

10 tips for sensible systems administration

Maintain your UNIX systems

Skill Level: Intermediate

[Martin Streicher](mailto:martin.streicher@gmail.com) (martin.streicher@gmail.com)

Software Developer
Pixel, Byte, and Comma

10 Mar 2009

Benjamin Franklin: scientist, scholar, statesman, and . . . systems administrator? Yes, 200 years or so before the birth of UNIX®, Franklin scribed sage advice to keep systems humming. Here are 10 of Franklin's more notable tips.

Ask anyone about Benjamin Franklin, and you'll likely hear about his experiments with electricity, his participation in the founding of the United States, and his invention of bifocals (see [Figure 1](#)). Less well known but nonetheless impressive, Franklin also advocated the adoption of paper currency, printed money with innovative anti-counterfeiting techniques, established the United States Postal Service, and formed the colonies' first fire department—the Union Fire Company—in Philadelphia in 1736. In fact, centuries before Smokey the Bear, Franklin proffered sound fire safety advice, famously stating, "An ounce of prevention is worth a pound of cure." Wise words and pertinent to this very day—especially if you're a UNIX systems administrator.

Figure 1. Benjamin Franklin: statesman, inventor, and would-be UNIX systems administrator (notice the long hair), as captured in 1777 by painter Jean-Baptiste Greuze



Building on Franklin's wisdom (he also once quipped, "Guests, like fish, begin to smell after three days"), here are 10 tips to keep your UNIX systems humming.

Franklin on security

"Distrust and caution are the parents of security."

Keeping a system secure is difficult. There are no magic bullets and no product you can purchase and install to definitively protect against all threats. Instead, defending a system requires constant vigilance to create, find, apply, test, and adjust any number of individual measures. (A little) paranoia is also healthy. Acknowledging the ephemeral nature of security, Franklin cheekily said, "Three can keep a secret, if two of them are dead."

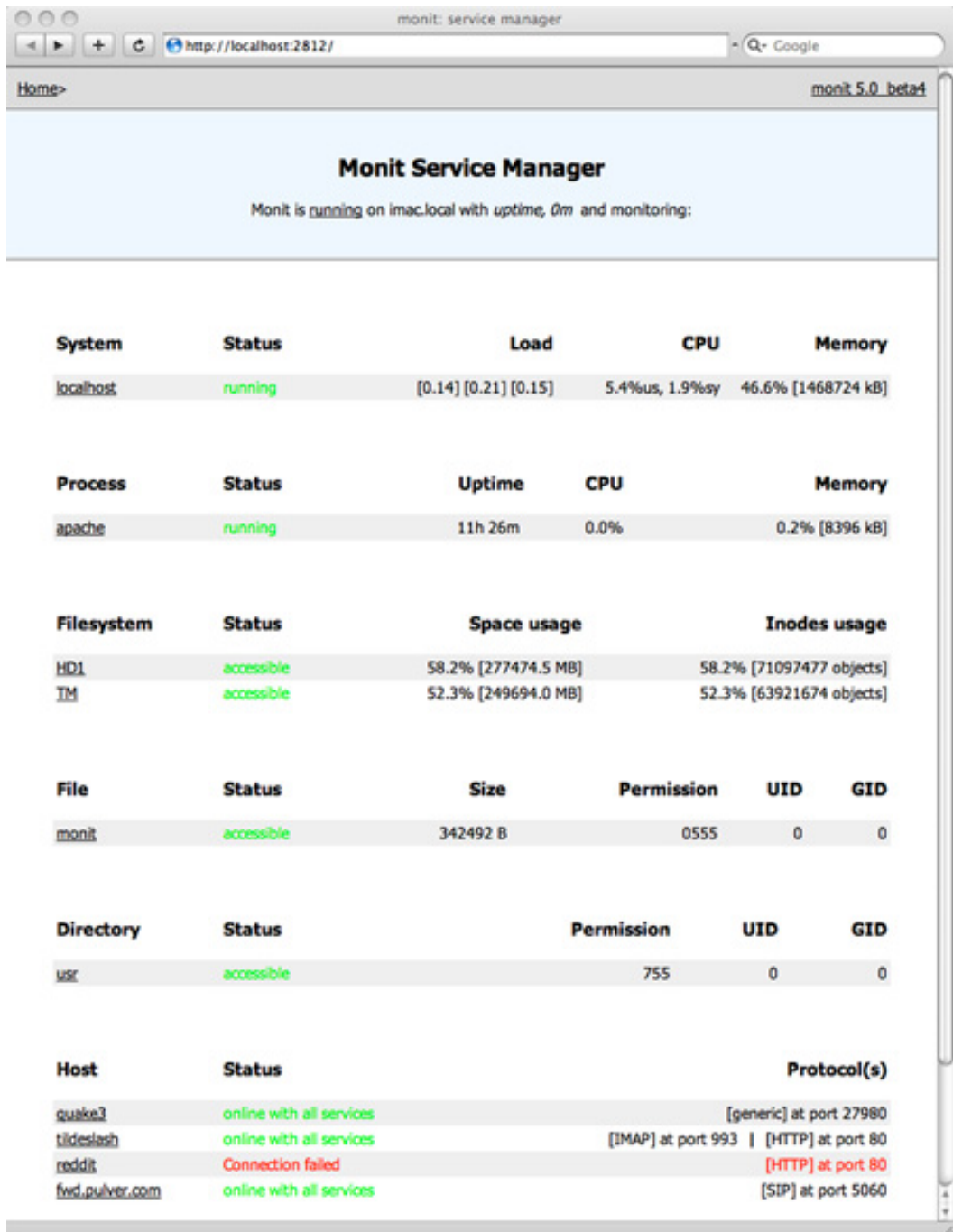
Here are some actions you can take to better protect your system:

- Subscribe to and read security bulletins to learn of threats to your operating system and application software, and determine whether each threat endangers your specific configuration. (For example, recent issues with Apache HTTP Server were restricted to its `mod_proxy_ftp` module, which many sites disable.) When action is needed, form a plan to counter the peril, such as applying a patch to the source and recompiling or updating the software from your vendors' package repository.
- Grant minimal access to each user and application. Provide only those

rights required to accomplish a task and no more, and never permit direct access to the `root` login. If an application must run as root, consider an alternate solution.

- Never compromise security for expediency.
- Deploy Monit to detect intrusions. Monit, among other similar, capable tools, watches the mode, time stamps, hash value, and other attributes of files and directories and alerts you when something (perhaps errant or diabolical) makes a change. (Monit also monitors processes, such as Apache, and can restart daemons in the event of failure. [Figure 2](#) shows Monit in action.

Figure 2. Monit monitors files, directories, processes, network ports, and more



Franklin on consistency

"It is easier to prevent bad habits than to break them."

Nothing is as maddening to a systems administrator as the "one-off"—the lone, unique, just-ever-so-slightly-different machine configuration or "little" permutation

required to effectuate some initiative. Although well intended, one-offs are easily forgotten, potentially harmful, something of a Pandora's Box, and best avoided.

That said, it is also unrealistic to preclude all exceptions. As Forrest Gump, another learned man, said, "S*** happens!" In general, consistency is the best policy. However, when the stuff hits the fan, take care to record divergence.

Here are some suggestions:

- Never make an unproven change to a running production system. Stage each change in a dedicated environment, then repeat it in production.
- Maintain your own repository of software, and use it as a canonical source to populate your systems. Yum (see [Resources](#)) is an excellent choice for local software package management and distribution, if your system uses the Red Hat Package Manager (RPM) package format.
- If you repeat a manual task more than a small handful of times, capture the operation in a script. A shell script encapsulates your intent, confounds errors, and saves time. By the way, if you write a script to implement a change, write its complement—the "undo" script—at the same time. Test both to ensure robustness.
- Use a source control system to maintain system configuration files. Source control allows you to "roll back" to a previous iteration of a file to restore the status quo or to recover from an error. Further, use a database or a system such as Trac or Lighthouse (see [Resources](#) for links) to follow the life cycle of a configuration change—from request to implementation to expiration.
- Set an expiry for each one-off, and undo the change on schedule (with few or any exceptions).

Franklin on preparedness

"By failing to prepare, you are preparing to fail."

Certainly, crises occur. A UNIX system tends to run and run, and left to its own devices (double entendre intended), uptime is excellent. However, Web traffic can spike, hardware can fail, applications can break, and people *do* make mistakes. Murphy's Law is the first rule of information technology (IT).

Hopefully, though, crises become increasingly infrequent, allowing you to spend the bulk of your time, perhaps ironically, preparing for and avoiding crises. Daily chores must include backups, log rotation, performance measurement (more on this in a moment), patches, and safeguarding all systems.

Here are some less obvious tasks to add to your daily maintenance routine:

- Validate your backups. A crisis is no time to discover that your backups are incomplete or unusable. Practice restoration drills often. Moreover, define a plan to recover and continue your business if all systems were lost to force majeure.
- Pay attention to upcoming renewals—especially to Secure Sockets Layer (SSL) certificates and any service or product you depend on where a lapse can interfere with or even interrupt access or commerce. Allow at least a month to renew a regular SSL certificate and a longer lead time for the new Extended Validation (EV) certificates. An impending renewal is also the perfect opportunity to initiate a search or negotiate for better service terms.
- Expect your upgrades to *fail*, and have a recovery plan. Perhaps Mr. Murphy will take the day off and your Oracle upgrade will proceed swimmingly . . . but don't count on it. Business continuity is essential.

Franklin on frugality

"Beware small expenses. A small leak will sink a great ship."

However cliché, it's true that IT budgets continue to shrink while demands for breadth and depth of service continue to multiply. Every dollar counts. Franklin's sentiment about fiscal responsibility can also be applied to effort.

Here are a few pointers:

- Franklin didn't coin the phrase, "Keep it simple, stupid," but he would have appreciated its irreverence and pragmatism. Be mindful to make one change at a time, and avoid the temptation to slip in "one more thing." For instance, if you're upgrading the kernel, don't change Domain Name System (DNS) entries at the same time.
- Disable—even delete—unused services and software. It's probably unnecessary for a public-facing Web server to run, say, Simple Mail Transfer Protocol (SMTP) for mail delivery or the Common UNIX Printing System (CUPS) for printer management. Every component is a possible source of work and trouble and a drain of processor cycles, disk space, and time; prune anything you can.
- Optimize, optimize, optimize, then optimize some more. To paraphrase Bruce Schneier, "Optimization is a process, not a product," and like security, there are no silver bullets for poor performance. Installing more memory or an additional processor can certainly help—at least until those

novel resources are consumed, too—but more cycles and more RAM may merely mask endemic inefficiencies.

Franklin on information

"Who you gonna believe, me or your own eyes?"

Actually, this quote does not belong to Franklin but to Marx. Chico Marx, that is (playing Groucho Marx in the classic *Duck Soup*).

Of course, one prefers to rely on empirical data, not hearsay or speculation, to make good decisions. Thus, it's vital that you collect system metrics to guide planning and evaluate operations. Continually monitor usage and capacity to establish what's "normal." Aberrations, when they then occur, can be readily identified. Baselines are also invaluable for planning, as you can correlate an anticipated or observed up-tick in usage to an expansion of resources.

Here's a bit of guidance:

- There is a vast selection of automated system-monitoring tools, and most are open source. Look into ZABBIX, Nagios, and Cacti, for instance (see [Resources](#) for links). Tie such a system to e-mail and pagers to alert staff to an emergent problem before crisis strikes.
- If your system is capable, enable hardware monitoring, such as IBM® SMART HDD alerts. Many hardware vendors also provide software agents to monitor hardware; install and run those daemons. Again, broadcast alerts to staff to proactively intervene.
- Review the utilization patterns for the previous 24 hours, the previous week, and the previous month. Pay particular attention to load average, disk space consumption, and memory usage; then, formulate trends. Add hardware before its needed.
- Watch syslog for important messages, such as failed login attempts and daemons with errors.

Franklin on education

"If a man empties his purse into his head, no one can take it from him."

Few, if any, industries change as quickly as the computer industry, and IT changes even faster still. The best investment any systems administrator can make is in education.

Take training courses and obtain certifications for career growth and for tackling large projects. Read magazines to be informed and to frame opinions and trends with context. Subscribe to feeds from peer systems administrators for practical advice.

And most important, "RTFM!" That is, read the, ahem, fine manuals, man pages, release notes, README, and other documentation that accompanies each piece of software you operate.

Franklin on open source

"As we enjoy great advantages from the inventions of others, we should be glad of an opportunity to serve others by any invention of ours; and this we should do freely and generously."

Franklin was a prolific inventor, yet eschewed patents. This particular quote, in fact, was a response to the Governor of Pennsylvania's offer to patent Franklin's novel stove, thereby granting Franklin a monopoly on its manufacture for several years. (Until 1790, each state had its own patent legislation and issued patents independently. The Federal Patent Act passed in 1790 made patents the purview of the Federal Government, as mandated in Article 1, Section 8, Clause 8 of the United States Constitution.)

No doubt, Franklin would approve of open source and would likely side with critics of both software patents and copyright legislation run amuck.

Franklin on cooperation

"We must, indeed, all hang together or, most assuredly, we shall all hang separately."

IT surely has its bailiwick, as this article bears out. However, IT does not exist in a vacuum. Its goals, priorities, expenditures, and services—indeed, its daily chore list—must align with the intentions of the business.

No matter how well tuned a UNIX box is, it's optimized only for a set condition; if the condition changes or is replaced, the UNIX administrator has to adjust accordingly.

Franklin on vendors

"The art of concluding from experience and observation consists in evaluating probabilities, in estimating if they are high or numerous enough to constitute proof. This type of calculation is more

complicated and more difficult than one might think. It demands a great sagacity generally above the power of common people. The success of charlatans, sorcerers, and alchemists—and all those who abuse public credulity—is founded on errors in this type of calculation."

Translation from the Old English: Run your own benchmarks.

Like alchemy, finding the right mix of hardware and software ingredients to optimize your specific application is a black art—and one practiced most effectively in your own lair. Identify your requirements and your crucial benchmarks before you approach any vendor. Solicit proposals from multiple vendors, and ask your top two or three choices to loan you prospective equipment before you buy. Run your benchmarks, perhaps even inviting the vendor to participate to fine-tune for your specific needs.

Franklin on inevitability

"[I]n this world nothing can be said to be certain, except death and taxes."

Here, Franklin is only partially correct. It's probably more accurate to say, "[I]n this world nothing can be said to be certain, except death, taxes, and crashes."

Even the most mindful, meticulous, and maniacal systems administrator cannot prevent the inevitable: Hardware fails, and typically at the most malapropos time. Yes, it's all about preparation, preparation, preparation—and spares. When purchasing a piece of hardware, consider its mean time between failures (MTBF), and stockpile reserves accordingly. Here are some rules of thumb:

- Purchase at least one spare component for every 10 to 15 components being placed into service. Typically, it's most economical to purchase spares and production equipment at the same time.
- Notwithstanding the previous guidelines, purchase at least one spare hard disk for each model hard disk you deploy. For example, if you buy three Super Spindle 100s and two Disc-o-Tech LXs for production use, buy an extra Super Spindle 100 and an extra Disc-o-Tech LX to have on hand. If either spare is forced into service, replace it immediately.
- If you support aging hardware (and who doesn't?), consider purchasing two or three replacements if a part should conk out. The older hardware becomes, the more difficult it is to find parts. Corner the market when you can.
- Adopt a rich configuration-management and deployment system such as

Puppet (see [Resources](#) for a link), so that a new host can rapidly be brought up to replace a failed one with minimal operator involvement.

There is no single, right strategy for reserves, and your approach must be tailored to your architecture. If your servers are pooled in a compute cluster, the head node is the only crucial machine; the rest can start and stop and crash freely. If your systems are interconnected to failover to hot spares automatically, your stockpile can be slight.

A lone, dedicated machine providing some crucial service—say, DNS or centralized authentication—is an Achilles heel and warrants a bullpen for backup. Additionally or optionally, plan ahead to migrate critical services to another system in the case of emergency.

From words to wads

Benjamin Franklin once heckled, "A countryman between two lawyers is like a fish between two cats." Founding Father and stand-up comedian.

In addition to Benjamin Franklin and Chico Marx, I would like to thank Ted Boggs, Mark Bouman, Oleg Brodtkin, Dana French, Tom Georgoulas, Steve Holdoway, Altan Khendup, Jeff Kimble, Lubos Kolouch, John Mascio, Philip Mather, Nathan McCourtney, Ray Robert, Matthew Sacks, and Jesse Sipprell for contributing their own wise words to this article.

"No matter how well tuned a UNIX box is, it is optimized only for a set condition; if the condition changes or is replaced, the UNIX administrator has to adjust accordingly" is a direct quote from Altan Khendup.

"Never compromise security for expediency" is a direct quote from Dana French—also an IBM developerWorks® author.

Hopefully, you've found Franklin's and the others' advice helpful. Who knows? You may even find that Franklin's words put Franklins in your pocket.

Resources

Learn

- The United States National Security Agency (NSA) provides recommendations to secure several of the major operating systems, including Sun Solaris, Microsoft® Windows®, Mac OS X, and Red Hat Enterprise Linux®. You can find the guide online at the [NSA's downloads site](#).
- *Extended validation* refers to new, enhanced processes of validating Web sites before issuing a certificate. You can find more about these high-security certificates at [Verisign's site](#).
- The [AIX and UNIX developerWorks zone](#) provides a wealth of information relating to all aspects of IBM AIX® systems administration and expanding your UNIX skills.
- [New to AIX and UNIX?](#) Visit the New to AIX and UNIX page to learn more.
- Browse the [technology bookstore](#) for books on this and other technical topics.

Get products and technologies

- According to its Web page, "Trac is an enhanced wiki and issue tracking system for software development projects," but it works equally well for administration requests. Trac is available at no cost and can be downloaded from the [Trac download site](#). As another option, [Lighthouse](#) is an issue-tracking system provided as software-as-a-service for a nominal fee. Both solutions integrate well with e-mail.
- [Monit](#) is a free and open source system monitor. You can download it from its home page.
- [Yum](#) manages RPM packages and distributes them to remote systems. You can use Yum to keep your systems in sync.
- Puppet is a system for automating systems administration tasks. Learn more and download the code from the [Reductive Labs site](#).
- [ZABBIX](#) is an enterprise-class, open source, distributed monitoring solution. It has many features, including escalations and repeated notifications and support of Internet Protocol version 6 (IPv6).
- Nagios is another popular monitoring solution, with more than 10 years of ongoing enhancements. Nagios can scale to more than 100,000 nodes. You can download the source from the [Nagios site](#).
- To visualize system and network performance, try [Cacti](#).

Discuss

- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).
- Participate in the AIX and UNIX forums:
 - [AIX Forum](#)
 - [AIX Forum for developers](#)
 - [Cluster Systems Management](#)
 - [IBM Support Assistant Forum](#)
 - [Performance Tools Forum](#)
 - [Virtualization Forum](#)
 - More [AIX and UNIX Forums](#)

About the author

Martin Streicher

Martin Streicher is a freelance Ruby on Rails developer and the former Editor-in-Chief of [Linux Magazine](#). Martin holds a Masters of Science degree in computer science from Purdue University and has programmed UNIX-like systems since 1986. He collects art and toys. You can reach Martin at martin.streicher@gmail.com.

Trademarks

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

IBM, AIX, and developerWorks are registered trademarks of International Business Machines Corp in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are registered trademarks of Microsoft Corp in the United States, other countries, or both.