A Plea to Software Vendors from Sysadmins - 10 Do's and Don'ts

view issue

by Thomas A. Limoncelli | December 22, 2010
Topic: System Administration

What can software vendors do to make the lives of sysadmins a little easier?

THOMAS A. LIMONCELLI, GOOGLE

A friend of mine is a grease monkey: the kind of auto enthusiast who rebuilds engines for fun on a Saturday night. He explained to me that certain brands of automobiles were designed in ways to make the mechanic's job easier. Others, however, were designed as if the company had a pact with the aspirin industry to make sure there are plenty of mechanics with headaches. He said those car companies hate mechanics. I understood completely because, as a system administrator, I can tell when software vendors hate me. It shows in their products.

A panel discussion at CHIMIT (Computer-Human Interaction for Management of Information Technology) 2009 discussed a number of do's and don'ts for software vendors looking to make software that is easy to install, maintain, and upgrade. This article highlights some of the issues uncovered. CHIMIT is a conference that focuses on computer-human interaction for IT workers—the opposite of most CHI research, which is about the *users* of the systems that IT workers maintain. This panel turned the microscope around and gave system administrators a forum to share how they felt about the speakers who were analyzing them.

Here are some highlights:

1. DO have a "silent install" option. One panelist recounted automating the installation of a software package on 2,000 desktop PCs, except for one point in the installation when a window popped up and the user had to click OK. All other interactions could be programmatically eliminated through a "defaults file." Linux/Unix tools such as Puppet and Cfengine should be able to automate not just installation, but also configuration. Deinstallation procedures should not delete configuration data, but there should be a "leave no trace" option that removes everything except user data.

2. DON'T make the administrative interface a GUI. System administrators need a command-line tool for constructing repeatable processes. Procedures are best documented by providing commands that we can copy and paste from the procedure document to the command line. We cannot achieve the same repeatability when the instructions are: "Checkmark the 3rd and 5th options, but not the 2nd option, then click OK." Sysadmins do not want a GUI that requires 25 clicks for each new user. We want to craft the commands to be executed in a text editor or generate them via Perl, Python, or PowerShell.

3. DO create an API so that the system can be remotely administered. An API gives us the ability to do things with your product you didn't think of. That's a good thing. System administrators strive to automate, and automate to thrive. The right API lets me provision a service automatically as part of the new employee account creation system. The right API lets me write a chat bot that hangs out in a chat room to make hourly announcements of system performance. The right API lets me integrate your product with a USB-controlled toy missile launcher. Your other customers may be satisfied with a "beep" to get their attention; I like my way better (http://www.kleargear.com/5004.html).

RECENTLY ON SLASHDOT

- Please Step Away from the ASR-33!
- You're Doing it Wrong
- Visualizing System Latency

RELATED CONTENT

TESTABLE SYSTEM ADMINISTRATION
Models of indeterminism are changing IT management.
Mark Burgess

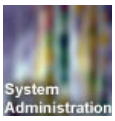
SYSTEM ADMINISTRATION SOFT SKILLS
How can system administrators reduce stress and conflict in the workplace?
Christina Lear

COLLABORATION IN SYSTEM ADMINISTRATION
For sysadmins, solving problems usually involves collaborating with others. How can we make it more effective?
Eben M. Haber, Eser Kandogan, Paul Maglio

VIRTUALIZATION: BLESSING OR CURSE?
Managing virtualization at a large scale is fraught with hidden challenges.
Evangelos Kotsovinos

BROWSE THIS TOPIC:

SYSTEM ADMINISTRATION



4. DO have a configuration file that is an ASCII file, not a binary blob. This way the files can be checked into a source-code control system. When the system is misconfigured it becomes important to be able to "diff" against previous versions. If the file can't be uploaded back into the system to re-create the same configuration, then we can't trust that you're giving us all the data. This prevents us from cloning configurations for mass deployment or disaster recovery. If the file can be edited and uploaded back into the system, then we can automate the creation of configurations. Archives of configuration backups make for interesting historical analysis.[1]

5. DO include a clearly defined method to restore all user data, a single user's data, and individual items (for example, one e-mail message). The method to make backups is a prerequisite, obviously, but we care primarily about the restore procedures.

6. DO instrument the system so that we can monitor more than just, "Is it up or down?" We need to be able to determine latency, capacity, and utilization, and we need to be able to collect this data. Don't graph it yourself. Let us collect and analyze the raw data so we can make the "pretty picture" graphs that our nontechnical management will understand. If you aren't sure what to instrument, imagine the system being completely overloaded and slow: what parameters would we need to be able to find and fix the problem?

7. DO tell us about security issues. Announce them publicly. Put them in an RSS feed. Tell us even if you don't have a fix yet; we need to manage risk. Your PR department doesn't understand this, and that's OK. It is your job to tell them to go away.

8. DO use the built-in system logging mechanism (Unix syslog or Windows Event Logs). This allows us to leverage preexisting tools that collect, centralize, and search the logs. Similarly, use the operating system's built-in authentication system and standard I/O systems.

9. DON'T scribble all over the disk. Put binaries in one place, configuration files in another, data someplace else. That's it. Don't hide a configuration file in `/etc` and another one in `/var`. Don't hide things in `\Windows`. If possible, let me choose the path prefix at install time.

10. DO publish documentation electronically on your Web site. It should be available, linkable, and findable on the Web. If someone blogs about a solution to a problem, they should be able to link directly to the relevant documentation. Providing a PDF is painfully counterproductive. Keep all old versions online. The disaster recovery procedure for a 5-year-old, unsupported, pathetically outdated installation might hinge on being able to find the manual for that version on the Web.

Software is not just bits to us. It has a complicated life cycle: procurement, installation, use, maintenance, upgrades, deinstallation. Often vendors think only about the use (and some seem to think only about the procurement). Features that make software more installable, maintainable, and upgradable are usually afterthoughts. To be done right, these things need to be part of the design from the beginning, not bolted on later.

Be good to the sysadmins of the world. As one panelist said, "The inability to rapidly deploy your product affects my ability to rapidly purchase your

products."

I should point that that this topic wasn't the main point of the CHIMIT panel. It was a very productive tangent. When I suggested that each panelist name his or her single biggest "don't," I noticed that the entire audience literally leaned forward in anticipation. I was pleasantly surprised to see software developers and product managers alike take an interest. Maybe there's hope, after all.
Q

REFERENCES
1. Plonka, D., Tack, A. J. 2009. An analysis of network configuration artifacts. In *Proceedings of the 23rd Large Installation System Administration Conference* (November): 79-91.

ACKNOWLEDGMENTS
I would like to thank the members of the panel: Daniel Boyd, Google; Æleen Frisch, Exponential Consulting and author; Joseph Kern, Delaware Department of Education; and David Blank-Edelman, Northeastern University and author. I was the panel organizer and moderator. I would also like to thank readers of my blog, www.EverythingSysadmin.com, for contributing their suggestions.

LOVE IT, HATE IT? LET US KNOW

feedback@queue.acm.org

Thomas A. Limoncelli is an author, speaker, and system administrator. His books include *The Practice of System and Network Administration* (Addison-Wesley) and *Time Management for System Administrators* (O'Reilly). He works at Google in New York City and blogs at http://EverythingSysadmin.com.

Originally published in Queue vol. 8, no. 12—
see this item in the ACM Digital Library

COMMENTS

Peter Jeremy | Thu, 03 Mar 2011 20:27:04 UTC
A couple more don'ts: If your software interacts with hardware, it should accept that hardware occasionally fails and will need to be replaced. If I'm using your LVM to provide disk mirroring, I don't want the disk replacement procedure to begin "unmirror the logical volume and destroy the virtual device containing the failed disk" (eg Solaris Volume Manager) - this doesn't scale with virtual devices made up of multiple physical devices.

Also, your software shouldn't assume that there is only one instance of itself running on a host and rely on magic, undocumented sockets, ports or whatever, even for short periods of time. Oracle rdbms fails this test - starting multiple rdbms instances simultaneously leads to deadlocks.

Finally, do have your own configuration file, don't configure yourself based on the configuration for some other function, no matter how similar. You can have it default to that if you like but make it overrideable. Solaris IPMP fails this.

Rigs | Tue, 01 Feb 2011 15:23:03 UTC
Regarding point 6 I remember customizing a very lightweight proactive monitoring tool (before Quest Software convert it to commercial) mixing perl, python, awk and shell scripts called Big Brother (now BB4).

It was really helpful.

Paul Anderson | Thu, 27 Jan 2011 08:07:11 UTC
Interesting to see what's changed (and what hasn't) since 1992 :-) http://homepages.inf.ed.ac.uk/dcspaul/publications/Workshop_Report.pdf

Mark | Tue, 04 Jan 2011 14:03:09 UTC
Can't entirely agree with point 2 - I'd say have both a GUI and a CLI. Scriptability is good, but so is discoverability, particularly for small environments. The more hoops need to be jumped through to learn a product, the less likely it is a reseller will embrace it.

Robert Wipfel | Tue, 28 Dec 2010 15:39:36 UTC
Many of these recommendations appeared in Eric Raymond's "Art of Unix programming", about Unix (versus Windows) philosophy in general:

http://www.faqs.org/docs/artu/

I would add two more; (1) for remotable APIs intended to support mass operation: do allow SSH and log everything (2) for scribbling on the disk; see Linux Standard Base (LSB).

Dan Cross | Mon, 27 Dec 2010 09:09:54 UTC
I'll go one further on configuration and automation:

DO (if it doesn't compromise security) use a pre-existing, text-based scripting language like Tcl or Python for configuration data. Using a full language (again, if it doesn't compromise security) opens up new worlds of possibility for a product.

DO provide bindings for a scripting language like Tcl or Python if you provide an API. This avoids the proliferation of badly constructed third party scripting language bindings that inevitably pop up. If you provide a remotely accessible API for administration, similarly wrap the protocol in some sort of module loadable in a scripting language; preferably, use the same language you use for configuration.

Christian Drexler | Fri, 24 Dec 2010 19:32:50 UTC
I have another one: respect and use the packaging mechanism of the platforms you are supporting! Don't give me a tarball containing a badly

written script that copies stuff to random places but build a nice deb/rpm/SysV-pkg that installs without being --forced.

mathew | Fri, 24 Dec 2010 16:52:09 UTC
For server software on POSIX systems, include a standard /etc/init.d script to start and stop the software.

I'm looking at you, IBM.

kregg | Fri, 24 Dec 2010 11:29:44 UTC
Well, this is a good insight to a sysadmin's life. I'm going to use this article to help me with my work and offer my client a decent solution in terms of all the points listed above. :)

Pascal Sartoretti | Fri, 24 Dec 2010 10:56:30 UTC
@Greyed : if your config files are in XML, you can always use a "diff" utility which doesn't take into account whitespace. I use DiffMerge for this.

Pascal Sartoretti | Fri, 24 Dec 2010 10:54:55 UTC
A very good list !

For item #4 however (configuration files in ASCII and not binary form), ASCII can be problematic for us Europeans with accentuated characters : you never know what is the encoding (UTF-8 ? ISO-LATIN-1 ? other ?). Having the configuration file in XML is the obvious solution.

In addition, XML solves time-consuming issues like empty lines, line termination (LF? CR? CR/LF?), etc.

Dirk | Fri, 24 Dec 2010 10:11:34 UTC
Please please check your software to run well under non-admin account! Do not write into areas of registry and file system where a normal user is not allowed to do!

Greyed | Fri, 24 Dec 2010 08:27:05 UTC
You missed something on the ASCII configuration file.

DON'T make it one freakin' long line. DO use line breaks, commenting and logical indention.

I don't know how many times I've had to deal with a config file in XML which is a single line. Makes revision control worthless. A similar problem to the time I wanted to use revision control on a document I was writing in OOo. I could unzip the ODF but the document XML was a single... freakin... line! XML parsers don't care about whitespace, humans and revision control do!

un1xl0ser | Fri, 24 Dec 2010 04:27:54 UTC
This reads like a checklist that I use when reviewing vendor products. It is pretty much the same for everything from software to appliances and having a standard list of issues (customized to your environment of course) so you can actually spend your time focusing on what the software should do. If a vendor fails this up front, you have your work cut out for you in the future. I would also say DO support SNMP (for monitoring and statistics) also. For appliance-like solutions, recovery media is a must for EVERY revision (this is mainly for fake appliances).

bitmap | Fri, 24 Dec 2010 04:05:03 UTC
I totally agree with Crissy K, an offline documentation PDF or even HTML ZIP is mandatory. Such documentation can be stored away with the installation files or media, and remain available even if the vendor goes out of business and website is no longer available. Similarly, installation files and patches should be downloadable for historical purposes or installation in non-Internet connected systems. A "download manager", "online installer", or "active update" can be optional, but there MUST be another way to get the documentation, installation or updates to an offline system.

Crissy K | Fri, 24 Dec 2010 02:57:54 UTC
"10. DO publish documentation electronically on your Web site. It should be available, linkable, and findable on the Web. If someone blogs about a solution to a problem, they should be able to link directly to the relevant documentation. Providing a PDF is painfully counterproductive."

Last sentence is wrong. Reason: some computers will never see the public internet and a downloadable PDF of documentation that can be transported to that private network is mandatory - in addition to the html manual on the internet.

Skaperen | Thu, 23 Dec 2010 23:12:37 UTC
DNF ... that should be "don't mandate ...". There are times I specifically do NOT want to have to add users to /etc/passwd, and so having an alternate way to authenticate is good. And I don't want to have to run a database to do that, either. Simple files are often best. A BerkeleyDB file might be fine. A file per user can work. A flat file in the same format as /etc/passwd also works well. But if you want to use /etc/passwd, LDAP, or Postgres, all that should work, or allow a SO/DLL binary plugin via documented API to extend. An example of FOSS software that does this reasonably well is Dovecot. It has plenty of choices that should make most admins happy (at least I found what I like in there).

Adrian Bridgett | Thu, 23 Dec 2010 23:06:42 UTC
A very nice list - we have to hope that product managers listen rather than decide that we want "flash" graphics or a blackberry version.

One thing I'd add is a changelog in the release notes. I want to know what you've changed, what you've fixed, what might break, what I'll have to do differently. Each time I'm told "oh, can you reproduce the problem on the latest version" my response is "show my something in the changelog which shows that I'm not wasting my time". 99% can't. Or for Dell BIOSes, I can only get the _most_ recent change. How pointless.

Darael | Thu, 23 Dec 2010 22:42:09 UTC
There seems to be some debate here about #2, primarily based on a reading of it as "Give us a CLI, not a GUI". I don't read it like that. I read it with emphasis on "the". That is to say, don't force us to use a GUI for anything. By all means provide one, but everything - /everything/ - that

your admin interface does should be doable from the command-line. We probably /do/ want a GUI to set things up the first time, but having done initial setup for one machine that way (if we do!), we want to be able to automate the rest away from the command-line.

As for unix paths, FHS says we use /usr and similar unless we're not distribution software, in which case we use /opt. "Non-distribution software" seems best interpreted as "not in the OS provider's package archive". The point of #9 wasn't "use /usr and /etc", it was "Stick to the standard locations for the target system and be consistent". Those /are/ different things, as illustrated by the comment about config files in /etc and /var.

Neil Cherry | Thu, 23 Dec 2010 22:40:58 UTC
I see a number of folks talking about having a GUI, how many systems do you install at a time?. I like what the one person said, CLI with a GUI on top. When I need to configure/reconfigure systems (more than just a few) the the GUI just gets in my way. I recall the fiasco with WellFleet/Bay where to configure their routers you need their GUI system. It was pretty but pretty useless when your remote connection went down. That kind of thinking is why they no longer exist today. If you tell me that I have to use a web interface to configure a dozen complex systems then I'll probably whip out a Perl script to do the same in the time it took you to do 6. If there is no way to script it then expect to get billed for time wasted.

Add one more to the "Don't" list: Don't allow your developers to be willy-nilly with the error messages. If error messages start with % then they all do! Consistent formatting is extremely important. How else do expect the admins to figure out important error messages from the less important information when the amount generated is on the order of megabytes.

Ralph Weaver | Thu, 23 Dec 2010 22:34:41 UTC
Yes, yes, and more yes.

Mark | Thu, 23 Dec 2010 22:30:09 UTC
If #3 is done, then #2 will be done.

The problem with software development is:

1. Too many people who are doing it are not very good at it. They only think about Screen and database. So all the logic goes there. Doing an API that could be used for scripting or a system interface is now a pain because there is not a good foundation.

2. Management and Time to market. Although creating an API is not that much effort, TTM does not allow it and neither does management because "the users didn't ask for it."

DNF -this really has nothing to do with Java ... except if running on windows and in the browser. Windows security is PITA unless you are using Microsoft's products. As for SaaS, unless your organization uses one of the SSO standards, there is not much they can do. SSO (single-signon) is a PITA. And trying to have external vendors access your security system ....

Anyway, back to Java. You are focusing on the wrong thing. Java is used quite extensively so run across issues because this, not because of Java itself. Try setting up a .NET app on Linux or Unix. ;) (of course unless they used Mono). Joking aside, try talking to a .NET developer. They seldom seem to think outside the box. I spent a few hours last week trying to get a senior developer to see past his Microsoft centric thinking.

ARos | Thu, 23 Dec 2010 22:28:49 UTC
- DO provide download URLs for your programs that do not require authentication and cookies (Oracle violates this best practice).

Eamon | Thu, 23 Dec 2010 22:13:49 UTC
Excellent article! I posted a link to it on my blog - blog.eamonbauman.com.

John | Thu, 23 Dec 2010 21:49:09 UTC
Build the GUI configuration that can read and write the CLI file. This makes it easy to initially configure the app then use the CLI to install the rest.

Michael Martell | Thu, 23 Dec 2010 21:36:54 UTC
One point I disagree with on your suggestions - Do make BOTH a GUI and a Command line interface available. While the point about copy and paste commands is well taken, learning yet another arcane set of command line jargon, syntax and options for every piece of software makes delegating administrative tasks, which are not always routine or repeatable, exceedingly difficult. With a GUI, I can generally give a lower level support tech the ability to perform some options, setups and other tasks without the detailed training, experience and frequent mistakes involved in learning a command line interface, and with much better limitations on the disastrous effects of a mistaken procedure copied without understanding from the wrong page in the documentation. I think overall the article espouses some excellent points, but I do think there is room for a few added points (such as publishing the documentation on the web site AND including it in a PDF format on the install media or better, within the installation process itself so that a copy is where you need it most, even when web access is not. Overall I think the suggestions in the article would be a great beginning to a standard by which developers should be judged as to the quality of the implementation for the admin. The author might remeber that there are other platforms than Linux, and other types of admin tasks to be accomplished as well, and expand on this to create an even better fleshed out set of guidelines. Overall, I would say an 8 out of 10; excellent ideas and a very good place to start.

Mary | Thu, 23 Dec 2010 21:35:15 UTC
Overall: AMEN!

A GUI can make it easier to do easy things, but also tends to make it harder to do hard things.

A CLI generally makes it easier to do hard things, though yes, sometimes that means it will be harder to do easier things *first time out of the gate.*

And a CLI is hands down the only way to do things repeatably at any kind of scale. Even if you want to automate a GUI deployment by (in Windows-land for example using AutoIt)...you still need to write the script to do it. Wouldn't it be nice if you could just have that to start with?

Craig | Thu, 23 Dec 2010 21:34:57 UTC
On the point of configurability, Cellar, we are in violent agreement.

Cellar | Thu, 23 Dec 2010 21:23:30 UTC
Craig, true enough. In linux basically everything ends up under /usr (that in turn was there for "user" applications, ie non-system ones. that got blurred a bit.) where certain other systems put all the packages under /usr/local (freebsd) or /usr/pkg (netbsd, which leaves /usr/local for really local things). Sorting all that out takes a package manager, some of which allow for "retargetable" packages, so you could at least in theory say "install this package under /opt". And you can do the same with FOSS especially if it comes with a configure script (--prefix=/opt). That basically leaves commercial binary-only software. There's also the linkfarm approach of putting software in its own tree and dropping links to executables in the common bin, with some manager to sort that out. Nowadays you could even opt to run a VM and let the software scribble all over that if it doesn't want to let you put it where you want it. Hand out application admin duties on the VM level and a lot of access level hassle goes away. There's plenty you can do; do whatever works best for you. But that wasn't my critique. That was the plea for changing world-wide defaults from something well-known and entrenched to something, er, not quite so well-known. I'd put it differently and say that some like it in /opt, and some like it in yet a different place, so please do provide configurability. If software fails that low, low standard, I'd definitely ditch it.

j | Thu, 23 Dec 2010 21:21:41 UTC
One addition:

Do keep your software up to date with dependencies. There's nothing worse than an application that needs a years old version of java or internet explorer.

AngryCustomer | Thu, 23 Dec 2010 21:21:15 UTC
DO - accept that your software is not the single most important reason why our company bought 2000 PC's. behave accordingly. - accept the fact I do not want to blindly trust you or your judgment. I use your software because I have no other option, not because I like your software. - assume that it is not a named user that is installing. Assume the installation will be automated using an account that has no home directory. - provide a mechanism to alter ALL application settings during deployment and for me to supply the default user settings. - provide an update mechanism that allows proper update deployment, NOT involving each PC trying to download its own update from the internet. Preferably integrated with the OS's built-in system for this.

DONT - let updates make any configuration changes. I really am not interested in that desktop icon. - bundle stuff. If I need it I will install it myself, don't worry.

Seriously, there are several companies out there I would rather see go bankrupt today than tomorrow because they do stuff like this. If those companies where real persons: If I saw them dying on the street I wouldn't even call 911.

Cellar | Thu, 23 Dec 2010 21:11:06 UTC
"Dr Steve", I fear GUIs have robbed you of your wits and your ability to spell. You're mixing up a couple of things. First, you seem to have forgotten why you have sysadmins. A good sysadmin prefers to not do any mundane taks, but to automate it away. You pay them 200 kilodollar a year not to do mundane tasks, but to automate mundane tasks so nobody has to do them. You can create GUIs to drive your collection of automated tasks, and sure, if you feel like it, you can do so. Second, what Tom was getting at is that you need something to automate /with/. Trying to automate GUIs (yes, I've tried it) is a losing proposition. Trying to automate CLIs empathically is not. Therefore, software intended for running under admin care such as most server software --repeat after me: real servers are headless-- and anything that might end up rolled-out over more than a few machines (like the 2000 mentioned above) must provide the hooks for automation. Which starts with a CLI. SNMP was specifically invented to provide programmatical hooks to data gathering applications, and indeed some of which provide GUIs and graphing and whatnot. But software vendors need to start with providing those hooks so others can make their preferred interfaces, GUI or not. And third, copying and pasting shell commands --perhaps through the TFI: Ten Finger Interface, you know, copy by eyeball and fingers-- might be necessary on occasion, like when you get to find out if your underpaid sysadmins are worth the money, as in whether they can restore order from some catastrophe or other. Your site documentation (you have that, don't you?) especially the part dealing with disaster recovery and bare metal backup restoring (you dry-ran that lately, didn't you?) might well contain commands to copy-and-paste. A bit of copyable text is far easier to keep up-to-date and far less prone to errors than screenshot upon screenshot of some GUI application that might be ten versions out of date by the time you need it. To sum up: Nobody said you couldn't have a GUI. The message was for vendors to provide for the hooks to build automated tools with. That, you do with things that automate easily, and that happens to be the CLI.

aspro | Thu, 23 Dec 2010 21:10:11 UTC
@NikT As soon as the users stop whining, so will the admins.

Craig | Thu, 23 Dec 2010 21:09:32 UTC
Well Cellar, /etc, /bin etc. traditionally were for system files. And before Sun started to push things toward /opt, there was /usr/local which provided a nice place to put users local programs and configs.

I always try to contain an entire installation in a directory such as /usr/local/foo so that when I get it installed, configured and tuned on one server, replicating it out to additional servers is usually as easy as copying a tarball and untar'ing it. Cleanup is usually very easy also.

I hate packages that spread their detritus all over the filesystem.

Cellar | Thu, 23 Dec 2010 20:46:51 UTC
I'd like to take the opportunity to chime in on Twirrim's XML gripe. Yes, it's all the rage. No, making sure your output is well-formed is apparently entirely optional. Not your problem if the parser chokes, we'll just patch it up somehow(!). I've seen client-server software projects invent custom DTD-free XML-in-XML (with some SOAP inbetween) only to get stuck dead in the water while changing the quoting method (indicated in the "documentation" through unexplained ink colour differences), run itself aground on encoding ambiguities(!), and crash and burn on failing to produce a proof-of-concept client for the server. And, of course, full steam ahead on the denial that any problems could possibly exist. It's XML, right? Right?

Since nobody can be arsed to do it properly, I might as well, and so I hereby do, declare XML to be honorary binary.

Dr Steve | Thu, 23 Dec 2010 20:46:07 UTC
I am not sure I agree with the No GUI. GUI's can be created that generate logs, manage authentication and create repeatable sets of task. In one instance we created a two phase authenticate and a proofing system for large roll outs. (Jr+Sr. had to authenticate.) I think this idea is 20 years

instance we created a two phase authenticate and a proofing system for large roll outs. (Jr + Sr. had to authenticate.) I think this idea is 20 years old and based on old ideas, many bad GUI have 25 check boxes. I have created and used others with preset tools which a Sr. admin and sec. professional can create and review the process while the Jr.'s responsibility is to run the task. GUI are useful especially allowing Jr. people to perform complex task in an effective, repeatable and auditable manner, if you want to spend plenty of money on roll outs with you best people doing mundane task then yes do that copy and paste 1980 thing or ship it over seas for less money per hour. Sr. people love companies and departments that waste their time on mundane task while at the same time want them to "work smarter not harder." I love paying someone 160 a year to copy and paste shell commands. I get better results with my Sr. people developing the process and QA the process while Jr. run and manage the deployment.

If like the rest of computing you want to do some task in a far more productive way while at the same time simplify the process think GUI. This way you can have a lower level admin deal with the deployment of the task and Sr. admin deal with the development of the task. GUI are not for everything but do not sell them short or take them out of your tool kit. Off the self GUI are not the answer, but it you have really good admins using Python or Perl, they can also create a GUI that can do amazing things and make everyone happy. People hated IDE and now they are used to create all kinds of Artifacts. I think they rather do that and put that on the resume then staying all night to do a upgrade.

This is from someone that loves CLI and fights every day not to live in the CLI past. Their is a reason GUI are used to do most of the useful computing task, do not fight, use it! GUI can become a tool, not a problem.

Cellar | Thu, 23 Dec 2010 20:31:16 UTC
Philip, you're welcome to your wish for a GUI. But do note that you can build a GUI on top of a CLI far easier than the other way around. A CLI simply isn't optional, it must be present. A GUI is nice for some people, but for people obsessed with automating stuff, as any good sysadmin is, it simply is but yet another obstruction to getting things done. Warm fuzzy feelings nothwithstanding.

And, er, you're the sysadmin. You don't get to handwave and not know things like your users. You get to learn and explain it to them, or hide the complexity in some script tucked well out of their way. You get to pull things out of the pot. Yes indeed, you do get to read the manuals and make the software do your bidding. A good reference manual is far more important to getting things done than fulfilling your wish for a GUI out of mental lazyness. If you want one, build it yourself out of a quick-fab GUI language that can poke the right CLI commands into the app. After all, you're the sysadmin.

NikT | Thu, 23 Dec 2010 20:30:56 UTC
Oh gawd, somebody stop the whining sysadmins!!!

VKVCL | Thu, 23 Dec 2010 20:30:40 UTC
@DNF, I am involved in specifying a SaaS platform right now, and I want to make sure that it is easy for IT to manage. What are the main frustrations you're encountering, and how would you fix them?

Al | Thu, 23 Dec 2010 20:26:03 UTC
@DNF one of the reasons I think Java stuff doesn't use the built in auth is because it is designed to be portable. If you make it dependent on Unix authentication mechanisms, it won't work on Windows and vice versa. Even between Unix systems, the auth can differ quite a bit (PAM against an external source, passwd file, etc.), so it's a difficult problem to tackle.

Cellar | Thu, 23 Dec 2010 20:23:26 UTC
I'll disagree with W^L+ here:

/opt is one of those things that's highly useful for some things. But it's not a good default for all things. And worse yet, it's not the default. /etc and /var and whatnot are, and that's as it should be. Don't go advocating to the world, or even begging the world, to change their defaults of thirty years to what's good for your site. There are other ways to accomplish delegation, like ACLs, MAC policies, or even the venerable unix group. It works pretty well, in fact. Use the chmod, luke.

That said, I expect software to be configurable --at compile time in the case of FOSS, a command line option in all cases, an environment variable if you somehow must, that sort of thing-- as to where it finds its configuration and (perhaps through that configuration) the other things it needs. That should simplify putting things in /opt if you so choose to do so. Wrap it up with a small shell script, done.

Philip | Thu, 23 Dec 2010 20:19:45 UTC
I agree except for #2. As a sysadmin, GUIs make my life much, much simpler. I realize that a CLI is very important and I agree that one should always exist, but so should a GUI. If your application does not have a GUI, I am going to complain about it all day long. I have much more important things to do than read manuals about which CLI switches to use. Please always include both for those of us that like CLIs and those of us that like GUIs. :)

W^L+ | Thu, 23 Dec 2010 19:55:08 UTC
A lot of us that have to admin the network or system are not full-time administrators. We often do not have full admin rights for everything. We have other duties as well. Please ... for our sake, instead of putting configs in /etc and binaries in /bin or /usr/bin, put everything under /opt/appname. It makes it easier for the full-timers to give us the appropriate rights.

On Windows, allow remote administration and even when another user is logged into the computer. it is a big pain to have to log user out to install / update / remove an application.

But otherwise, I agree with most of the above.

Hellvis | Thu, 23 Dec 2010 19:17:50 UTC
And please stop building programs that break the windows security model. It's pointless to have users in groups with less privs than local machine administrator if the program won't run unless the user is a local machine administrator.

Twirrim | Thu, 23 Dec 2010 18:31:29 UTC
DNF++. So tired of setting up yet another authentication system.

On the API front, my personal favourite plea: Please keep your API documentation up to date. I'm so tired of finding undocumented changes to APIs that mean the documented section is almost entirely pointless. I think R1Soft's docs were the first time I was ever able to use the API exactly as documented. It was the first time I only had to troubleshoot procedural mistakes in my code rather than API quirks.

Minor related plea: If you provide XML output, please provide a clear, logical and as simple as possible schema. On numerous occasions I've had to alter XML parsers to handle a schema that wildly changes based on trivial differences. e.g bulk content appearing under different levels of the document or worse.

DNF | Thu, 23 Dec 2010 17:07:28 UTC
I would add one big other big one: "don't have your own user database and/or authentication system." The OS has all of this built in, use it. In fact, if it's reasonable to use the OS's mechanisms for authorization, even better.

There's nothing more annoying than having to recreate accounts/passwords for dozens (or hundreds, or thousands) of people just to access your application. Plus, it's much better security to disable users in one place instead of each of a dozen different applications.

Admittedly, this is getting somewhat better these days, but we still have a long way to go and I encounter plenty of applications that don't do this. (Java-based apps seem to be the worst for this kind of thing and just about every other item on your list. What is it about the java language/environment which causes developers to think theirs is the only application in the enterprise and we all have hours a day to click on buttons to configure/install it?)

Unfortunately, SaaS providers are pushing things backwards on this front, too.