21
Dec
2009

Given 250,000 tools on the shelf, how do you manage them?

Although I haven't seen a thoroughly researched study, I figure there must be at least 250,000 FOSS (Free and Open Source Software) tools available to every systems administrator on the planet (230,000 at SourceForge + 15,000 at Launchpad + 12,000 at CodePlex + 5,000 at Google Code and that doesn't count the Linux kernel or any of the myriad other self-hosted projects). These 250,000+ resources comprise the full "toolbox" that admins can use for building solutions with FOSS; they represent the FOSS equivalent of COTS (Commercial Off-The-Shelf). Of course, if you add open source but non-free or commercial tools, the problem explodes combinatorially.

How can a systems administrator support the largest possible subset of these "on the shelf" resources to best service the next need from a stakeholder (like the boss or a new client)?

First let me emphasize the difficulty of the task with a list of items that systems administrators and systems management firms like LinuxForce are expected to do whenever a stakeholder presents a software need:

- Find and Evaluate software that can meet the need:
    - Identify several candidate applications that might meet the business requirements for a given project, function, or need
    - Research the options to assess their ability to meet the requirements (actually we, the systems administrators of the world, are actually expected to know which tool is "best of breed": just from our past experience. The false assumption is, if it isn't well known it must not be any good. The long tail applies to the 250,000+ FOSS tools also!). In our experience such research is essential, unfortunately, there is rarely enough budget to carefully explore the options.
    - Install the tool(s) in a "sandbox" to allow the stakeholder to "try it out"
    - Select a tool to use or look for more options
- Put the tool into production
    - Read the docs to identify best practices for the software's configuration
    - Prepare an installation plan that will address (as best as possible) any upgrade glitches (yes, you have to anticipate them now or suffer the consequences later!) so that you're prepared for when a security advisory is released (or when the stakeholder starts begging for features from a new release)
    - Figure out a support plan to handle the inevitable questions that will arise during operations
    - Integrate these considerations into the process of either installing a package or using the "make, configure, make install" steps that most FOSS tools provide for installation
    - Carefully document the "as built" configuration including all assumptions and anticipated glitches to help yourself or future admins during the maintenance phase
- On-Going Maintenance
    - Monitor the software

- Subscribe to any relevant security mailing lists for the software so that you are apprised when a security (or other major) problem is detected
- Track general trends relating to the software and its alternatives so that you are ready to respond if the project goes dormant or is eclipsed by newer, superior technology.
- Upgrade routinely

About 15 years ago I noticed that the explosion of ready to use FOSS tools plus the trend toward general purpose tools and away from custom software was leading to a combinatorial crisis in software maintenance. I saw that it was the systems administrator's responsibility to address the situation.

It has become apparent to me that the solution would require use of *convention*, *standards* and *policy* to reduce the complexity of the problem to manageable proportions. I searched for the most "standardized" conventions and policy-enforcing environment that would also provide the most flexible access to the most FOSS tools. The solution I found is **Debian/GNU Linux**, *the universal operating system* (although Ubuntu and other Debian derivatives also provide most of these benefits as well).

Debian simplifies the software evaluation process (apt-get [search|show]). Debian simplifies installation (apt-get install), security and new version upgrades (apt-get [upgrade|dist-upgrade]). Debian uses conventions and packages to simplify identifying best practices for administering the software (/usr/share/doc/[package]/, /var/lib/dpkg/info/[package].postinst, and wikis, mailings lists, bug reports, etc.). But the key benefit for managing the combinatorial explosion of FOSS tools is the Debian community's value of striving to configure each package to automatically support the most common use cases while also providing support for unusual configurations (so you save tons of time in configuring the software).

In summary, the Debian/GNU Linux system provides the infrastructure needed to manage the combinatorial explosion of *off the shelf* FOSS tools cost effectively. If you have to service a lot of users, customers, or clients with challenging, diverse needs, I think Debian is the most cost effective way to meet their needs and deliver quality maintenance on an on-going basis year after year after year.

Posted by CJ Fearnley *in* Debian, Systems Management, *1 comment*