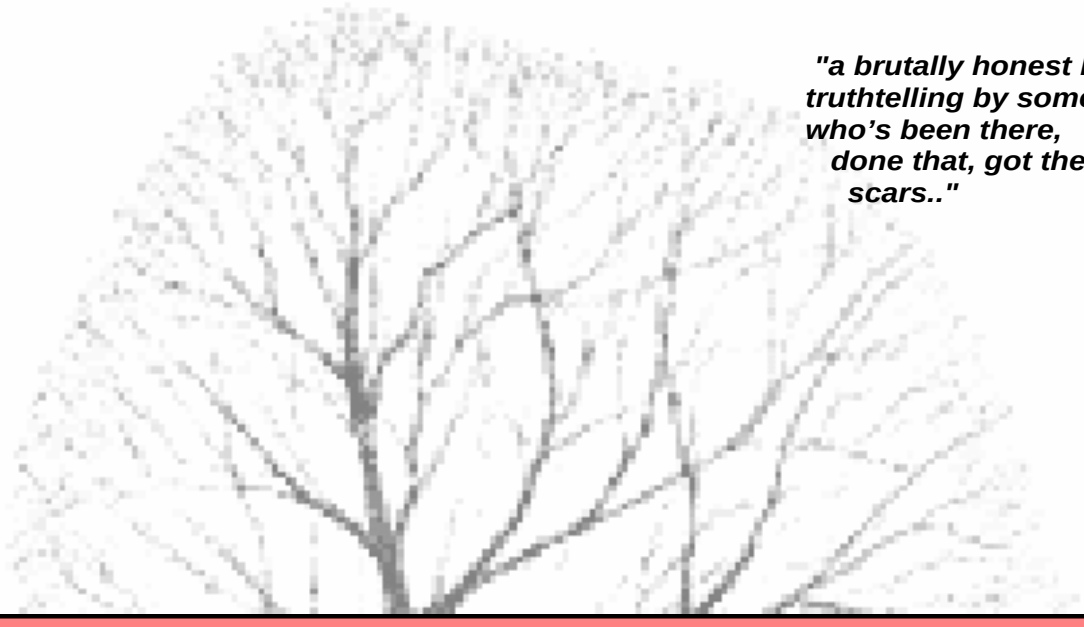
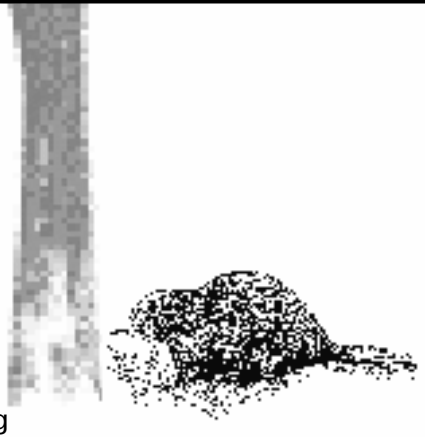


"a brutally honest bit of truthtelling by someone who's been there, done that, got the scars.."

The Unix Guide To Defenestration

Murph P. Murphy



"Murphy treats sacred cows as so much steak on the hoof as he explains why so many systems projects fail and what to do about it. "

Duane Gran
SolarisCentral.org

The Unix Guide to Defenestration

Paul Murphy

murph@winface.com

The Unix Guide to Defenestration

By Paul Murphy

Third Edition: Oct. 2005 . Printed in Ashland, Ohio.

ISBN 0-9689004-0-3

Notice

By necessity, given the nature of the thing as a book of opinion and criticism, this book names names, discusses prices, and gives my opinions on the performance and value of various products. Obviously:

1. all brand names, product names, web site names, and related materials as used in this book are, or may be taken as, trade names, service marks, trademarks, or registered trademarks of their respective owners; and,
2. while every precaution has been taken in the preparation of this book neither the publisher nor the author assumes any legal responsibility for errors or omissions, or for damages arising from the use or distribution of the ideas and/or information, contained herein.

Table Of Contents

Chapter 1
Introduction

1.1 The Series

There are three books in this series:

BIT (*Business Information Technology*) is aimed at helping non systems people understand what drives systems decision making, what technologies are important, and what to look for when asked to contribute to a systems project or decision.

This one, *The Unix Guide to Defenestration*, is aimed at people who work in systems, are frustrated by their inability to meet user and executive expectations, and are willing to work toward changing the system.

Number three: *Yea or Nay: the Board Member's Brief on Computing*, provides a high level, no holds barred, briefing on Information Systems for board, or senior management committee, members.

Brief includes a checklist for failure -the 'tells' in IT proposals that give away a doomed project- suggesting what questions to ask, and what guarantees to demand, before making the yea or nay decision.

1.2 What this book is about

The primary issue in systems management is not cost or technology but service orientation --are your IT staff customer focused? are they responsive to user needs? is the infrastructure provided empowering the business or restraining it? That's what this book is about: *doing the systems job right.*

Doing the job right is much easier than doing it wrong - but only once your organization has made the transition from traditional systems thinking to service oriented meritocracy.

That's very hard to do: typically the senior people within IT will have one set of ideas, your users will have an almost opposite set, and most of your fellow managers or technical colleagues will put their careers ahead of service to the employer. Thus getting there, *the process of data center defenestration*, is also what this book is about.

With today's technology, you do the job most effectively by adopting and using Unix[1] and the very flat organizational structures it requires --exactly the things you find in the research and embedded systems communities where, contrary to business experience, systems usually work.

That's why this book about doing the systems job right, is equally about *doing Unix right.*

1. For this book "Unix" includes Linux, BSD, Solaris, HP-UX, and Tru64, as well as a number of other variants, like AIX, Unixware and Mach, that have smaller markets but share a common heritage and overall technical structure.

1.3 What this book is not

This is not a technical best practices book, nor is it a how-to manual; this book is about understanding the technical background, costs, organizational structures, and behavioral imperatives that lead to institutionalized systems failure and acting on that knowledge by adopting and using Unix as the first step in a positive transformation process --a process I think of as *data center defenestration*.

1.4 Book Structure

Chapter Two provides the background information on what makes Unix different, and how those differences affect management decisions.

Chapter Three is aimed at current or future senior managers in Information Systems. It focusses on issues in Unix data center management, including appropriate organizational structures, operational concerns, and key remediation concepts.

Chapter Four covers much of the same material, but from the very different viewpoint of the Unix sysadmin hired to make things work.

Chapter Five looks at application and business issues to illustrate some ideas about what can be achieved, what the obstacles are, and how they can be overcome.

1.5 Caution!

The dog that doesn't bark, is neither heard nor fed.

You have to get the house in order before you can morph your systems group from cost sink to profit center. That means adopting and using Unix as the cheapest and most effective way of delivering high quality services; but, *be aware that the resulting professional success could kill your career.*

As you start to succeed with a Unix infrastructure your users will see less of you --there is, after all, no need to bother you when things work. Think about it: *the perfect data center is perfectly invisible.* Get good at providing high quality services and your group will fade quietly into the corporate background.

That sounds stupid; do a better job and your bosses will appreciate you more, right? Wrong, your bosses read *In Flight* Magazines and *know* with

utter and complete conviction, that there's really nothing to Systems; any kid with a $200 PC and the right attitude can handle it. Daily user complaints about your failure to succeed at it serve them as a tenuous bridge to reality; but, get good at it, and you will quickly confirm their belief that your entire group can be safely and easily replaced with casual labour.

This is organizational reality. If you have lots of service problems then your users will complain bitterly to their bosses and so make it easier for you to get budget approvals for hardware, staff, or services expansion. Do the job right, and no one will know you while your budget will get penny pinched at every opportunity.

That counts for your future too: when you go after your next job, the guy who spent ten bucks to your one and failed to meet user needs while hotly embracing all the newest trends - will look a lot better to the recruiter than you will.

Put in a high reliability infrastructure that just works and pretty soon you'll be the invisible man. If you check with your boss's secretary, you will find that she knows the people who fix the photo-copiers, but not the person who maintains the phone system. Think about it: in Systems, failure succeeds, and success fails.

There is an answer. As you spend less time working with your users over service delivery problems, put serious effort into spending more time working with people in your organization on other issues. Don't shrink from success; don't, that is, hand your cost savings back as budget reductions: invent or develop new and organizationally valuable services to deliver and make sure people see you doing it. Use your savings to get in their faces, push the service envelope, do interesting things; spend some time *every day* making sure that senior management knows and values your contributions.

Chapter 2
Core Unix Differences

2.1 Concept Preview

This chapter looks at six things that both individually and taken together, distinguish Unix from all other operating environments currently offered.

These are:

1. Scaleability;
2. Reliability;
3. Use of smart displays;
4. Usability;
5. Openness in innovation; and,
6. Networking.

These distinguishing characteristics separate Unix from, for example, Wintel [Microsoft *Windows* on In*tel* style chips] solutions, but are not themselves the causes of anything; they are merely symptoms that arise from the real causes buried in the origin and history of the system's design.

As such they function as enablers mediating between what we want to achieve with the system and our ability to do so. It is, for example, the system's inherent scaleability that makes it possible to run 80,000 Oracle users against a single database instance, but that scaleability is itself the result of fifty years of theoretical research and advanced programming practice.

The existence of these enablers does not, by itself, guarantee success in using Unix to achieve personal or business goals; for that we need to understand and apply appropriate management strategies and resources.

support multi-tasking. As a result users could easily page through document images while making typed notes and then cut and paste directly into the application forms - but as of mid 2005 they still do this with XP by layering documents and bringing either the source or target to the front as needed; losing concentration each time.

2.3.4.4 Innovation

The system we eventually proposed used Sun's "Sun Ray" smart displays - although we did the demo with NCD Explora Pros. The feature management likes best about these has nothing to do with performance, ease of use, or cost. What they responded to best is Sun's use of a smart card to identify the user to the system. That offers them a much higher level of confidence in overall system security because the card, combined with the centrality of services and normal Unix logging, strengthens their ability to track employee activity.

2.3.4.5 Networkability

By having all system administration tasks done, in the second example, from two central workstations we took advantage of the networking facilities native to Unix to provide a single point of control, with consequent standardization and reliability.

2.3.5 Summary

In all three cases the real bottom line was that the proliferation of both staff and servers had left no one uniquely responsible for anything. At the physical level, server consolidation reduced capital cost while improving system performance considerably. The more important gains came, however, at the organizational level where staffing reductions and the elimination of conflicting agendas produced substantial long term cost savings while virtually eliminating reliability problems.

In Case three, for example, the simple recognition that Oracle 8i lets you attach images to row level records allowed us to eliminate 54 NT servers, 17 support staff, several hundred lines of client side duct tape and chewing gum, and two group managers. That's where the big gains came from: flattening the organization and eliminating the need for co-ordination left people with clear responsibilities and a conflict free service agenda.

From a structural perspective the UBA as implemented for the scenario company would consist of two data center teams separately responsible for the two halves of a mirrored system built around a pair of Sun Fire 6900s delivering both application and Mad Hatter office automation services to Sun Ray desktops and Mac laptops.

Data center staffing would consist of:

1. Four sysadmins in each group;
2. Two application specialists in each group;
3. One administrative support person in each group;
4. and, three people in the office of the CIO - two administrative, one management.

for a total of 17 people split into two operational groups both reporting to the CIO.

Notice that the five year UBA total, about $18,443,000, includes the costs associated with the mobile phones and the addition of IP integrated voice and video conferencing to every desktop on the network. It is, nevertheless, almost 30% less than the five year total of the Wintel costs shown without salary inflation and without telecommunications.

Cost savings like this are important as saleable tangible benefits. In reality, however, it is the other benefits that are more valuable. For example:

1. By design and implementation, this Unix architecture is functionally impervious to the security and reliability issues that can turn the Windows architecture solution into a daily struggle. The problem simply doesn't exist.
2. IT staffing is enormously simplified. With somewhere between a quarter and a third as many positions to fill, companies can pick better people, offer higher rewards, and build stronger, more stable, operational teams.

   Initially, of course, it will not usually be possible to hire 14 good Unix people during a reasonably short recruitment period, but the transition doesn't need that many. Companies starting with a good CIO and two experienced Unix team leaders can usually flesh out the rest of the staffing needed by hiring science graduates with little or no previous work experience while relying on Sun to do initial setup. Given the right atmosphere and leadership, such people become solid sysadmins and DBAs much more quickly than IT professionals who must first unlearn inappropriate habits of thought and action.

Organizationally you'll have only three hubs: the two operational centers and your headquarters staff group for a total complement of perhaps 120 or so people instead of the 900 needed for the Wintel model.

How not to do the sysadmin job

The impact of trying to apply management principles and ideas appropriate to a 1980s proprietary data center to Unix environments isn't limited, of course, to bad technical decisions. The more important impact is on how systems people behave.

I sat in on a meeting some time ago between some friends and a Unix systems administrator who nominally works for them, but actually reports only to Systems management. They were asking for a number of trivial set-up changes on a Solaris machine that, for organizational reasons outside their control, had been given to Systems to run.

This sysadmin came in, put his elbows on the table, and proceeded to give a display of arrogance and ignorance straight from the client manager playbook of a 1970s mainframe shop.

Had he worked for me, that meeting would have been his last for the organization, but neither he nor the managers he reports to are unusual - *and they're the reason people buy Wintel regardless of cost and performance issues.* Were this guy in control of my access to systems services, I'd be looking at alternatives like Windows too, just to get away from him. That's what the PC offered users --something which, however poorly it ran, was at least gave the finger to Systems and established some user independence.

Organizationally, what's going on here is centralization of control without a matching commitment to user service - creating a kind of resource based Systems dictatorship that should be unacceptable today but is actually being replicated in highly centralized Wintel deployments.

What made this guy's behavior acceptable in the 70s was the fact that the corporate processing resource was very expensive and so the sharing process that evolved set high barriers for access and gave out tiny bits of the resource. With modern Unix, however, the resource is cheap, security isn't an issue, and controls are expensive - so skimping on the controls and handing out big resource chunks simply makes more sense.

when senior people want to play, and minor changes will be taken care of without anyone else ever needing to hear about them.

Major business change, however, is another matter entirely. Many senior executives, especially in publicly traded companies, like to attach a patina of secrecy to “sensitive” issues like merger or acquisition talks.

That can lead to very unpleasant surprises -- there’s just nothing quite as disheartening when you’re running a successful Unix data center as being told some otherwise nice morning that: “oh, by the way, we’re merging with so and so and they’d like us to use their systems.” Followed, of course, by the real betrayal: “they’re smaller than us but they’re spending about ten times what we do on systems, so probably a lot better than what we have, eh?”

If you know about major corporate change early enough, and have the right organizational positioning and resources in place, you can initiate, and therefore control, the consequent IT change process.

Whether that change results from a hostile takeover, the board’s decision to sell a major division, or some user’s B2C brainstorm, the key to turning a potential crisis into a service opportunity is to know about it early and then seize the initiative.

In these situations, readiness is everything. To be ready to respond quickly to change opportunities means that both your organizational posture and your regular training, preparation, and contingency planning must support the ability to react rapidly, *and in unforeseen ways*, to situational change.

The traditional planning and development group doesn’t normally work as well as you would want for ad hoc adaptation. In my experience, at least, staff people work best when dealing with “trendable” change. What you need instead for ad hoc adaptation to new opportunities is people who are sufficiently, and justifiably, confident of their own skills to cheerfully evaluate and undertake serious operational risks without too much delay and introspection about possible negative consequences.

Since that’s a mindset you want to stamp out in the service delivery business the inherent contradiction would seem to create some staffing problems. In general, however, people are extremely good at dealing with contradictions like this and I recommend that the same people be trained to practice both roles.

Use cron to fire it off every night and manually clean up the directory every couple of weeks while being sure to leave some older files in place.

Then, when you get that user phone call accusing you of making SAS (or whatever) suddenly fail, you can re-run the "ls -laR" for the likeliest candidate file system, output the result to a file, and use diff or comm to tell you exactly what changed.

Once you know, resetting permissions, undoing the symbolic link, or recovering the file from backup is usually quite easy - Sadly this doesn't always work: a Sun patch cluster(!) that killed CDE on a workstation affected just over 19,000 files....

Tech Tip #15: "and by indirection find direction out."

Shakespeare presents Polonius as a babbling old fool, and that is probably just what some of the other people in your IT department think of you. After all they out number you and, besides, you still believe in Unix while they're fully qualified MCSEs; I mean, gee, isn't Unix so last millennium?

Not hardly, but you'll generally find it easier to meet and marry Alias girl than to convince a bunch of Wingots that the fundamental VMS ideas behind Windows 2003 Server predate those in Solaris 10 by 15 years.

So when the next interesting project comes up there'll be quite a a lot of support for treating it as an opportunity to get the latest Windows server products on yet another rack full of PCs and not much for doing it on Unix.

You could wait until the Windows project fails or is so far behind deadline that the bosses get desperate, and then hope to get asked to the dance; but there's a better strategy. Be helpful. Line up with the Wingots, support their hardware and licensing requests, contribute to their project planning, and supportively implement a parallel system --for testing and prototyping only of course-- on your Sun or other archaic Unix box.

If they're right and the Windows solution works for the organization --that's great: the organization gets a success and you have some new friends. Cool! but if you're right and the Windows solution starts to drag out, your new relationships with the key players will let Unix step in with something that, for a few hardware dollars and some code cleanup, can be made to work almost right away.

Everybody participates in the success, you get a new Sun box, *and you get to play good guy* --really, what could be better than that?

2 For a roving remote user, for example someone connecting from hotel rooms while traveling, the better approach is to process the user's email and scheduling information under a secure web server. Almost all major browsers have encryption capabilities built in, so a remote user just fires up the browser and creates a fully secure session by logging into the office https server.

   This takes some office set-up and requires an additional processing step on the server, but has a fixed cost that changes only with large jumps in the number of users. For one user the $5,000 or so required in the Windows server environment (it's free for a Linux server) is a lot; for 500, it's about the same as a can of Hub's roast peanuts.

Both, of course are cheaper and more reliable than using something like the CISCO VPN client for Windows 2000 but, sadly, the more common answer is for the wingots to insist on a Windows client and so use your choice of Linux at home as an excuse to wash their hands of the problem.

Option three gives you a sneaky way of addressing that attitude by making it a Windows networking problem, but it too requires some degree of co-operation by the office gurus.

VNC connects a client, or viewer, with a server but doesn't really restrict what those are, so people have done versions for devices including the Geo communicator and Palm Pilot - just in case you want to run your Windows 2000 based Excel spreadsheet from your playphone.

In fact, if both ends support true multi-tasking you can run both the server and the client simultaneously on each of two machines to produce an apparently endless recursion of screen images containing screen images - like holding a mirror before a mirror.

For this option you use either the freeware or shareware versions of the VNC product from AT&T's British research lab. This comes as two parts: a server which you run on the remote host (your office pc) and a viewer you run on your home machine.

VNC connects the mouse and keyboard on the remote host to their equivalents on the local, or viewer, machine while mapping the remote monitor to a local window. As a result it gives you your complete office desktop in a Linux window at home.

Setting up either version for default configurations is utterly trivial - basically click "install" and go. Getting it to work with secure shell and/or

Do you put a triply redundant machine that's proof against everything down to a nearby nuclear blast in place if it costs $125,000 to do a job easily handled by a V240 at $2,500? You do if lives depend on that machine running properly; you don't if the worst loss to be suffered is that a few web hits go somewhere else.

In hardware you mostly look for high value at low cost; things like server redundancy for a mission critical business application. Sometimes, however, even tiny reliability gains are worth real money relative to the cost of the systems protected.

One such place not to compromise is the identity (login and security) server where upgrading reliability to the best you can afford is nearly always justified, even for small companies --because a very small percentage change, say from 99.98% to 99.99% in annual system uptime, can have very significant impacts in terms of user time and frustration. When the company wide single login server is at stake the gap between one incident and three can mean your job.

I usually put the identity server on a pair of small Solaris SPARC machines, like Sun's 450 or 250, -making sure they're at least a generation behind current technology because:

1. The key developers use this environment for testing and therefore release critical security updates or patches for it first; and,
2. Older hardware has usually been completely debugged. That, in turn, means that a transitory glitch won't get classified as a possible hardware or OS bug and so allow an attack to go unnoticed.

For most other production gear, however, simple redundancy and hot swaps provide the two most critical defence layers.

Hot Swaps

Most modern systems allow hot swap - the process of replacing parts like CPU boards and disk drives without shutting down. Note that this approach implies the use of hardware level RAID storage or mirroring to protect against single drive failures.

On larger multi-processor Solaris machines you can, in fact, carry this idea forward to the point of upgrading the operating system and swapping CPUs without rebooting or stopping applications processing

Those conditions both include and assume:

1. Very long time frames. A typical business case in support of a development proposal might, for example, envision a ten year operational lifespan for the application and place initial systems delivery two or even three years into the future;
2. Stable, predictable, budgets within stable, predictable, corporate business plans. That same business case, for example, would often have tables showing the expected cost savings at break even --seven or more years into the future; and,
3. A simple enough business process that both systems and management staff can clearly understand, and communicate, what the application should do.

As a result the management processes built on those assumptions rapidly lose applicability as application complexity increases and time frames decrease.

Another way to see the effect of changing the systems role

In the traditional organizational model, systems functions as a corporate service maintained and controlled independently of the production and sales functions. The Unix CIO, however, strives to make systems functions so integral to daily operations that they cannot be distinquished from them.

Sun's current marketing mantra that "the network is the computer" reflects an early eighties view of this and should now, I think, be changed to "the company is the computer" to predict the role changes Unix will see over the next decade or so.

That, of course, does not mean that people who learned their trade twenty years ago don't keep right on trying to make the process work where it doesn't apply; it only means that about 95% of large in-house application development efforts fail to meet all identified user needs; about 75% are considered failures by their sponsors; and about 60% are simply discarded or abandoned before implementation.

Most reviews of such failures place the blame on things like a failure of communication between users and developers; on changing or growing requirements; and on technical failures within systems such as inadequate testing.

INDEX

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

The Unix Guide to Defenestration

This book explains that most commercial systems work disappoints because the incentives favor exactly the kind of continuous low level failure we usually see. Systems management careers are enhanced by budget growth and staff expansion, both of which are maximized by maintaining a level of non performance that's just short of catastrophic; and correspondingly hurt by the successful execution of the cost reduction, or cost avoidance, mandates that usually go with the job.

Spend enough money as a CIO and you'll make somebody's annual "100 best IT users" lists, but deliver services with the effectiveness and reliability of the phone system while meeting your mandate to cut costs and you won't get promoted; you won't be attending executive committee meetings; and most people −−including recruiters, your colleagues, and your bosses −−will dismiss you as a loser whose budget and visibility are fading while theirs are growing.

The organizationally right answer seems simple: reposition systems as a profit center instead of a cost sink, provide incentives for personal success that align with corporate goals, and stand aside while your systems people grab hold of the stuff that works and throw out the stuff that doesn't.

We've all seen this recipe work. Embedded systems developers, like most of the freeware community, have personal incentives and business controls that align directly with those of the end user −and as a result the software for your car engine, video camera, network router, or Linux server will work correctly and reliably all day, every day −−at a tiny fraction of the cost of the software on your PC.

This book's message, for business executives and systems people alike, is that getting the house in order is the system manager's first step in that transformation and starts with a hard clear look at the issues, choices, and technologies involved. That's the technical focus of this book: improving service, and saving money, by understanding and adopting the structures and technologies that work and throwing out those that don't.

As a result there's lots of practical advice here on making things work; but there are also war stories and hard lesson commentaries that bring out the moral and ethical issues facing systems staff who try to get the job done in an environment where the incentives, and therefore most of the other people, favor failure

ISBN 0-9689004-0-2
52995
9 780968 900406

US $29.95
Cdn $47.95

Get current updates and join discussion forums at winface.com