I use fvwm), their leading position is well-deserved.

Since they are now both widely accepted, many distributions end up as being either KDE-centric or Gnome-centric.

In the case of commercial Unix distributions, CDE has been a de-facto standard. It wasn't really imposed, it's just that it became so widely accepted that commercial Unices started shipping with it. The result was that changing the Unix vendor would normally have a minimum impact on the desktop. Sometimes, this is not the case with Linux. Changing Ubuntu for KUbuntu not only causes a new desktop environment to roll out, but also another PDF viewer, another media player, another IDE maybe and so on.

Standardization in this area wouldn't touch old-time users. Gnome fans would remain Gnome fans and KDE fans would remain KDE fans. However, new users would at least have a starting point when choosing from the incredible multitude of environments.

Choosing one would obviously be a problem. Gnome is completely GPL, but it's often chaotic, unsupported and unstable. KDE on the other hand is always stable and consistent, but it's blamed for being horribly slow and, being based on Qt, there are some licensing problems.

Individual window managers don't count up as desktop environments. Sure, there are window managers so capable that they can end up not needing something like Gnome (take the case of WindowMaker or fvwm, for instance). However, they don't provide a complete environment – they just provide window management and maybe some GUI-related configuration options.

4.2 Multimedia

Multimedia is becoming a very serious issue, as Linux is being more and more used as a desktop platform. In this area, I believe much could be learned from Apple's Mac OS, which is a Unix for desktops, in the same way that Linux is. Okay, maybe Linux is in fact an almost-Unix for the desktops but that doesn't change it.

The fact is, Linux still doesn't have a standard video player, a standard media player, a standard web browser, a standard sound server and so on. While experienced users have no problems choosing one, beginners get confused. Totem? Mplayer? VLC? Xine? Desktop environments try to roll out their own, even when it's so bad that it would actually be better to use the competitor's.

The lack of standards when it comes to general, high-level applications issues is not necessarily a big hassle. Most distributions will install either just one, or install them all, so after a few minutes of experimenting, the user could choose his own. But what do we do when we're dealing with more low-level apps?

I recently came across this problem when a friend of mine asked me for help. He would need a client-server application. The client part of the application needed to ring at certain given moments, and by 'ring' I mean make a sound. In the end, this proved to be the most tricky part of it. Some of those who would use it had ARTs installed. Others used esound. One was working with software that needed low-latency so he was using the JACK sound server. Others used none at all and simply used mplayer to play their sounds. Others didn't like mplayer but they just used XMMS since they didn't rely on sounds anyway. Getting the software to play something on every computer in that network without having the user mess up specifying a command himself took us about two days to figure out (and write some 200 lines of code at most), while coding the rest of the application took less than a day and involved way more code.

The point is, the only standard in multimedia seems to be OpenGL (which is actually used in LSB), but there's much more about multimedia out there. Choosing standards in all areas would benefit end users.

4.3 System applications

Application executables have their own special places in Linux. The problem is, there are too many places.

One may often find it confusing to see how some apps go in /bin, others in /usr/bin, others in /usr/sbin, /usr/X11R6/bin, /usr/local/bin and so on. It's very difficult to manage, and this is often left to the decision of the one who writes the software and its install target in the makefile, or to the decision of the packager. This makes applications dificult to manage and, at times, can lead to confusions if they end up installed in a directory that is not in the PATH of the one trying to execute it. Getting a general guideline (indispensable executables in /bin, X11 aplications in /usr/X11R6/bin and so on) would help managing all these files. A universal way of installing binary packages would also be a good choice. This is already present in LSB, but the acceptance of RPM is problematic. A completely new package manager is more likely to be universally accepted than Red Hat's solution.

There are certainly more places where standards could be applied (the /dev hierarchy, usage of the /proc system and so on). However, these have less impact on the destkop user than the ones listed above and I'd rather avoid them.

5 The but-s

The first person who saw this document read it carefully two times and replied me with a single phrase: "Alex, you're kindda nuts, you know?"

The first objection you may have is that applying such strong guidelines when it comes to Linux would kill off most of the distributions. Well to be sincere, I'm fine with that. Distrowatch features hundreds of Linux distributions. I myself use quite a few (Gentoo at home, Programmer's Backpack Live CD when I need to program somewhere else, GNUStep Live CD for learning GNUStep and Objective-C) but this doesn't change a fact: there are many distributions that just do the same thing. Those who are really original are the big players – Debian, Fedora, SuSE, Gentoo, FreeBSD, NetBSD, OpenBSD (for servers), and there are a few distros with a certain focus. Others do nothing but maybe provide a slight change in wallpaper and appearance. What's the point? I'd rather have 40 good distros, with some having a certain, particular purpose, instead of having 400 of which 40 are good, 300 do the same thing and 60 are plainly bullsh...um, bad.

These guidelines don't make all distributions the same. They just change focus on inovation and/or originallity instead of bright wallpapers and fuzzy themes of the same desktop environment.

Another question I was asked was "Are you expecting people who've been fighting vi vs. emacs for twenty years to settle in Gnome vs. KDE?". This question is missing the point. A standard doesn't mean that one is better than its competitors. When the Open Groups (Open Software Foundation at the time) adopted Motif as a standard instead of OpenLook, it was the most obvious example. Call me subjective but OpenLook was not nearly half as ugly and slow as Motif – it just wasn't supported anymore. It would be the same now: nobody would impose anyone to install KDE if KDE were chosen as a standard. KDE would just be the one who guarantees certain things, in the same way that, for example, choosing emacs as a standard text editor would guarantee you that you can run your Emacs-LISP scripts anywhere.

6 Ending

Summing up, there is already one example where standards led to a solid desktop environment. Apple's desktop environment on Mac OS has won many acclaims among developers, even among those who didn't really fancy Apple. Of course, the environment is not even half as dynamic as the Linux environment is. However, shifting dynamism towards more useful purposes than reinventing the wheel, only with a different paint each time, would be a good thing.

Linux is a very good environment for innovation – it is a dynamic environment with many users around the world. Simply setting a stable base for it would be enough to give it a big, big boost. And general consistency is a first step. Thank you.

7 Credits

I don't usually like big thank-you-s in the end, but this time I really feel like doing it. So, in no particular order, if you enjoyed reading, you should thank:

- Sabina Munteanu for asking a lot of (what seemed to her as) stupid Linux-related questions when they were actually questions about "how do I do that? It says this in the manual but it doesn't work at all :-D)
- Ron Eddington for carefully pointing me out that CDE was never adopted as a standard, it just ended up acting like one.
- Written with TeXmacs should say it all ;-)