

ITPro™  
SERIES

Windows  
& .NET MAGAZINE

eBooks

# Best Practices for Managing Linux and UNIX Servers

By Dustin Puryear

 netiq  
Work Smarter.



## Table of Contents

<b>Chapter 1 Introducing Best Practices</b> . . . . .	<b>1</b>
Systems Managers vs. Systems Administrators . . . . .	1
The Need for Best Practices . . . . .	2
Increased Security . . . . .	2
Increased Reliability . . . . .	2
Increased Cost-Effectiveness . . . . .	3
Implementing Best Practices . . . . .	3
A Holistic Approach to Best Practices . . . . .	3
Infrastructure and Data Security . . . . .	4
Backup and Restoration . . . . .	4
Change Management . . . . .	4
Performance Management . . . . .	5
User Management . . . . .	5
Fault Management . . . . .	5
Task Automation . . . . .	5
Defining Policy: The Crucial First Step . . . . .	5
How to Organize a Policy . . . . .	5
Service Level Agreements . . . . .	7
Knowing UNIX: Overview and Idiosyncrasies . . . . .	8
Linux and UNIX are Diverse . . . . .	8
Sidebar: What's In a Name? . . . . .	8
Unified Deployment . . . . .	9
Unified Management . . . . .	9
File-Centric Resource Access . . . . .	9
Simplicity Found in Complexity . . . . .	9
The Final UNIX Truth: Automation . . . . .	10
Knowing Your Infrastructure . . . . .	10
Infrastructure Servers . . . . .	11
Data Servers . . . . .	11
Application Servers . . . . .	12
Interactive Servers . . . . .	12
Workstations . . . . .	12
Managing UNIX Management . . . . .	12
Ticket Systems . . . . .	13
Server and Application Documentation . . . . .	14
Installation, Configuration, and Recovery . . . . .	15
Service Layout . . . . .	16
Network Layout . . . . .	17
Conclusion . . . . .	18

## Chapter 1:

# Introducing Best Practices

Managing Linux and UNIX environments can be difficult. For administrators with years of UNIX experience, this statement might seem paradoxical because UNIX is an OS that is well suited to automation and large-scale deployments. But several factors contribute to both the perceived and real difficulties in using and managing UNIX. These factors include a complex administrative interface and the variations across systems that Linux and UNIX fragmentation (or perhaps more appropriately, divergence) causes.

These difficulties, combined with inadequate systems management training, can lead to a poor set of UNIX management procedures. The types of failure that improper management can cause include insecure installations, lack of availability in services, and unnecessarily complex and expensive management infrastructures. These failures affect end-user experiences and have an immediate and significant effect on UNIX systems' Return on Investment (ROI).

This ebook addresses Linux and UNIX systems management. That is, I focus on the how's and why's of managing a UNIX environment. This ebook will help you develop an understanding of existing systems management best practices and learn how to implement those practices in your environment.

The term *best practices*, as related to systems management, needs defining here. In most professions, including systems management, *principles* and *practices* determine how to best accomplish a goal. A *principle* is a basic truth or standard (e.g., a basic truth about how to manage Linux and UNIX systems). A *practice* is the way in which you implement a basic truth.

To better understand principles and practices, let's consider an example. A primary security principle is defense-in-depth. This concept states that you need multiple layers of defense between an attacker and a target. These layers might include a series of firewalls and hardened OSs and applications. The security manager's solution (i.e., firewalls and hardened systems) is the practice that implements the defense-in-depth principle.

In terms of Linux and UNIX best practices, we aren't concerned with just any practices. We're interested in the set of practices that have proven to be the best way to implement the principles that govern Linux and UNIX systems management.

Although principles rarely change, practices often change over time. This ebook discusses the current best practices for Linux and UNIX systems management. I cover best practices that implement several core Linux and UNIX management principles.

In this chapter I discuss several topics that are important to your understanding of how to apply Linux and UNIX best practices. For clarity throughout the book, I typically use the term *UNIX* rather than *Linux and UNIX* to refer to all UNIX and UNIX-like systems.

## Systems Managers vs. Systems Administrators

This ebook is targeted toward systems managers rather than systems administrators. A *systems administrator* performs the daily administration tasks for a UNIX system. These tasks can include reviewing log files, installing new servers, verifying backups, and creating user accounts. A *systems manager* oversees the design and implementation of a UNIX environment. Although duties can overlap

between systems administrators and systems managers, the goals of the two roles are different. In short, a systems administrator is concerned with the daily management of a UNIX system, whereas a systems manager is concerned with long-term management. Although this ebook will provide systems administrators with a solid foundation in planning and executing both short- and long-term management tasks, systems managers will most benefit from the book.

### **The Need for Best Practices**

Most companies have a set of internal procedures that the company has developed over several years. Best practices, especially as I discuss in this ebook, don't necessarily take precedence over a company's internally developed procedures. Instead, a set of best practices is an adjunct to a set of internally developed procedures. Best practices should provide a company with a more solid foundation on which to customize industry-wide practices to its needs.

Best practices often focus on higher-level issues than internally developed procedures do. Because of their community- and industry-wide development and testing, best practices offer companies more ideas and solutions than internal procedures offer.

Most people can verbalize the importance of following best practices but don't understand how doing so will affect their organization's performance. In regards to Linux and UNIX solutions, following best practices increases the security, reliability, and cost-effectiveness of those solutions, as I discuss in the following three sections. Moreover, security, reliability, and cost-effectiveness directly and immediately affect the services an organization provides.

### ***Increased Security***

In years past, organizations running UNIX have been attacked, either directly or indirectly, by various methods. These attacks range from deliberate Distributed Denial of Service (DDoS) attacks to internally mounted espionage attempts through hacking or exploitations of network file system weaknesses, such as the weaknesses in NFS.

Over time, entire sectors of our digital infrastructure, ranging from the military-industrial complex to higher education institutions, have adjusted their methods of managing UNIX systems to reduce their exposure to these attacks. The solutions employed have either succeeded or failed. Solutions that fail are thereafter rejected, whereas successful solutions merge and further develop into best practice solutions for protecting UNIX systems against attacks. These methods have slowly been absorbed into UNIX systems management best practices not because of regulations or law but because doing so made sense.

No single best practice exists for security or any other area of systems management. Therefore, best practices typically include a range of solutions for a variety of problems within a particular area of systems management.

### ***Increased Reliability***

Much like security best practices have evolved over time, UNIX reliability best practices have developed through years of systems managers' and front-line systems administrators' experience. Organizations with large UNIX environments have tried many reliability solutions and have incorporated the most effective solutions into reliability best practices. Using those best practices lets you take advantage of others' work and experience.

An example of a best practice that affects a UNIX network's reliability is centralized configuration management. Before the widespread adoption of UNIX and other mini- and microcomputer OSs, most systems were based on mainframes. These systems offered just one point for services configuration because only one machine existed to manage. But as UNIX systems spread, managing multiple systems became more important.

Unfortunately, managing many systems becomes more complex as the number of systems increases. Systems administrators eventually pushed for the ability to manage system configurations from a central location. They wanted revision control to track changes and to push changes out to the systems they managed. This method reduced the number of systems a systems administrator needed to visit when implementing changes. In addition, systems administrators had a better way to back up and restore configurations to systems. These changes translated into more reliable systems, because systems damaged by bad configurations were easier to restore and systems administrators made fewer mistakes.

### ***Increased Cost-Effectiveness***

The most important contribution you make to your company is its profitability. If the value of the services that your UNIX environment provides exceeds the cost of providing those services, then you are positively affecting your company's position in the global marketplace.

One of the most logical reasons for using best practices is that doing so saves money. Best practices have been developed externally, field-tested for you, and approved by a large portion of the UNIX community. Following best practices reduces or eliminates your need to extensively test most management methods. Instead, you can focus on previously proven solutions.

## **Implementing Best Practices**

When a company encounters a set of best practices that will alter systems management, the staff often is reluctant to change. This desire to maintain the status quo is natural. When considering best practices, you must examine on a case-by-case basis which solution works best for your environment: best practices or internally developed procedures. Best practices usually prevail because they are widely accepted by others in the industry and because companies feel pressure to follow industry standards. However, following internal procedures might work best in some cases.

If you determine that following best practices is a better long-term solution, you need to create a program for a well-documented switch from your company's internally developed procedures to the methods that best practices define. In addition, you should convert to the new methods slowly rather than make sudden and radical changes. Methodically switching from one set of procedures to another lets you easily monitor the results of your changes and quickly abort changes that produce negative effects.

An important best practice to follow is to document on paper the procedures your company follows for systems management (and indeed for any task necessary to the company's operation). A useful test to determine which information to document is to imagine what would happen if a key employee vanished from the company.

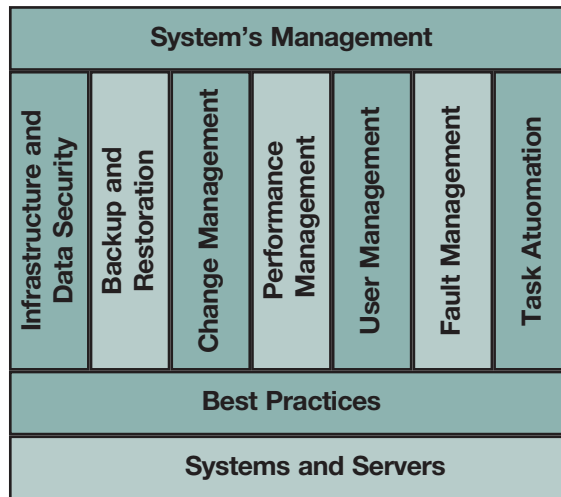
## **A Holistic Approach to Best Practices**

Managing UNIX is complex, and no single book can comprehensively cover the topic. Instead, I focus on core issues that face every UNIX systems manager. Figure 1 shows the seven UNIX

management topics that I discuss and how they relate to best practices for managing your systems and servers. I discuss these seven management topics in Chapters 2 through 8.

**Figure 1**

*Areas of Focus*



### ***Infrastructure and Data Security***

Despite what IT staff might say, IT departments often overlook security. The reason for this oversight isn't always funding but can be a problem of priorities. Because support staff is under constant pressure to maintain and create new and innovative services, they sometimes don't seriously consider how to securely build and maintain those services until the last minute—if at all. In Chapter 2 I concentrate on how to design, build, and maintain security in modern UNIX networks' physical and electronic realms, with an eye toward disaster recovery.

### ***Backup and Restoration***

Performing backups is a task that systems managers love to hate. Deploying a comprehensive backup procedure can be expensive, time-consuming, and labor intensive. Yet a good backup procedure is a vital component of a management plan. In Chapter 3 you will learn how to best design, implement, and verify reliable backups for UNIX systems.

### ***Change Management***

One of the most difficult aspects of systems management is managing change. The core concept of change management is to have the ability to properly schedule, execute, and verify changes to systems. Chapter 4 focuses on how to perform the actual scheduling, execution, and verification of changes in your UNIX environment and how to minimize the disruption that those changes cause your end-users.

## ***Performance Management***

Most environments don't have a well-defined and executed performance management policy and procedure in place. This lapse is surprising, considering the true cost of a poorly performing system. Chapter 5 discusses the performance information you need to monitor in a UNIX system, as well as how to utilize the information you gather to tune your environment and maximize performance.

## ***User Management***

One of the most complicated issues that organizations face is user management. User management includes not only creating and deleting accounts but also monitoring user activity and assigning roles and rights, among other concerns. Chapter 6 discusses these issues.

## ***Fault Management***

Fault management focuses on detecting errors in your UNIX environment, such as failing disks, resource overutilization, and key processes stopping (e.g., the Apache httpd Web server daemon failing on a Web server). The information in Chapter 7 will help you quickly and efficiently respond to faults and exceptions in your UNIX networks.

## ***Task Automation***

Task automation is the process of developing tools to handle repetitive tasks (e.g., restarting a stalled process, distributing account information about a new user to your servers). To accomplish task automation, you can use scripting tools such as the Bourne shell and Perl, as well as higher-level management tools from companies such as NetIQ and Hewlett-Packard (HP). Task automation is nearly synonymous with UNIX because UNIX provides a vast toolset for systems administrators to perform daily maintenance and systems managers to provide a more comprehensive management infrastructure. Chapter 8 integrates ideas from Chapters 2 through 7 into a more comprehensive set of policies and procedures for automating tasks in UNIX.

## **Defining Policy: The Crucial First Step**

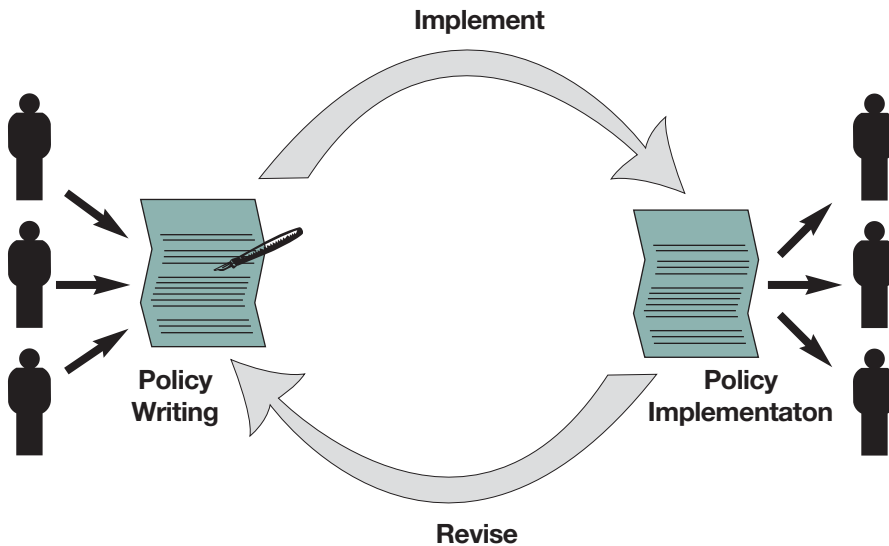
Policies should determine how your company uses computing resources. Resource use includes everything from end-user use to change management. Policy dictates procedures and guidelines, which in turn implement the policies' directives. To adequately discuss best practices in UNIX environments, we must begin with policy development.

Depending on legal and regulatory needs, a policy might be focused and straightforward or elaborate and often difficult to implement. The policies I discuss are specific to the needs of a UNIX systems manager who focuses on properly managing his or her UNIX environment to ensure high uptimes and service availability; these policies don't take into consideration a specific industry's legal requirements. You should be able to deploy the policies from this ebook in your organization. The policies that I define here will drive all the procedures that I discuss later in the book. Your company should follow the same rule: Procedures should be derived from policies.

## ***How to Organize a Policy***

Policies range from simple and effective to overly complex and ineffective. The policies in this ebook are straightforward and follow a standard, consistent format. Figure 2 shows the policy creation cycle.

**Figure 2**  
*Policy Creation Cycle*



Developing a usable policy typically involves a loop of two phases: policy writing and verification. In the policy writing stage you put pen to paper and work with all the involved departments to develop the overall policy structure and content. During verification the policy is put into place and field-tested. As you get feedback about the policy's real-world effects, you revise the policy. After you make revisions, you again put the policy into effect. This cycle continues for several iterations and can last from a few months to a few years, depending on your organization's size and how thoroughly a policy must be designed and written.

The need for this cycle of policy writing and revision is perhaps best illustrated with an example. I once worked as a UNIX systems manager for a company that developed proprietary software. This company needed a comprehensive security policy that controlled source code distribution.

After obtaining management approval, I started writing the policy. I worked with the software development team's head and other key company staff to develop the policy. When we implemented the policy and compared it to the company's daily activities, we quickly discovered that the policy didn't cover all software distribution methods (e.g., via email to an offsite programmer). We embarked on another round of policy writing and approval, then a new implementation. We finally achieved a policy that worked with how the company operated, rather than forcing the company to change its processes to follow a policy that didn't specifically address the company's day-to-day needs.

A policy has several important sections: Policy Summary, Responsible Parties, and Policy. The Policy Summary is simply a summary of the policy in question. This area should succinctly list the topics the policy covers. The Responsible Parties section lists the people or roles that own and are authorized to enforce the policy. Finally, the Policy section contains the actual policy.

## Service Level Agreements

Although they aren't directly related to best practices, service level agreements (SLAs) can be powerful tools in your collection. One way to define SLA is a policy between a resource provider and a resource consumer that determines an acceptable level of service. The SLA might contain a series of penalties against the service provider if the specified performance levels aren't met. SLAs define the minimum level of service that is acceptable to the consumer. For a UNIX network, such services might include email, Lightweight Directory Access Protocol (LDAP), or file storage.

Defining the service is typically easy. A more difficult task in establishing an SLA is defining the minimum level of service. For example, suppose that an SLA covers the service of access to an LDAP directory. The UNIX Services department centrally maintains this directory, which in turn powers several applications maintained across the organization. If access to the directory is impossible, obviously the SLA is violated. But what if users are able to connect, although a response takes several minutes? Because of this type of scenario, most SLAs define service levels in two ways: uptime and response time.

Uptime is the amount of time that a service is available and usable. Note that usable doesn't mean fast or responsive—simply that the service is up and available.

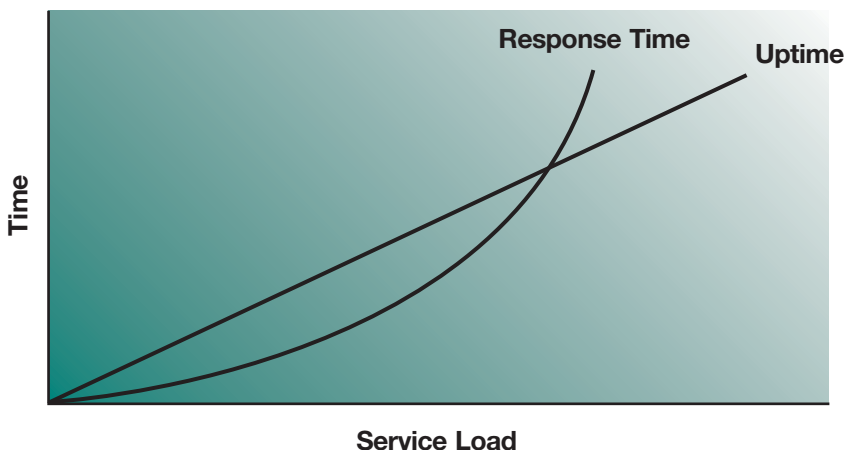
Most users care more about response time than uptime. Response time is the amount of time that a service takes to respond to a request. In a larger context, response time is what most people mean when they talk about end-to-end measurements, which are important in performance management.

Service load is how heavily a system is used. For example, service load can refer to how many services are running, as well as to the CPU, memory, and disk load.

As Figure 3 shows, sometimes no direct relationship exists between uptime and service load. However, a relationship always exists between response time and service load.

**Figure 3**

*Service Load, Uptime, and Response Time*



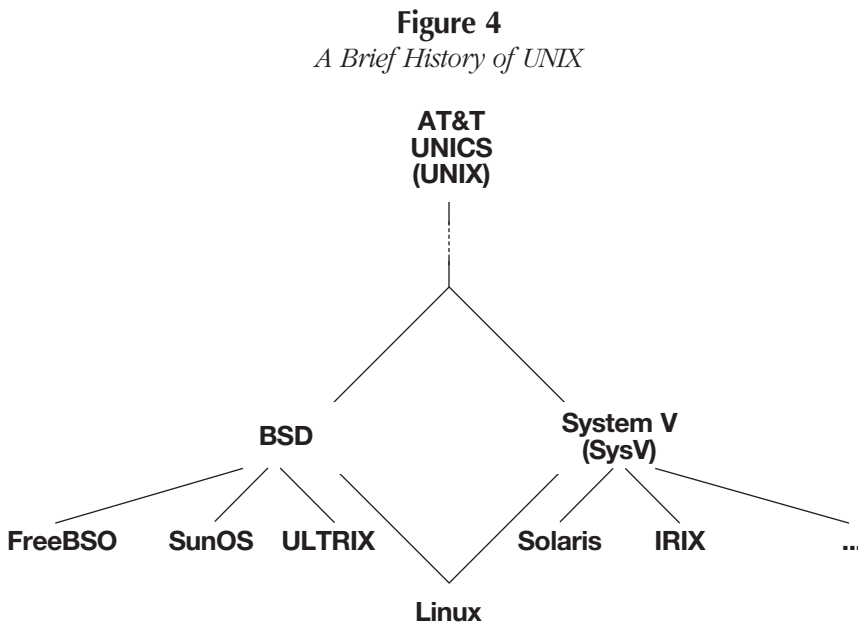
This ebook doesn't directly address SLAs. However, these agreements can form a powerful foundation for communication and even funding between your services group and other departments.

## Knowing UNIX: Overview and Idiosyncrasies

Now that we've covered the policy side of systems management, let's consider some issues that affect UNIX manageability.

### *Linux and UNIX are Diverse*

UNIX isn't an OS so much as it is a universe of OSs. (To learn what the UNIX label means, see the sidebar "What's In a Name?") Unlike most other systems, UNIX offers an incredible range of variation in implementations. As Figure 4 shows, all UNIX implementations trace their roots to the original UNIX developed at AT&T Bell Laboratories.



### What's In a Name?

Although most people consider UNIX to be a single OS or the entire set of UNIX and UNIX-like OSs, the word (or brand, as The Open Group notes) UNIX has a specific meaning. To receive the UNIX label, a vendor must pass The Open Group's (<http://www.opengroup.org>) UNIX certification program. This certification requires that the OS support the Single UNIX Specification, pass a series of The Open Group's tests, and meet various other requirements. Obtaining the UNIX brand can be a long and expensive process. UNIX-like OSs such as Linux might meet most or all of the specifications necessary to be a UNIX-branded OS but still not *be* UNIX until the certification process for that specific Linux distribution is complete.

Over several years, the original UNIX split into two camps: BSD and System V. Alas, even this split wasn't clean. A wide assortment of UNIX systems branch off the two main branches, some merge back into a branch, and others (such as the UNIX-like Linux) live between the two branches—offering the best and sometimes the worst of both worlds.

Understanding UNIX's diverse nature is important in learning how to manage UNIX systems in an enterprise environment. Although diversity lets UNIX implementers focus their optimizations and development efforts on solutions for specific markets, diversity also increases the management load for organizations with multiple UNIX systems to support.

Several solutions exist for managing the problems that UNIX diversity creates. I discuss many of these solutions later in the book, but here I discuss two of the most popular approaches: unified deployment and unified management.

### **Unified Deployment**

Organizations that follow the unified deployment tactic make concerted efforts to support only one flavor of UNIX. This solution is typically possible only with mainstream UNIX (e.g., Solaris, AIX) or a widely accepted distribution of Linux (e.g., Red Hat). Organizations that benefit from having various UNIX systems might not be able to take this approach, particularly if they tend to acquire or merge with other companies.

### **Unified Management**

A more complex but often surprisingly more cost-effective solution is unified management. In this approach, one management infrastructure controls a company's various UNIX systems. Unified management, which is the tactic I use in this ebook, essentially homogenizes the management of a heterogeneous UNIX environment.

### ***File-Centric Resource Access***

A UNIX peculiarity is the almost obsessive focus on offering a file-centric interface to the OS. This focus is evident in several areas, from the heavy reliance on text-based configuration files to using device files in /dev to access devices. This file-centric view of the world has greatly contributed to UNIX's success. UNIX offers a cohesive interface because the OS typically provides developers with one core interface for most services and systems administrators with a consistent method of using text-based configuration files to configure services. This structure lowers development costs, lets developers and administrators leverage their knowledge across systems, and increases the portability of software across UNIX flavors.

### ***Simplicity Found in Complexity***

One of UNIX's strengths is the simplicity in the OS's apparent complexity. UNIX is a set of tools built on top of a kernel that gives systems managers a robust collection of simple tools to use together to create powerful results. In contrast, server OSs such as Windows 2000 typically offer larger and more complex management tools that address entire ranges of issues rather than having each tool focus on a specific task.

## ***The Final UNIX Truth: Automation***

As the previous sections imply, the key to understanding how to best manage UNIX systems is to understand automation. Other OSs, such as Windows 2000 Server, don't immediately offer the same high level of automation that UNIX has. Although new Windows system tools are constantly under development, until recently no Windows push existed for the same types of built-in, custom-developed, and commercial automation applications that are available for UNIX.

If you aren't automating your UNIX system's management, at least for mundane and routine tasks, then you're wasting resources performing work that the system should be doing. Examples of tasks you can automate are log filtering and alerts, resource usage alerts (i.e., CPU and file system overuse), and process monitoring. This ebook continually discusses ways to automate routine processes that can keep you updated on system status and help you build reports to enhance your organization's long-term growth.

## **Knowing Your Infrastructure**

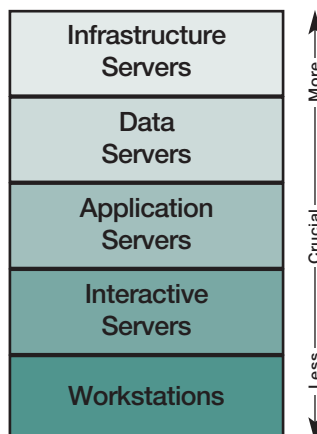
The UNIX systems you manage each perform some type of work. You can use the type of work that each machine performs to group the machines into classifications.

Classifying systems can be a difficult task. Many organizations manage multipurpose servers that perform multiple functions. Often, systems that offer firewall service for a remote location also offer other services such as email and a proxy or relay for authentication services.

Figure 5 shows the hierarchical classification system that I use throughout this ebook. A benefit of this type of system is that you can easily assign priorities for failure response time and general funding based on which level each classification is in. For example, a system in the Workstation layer will probably receive less attention if it goes down than a system in the Infrastructure layer would receive if it failed. Although this approach is common sense, making the classification concrete can be helpful in several instances, (e.g., budget discussions).

**Figure 5**

*Hierarchical Role-based Classification System*



Before I explain each classification, you need to know what kind of criteria determine a system's placement within the classifications. Classifying systems can be difficult when services are offered on multipurpose servers.

One classification approach is to identify the most important service a system offers, then place the system into the classification for that service. So, if a system offers IP routing between networks, the system would belong in the Infrastructure layer—regardless of which other services the system provided. This approach is simplistic and can lead to a misappropriation of resources because systems' importance is overemphasized or underemphasized. A solution to the problem of classifying systems is to move services for each classification layer to individual machines, or to group onto one machine the services that share a classification (providing for redundancy, of course).

### **Infrastructure Servers**

An infrastructure server provides the services that comprise a UNIX environment's foundation. DNS is one of the most common services an infrastructure server provides. Any service that the network requires for proper operation is usually classified as an infrastructure service, and any server running such a service is an infrastructure server. A central concept behind the infrastructure classification is that if an infrastructure server fails, and no redundancy is built into the network, then several other services and possibly even network clients might fail.

Table 1 lists candidates for the infrastructure classification. As an example, consider the LDAP service. Many UNIX environments use LDAP as an authentication service; in these instances, authentication would fail without LDAP. LDAP is obviously an infrastructure service because it provides a core service that higher layers in the hierarchy require.

**Table 1**  
*Infrastructure Service Candidates*

<b>Service</b>	<b>Function</b>
Administrative	Provides centralized servers for network management, server configuration, and forced failover.
DHCP	Provides IP address information to DHCP clients.
DNS	Provides domain name-to-IP address and IP address-to-domain name mapping.
LDAP	Provides logon information for users and services.
Network Information Service (NIS)	Provides logon information for users and services and distribution of configuration maps.

### **Data Servers**

Data servers usually fall into one of two categories: file servers and database servers. File servers provide network access to a file system, whereas database servers provide an abstracted interface to a set of data that the database server manages. In most cases, particularly with NFS in the UNIX world, file servers also provide an abstracted interface to the data in the file system—much like Server Message Block (SMB) and Common Internet File System (CIFS) provide users with a file system-independent method of accessing files on a Windows file server. (To be file system-independent means that the client doesn't need to understand NTFS to use Windows networking to access files on an NTFS file system over the network.)

### ***Application Servers***

Application servers are the computing world's middlemen. An application server is the server that provides the interface between a user and the back-end databases and processing servers. The interface is the method the user uses to get data into and out of the system. The application server also provides the business logic that determines what data is valid, how to format the data going into and out of the back-end databases and servers, and how to secure access to only authorized users.

The most obvious example of an application server is a Web server that an e-commerce Web site's customers access. In such a case, the Web server software Apache might provide the layer between users and the back-end databases. In addition, the Web server is where the Web site's e-commerce business logic runs, through a programming language such as PHP or ColdFusion.

### ***Interactive Servers***

An interactive server, or interactive logon server, provides users with remote logon capability. Telnet and Secure Shell (SSH) servers are interactive logon servers. Interactive logon servers let users access UNIX-based applications, check email, and use command-line tools such as Lynx (a text-based Web browser) to access the Internet. Servers that provide remote users with the graphical X Window System interface are also considered interactive servers because they give users and customers remote access.

Interactive servers have specialized needs. Because interactive servers give users local access to the file system and software, these servers have increased security vulnerability. In addition, interactive access servers often require specialized performance tuning. For noninteractive servers, tuning usually focuses on maximizing bandwidth and work done per unit time. For interactive systems, the focus is usually on reducing latency—often at the expense of bandwidth usage.

### ***Workstations***

Workstations, or desktops, typically fall at the bottom of the classification system. A workstation lets a user run programs on the local CPU and memory; a workstation usually has a local file system. (An example of a workstation that doesn't have a local file system is an X terminal that has only enough resources to bring up an X server.) Depending on your company's organization, workstations can be cornerstone components that require more attention than the organization in Figure 5 implies.

## **Managing UNIX Management**

One of the most important changes you can make when considering how to best manage your UNIX installation is to alter the way you manage your UNIX management. That is, you need to evaluate not only how you administer individual machines but also how you manage the entire network of UNIX systems.

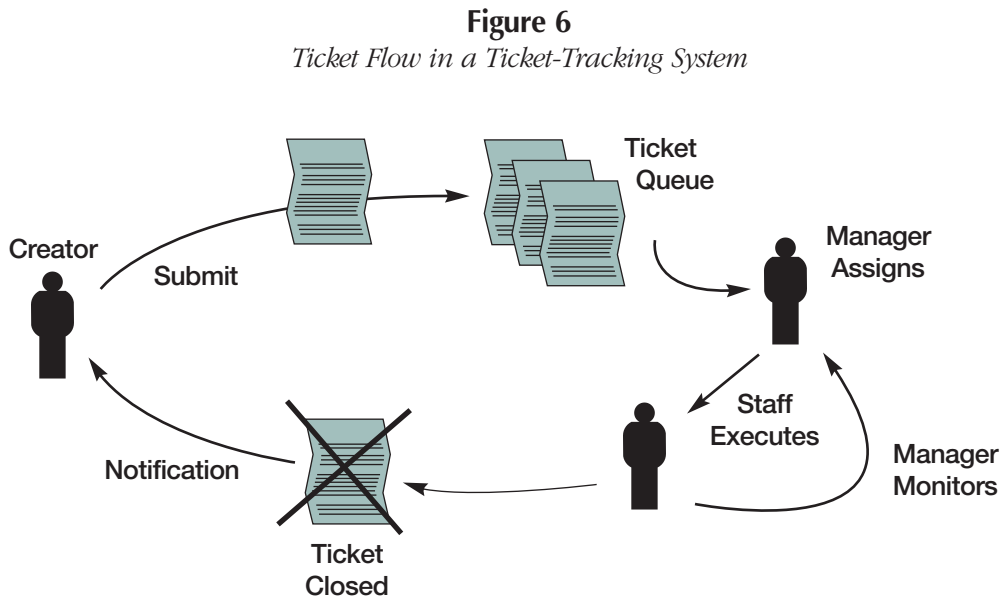
If you approach systems management as you would approach systems administration, you run the risk of letting your staff devolve into merely a fire-fighting brigade. But if you view systems management with a holistic approach that focuses on problem avoidance rather than problem solving, you can fight the fires that break out, as well as adequately handle your business's long-term growth.

In general, long-term management solutions always include solutions to short-term needs. For example, addressing long-term backup, data integrity, and disaster-recovery needs also lets you quickly solve server failures.

## Ticket Systems

One way to work toward problem avoidance rather than just problem solving is to track the problems your organization faces. Over time, you can analyze the data you gather and refine your management framework to better avoid the problems that continually occur. Using a ticket system helps achieve this goal.

A ticket system is a set of software that accepts a Help or change request from a user for review by a technician or manager. The ticket can contain just one problem or a long-term project synopsis. In either case, the ticket lets you track the problem and solution's progress as various employees and vendors work toward the solution. Figure 6 shows the flow of a ticket in a ticket-tracking system.

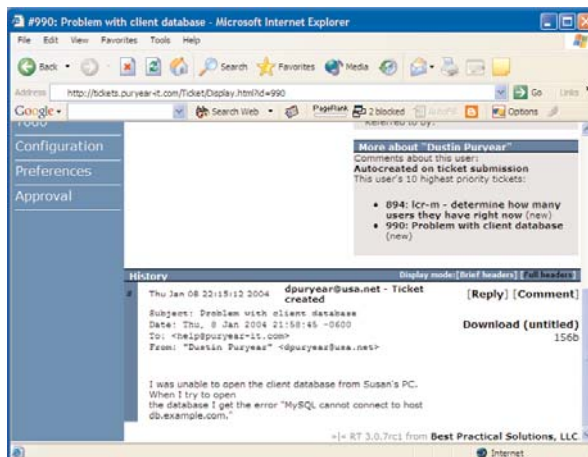


Ticketing systems vary and include commercial systems such as BMC Software's Remedy and open-source solutions such as Best Practical Solutions' Request Tracker (RT) and Open Ticket Request System (OTRS). Ticketing systems for smaller organizations also exist (e.g., PerDesk). These systems aren't as full-featured as other solutions, but they are easy to deploy and use.

An important benefit of using a ticketing system is the ability to use a standard format to monitor the progress of multiple tasks and assignments from a centralized location. You can immediately determine task priorities and see who is working on each task and where each person is in the solution process. In addition, you can easily reassign tasks, change priorities, and alert users of their ticket status.

Figure 7 shows an open ticket in RT. Notice the amount of history available. Also note the ability of anyone working on the ticket to correspond directly with other ticket owners (i.e., those assigned to work on the ticket) and creators (i.e., whoever initially input the Help request).

**Figure 7**  
*An Open Ticket in RT*



## Server and Application Documentation

Documentation is one of systems managers' and systems administrators' most neglected yet most important responsibilities. A systems manager must ensure that a methodical approach toward documentation is embedded in the management team's culture. Several forms of documentation must be maintained, including the three I discuss in this section: Installation, Configuration, and Recovery; Service Layout; and Network Layout.

When writing documentation, you need to consider your audience. If the document reader is a frontline technician (rather than a mid- or high-level administrator) who reinstalls remote office servers in remote locations, be sure to include sufficient details and background information.

You also must realize that documentation is often useful outside its original scope. For example, an auditor researching a system installation might decide to review the installation document to assess for variance between the system installation and the documented installation. Your documentation needs to consider future readers, or at least direct these individuals to additional information that addresses their needs—perhaps information available in other documents.

Documentation must be accessible to the target community. If you place the documentation for a network installation on the network, accessing the information will be difficult or impossible if the network is down. When you build and implement a documentation system, be sure to include a procedure for offering multiple forms of access. Two common methods are to offer access over the network, in the form of a Web and FAQ site or a file server, and to place printed copies of each document in a well-indexed documentation file. You might want to create a core document set for IT access at your main site, and maintain essential documentation at each remote site. At each remote site you would then give at least one person, preferably a site manager, the role of documentation maintainer. The documentation maintainer works with the central site to ensure that the remote site's copy of the documentation remains current and accessible to anyone who needs it.

Finally, remember the most important reason and one of the most powerful incentives for creating and maintaining documentation: You can go on vacation!

### Installation, Configuration, and Recovery

Installation, Configuration, and Recovery documentation concerns how to install and configure a new system with the designated services available for local or network use. Recovery is part of this type of documentation because the information is often necessary in crisis situations, in which you must quickly install and configure a new server because an existing server has failed catastrophically. Figure 8 is an example of this type of documentation.

**Figure 8**

*Example Installation, Configuration, and Recovery Documentation*

#### **Installation, Configuration, and Recovery**

##### **Documentation History:**

**Server:**

**Function:**

##### **Quick Install**

This area details how to quickly install the server, typically from a system image available on a network system image server (e.g., Red Hat Quickstart).

##### **Full Install**

This area details how to install the server without the Quick Install steps. This information would be used if you were installing the network installation server or if the network installation server weren't available.

##### **Service Configuration**

Service: Email Server

This server provides SMTP and POP3 mail service to the local user population.

1. Install the latest Postfix via RPM (<http://www.redhat.com/...>)
2. Configure main.cf with:
3. ...

Service: LDAP Server

This server runs OpenLDAP to cache the master LDAP directory for use by the local SMTP and POP3 mail servers. The local OpenLDAP server is used to increase performance and reduce load on the master LDAP server.

1. Install OpenLDAP via RPM (<http://www.redhat.com/...>)
2. Configure /etc/ldap/slapd.conf with:
3. ...

An alternative approach is to not include the detail service configuration section in each server's Installation, Configuration, and Recovery document but to instead maintain unique service configuration documentation for each service and to reference that documentation in each server's Installation, Configuration, and Recovery document. A benefit to this approach is that you don't need to maintain several copies of the same information in multiple Installation, Configuration, and Recovery documents. A disadvantage is that you still need to write customized instructions for each server if your installations vary across machines. Figure 9 is an example of this alternative documentation.

**Figure 9**

*Example Alternative Installation, Configuration, and Recovery Documentation*

Installation, Configuration, and Recovery  
**Documentation History:**  
**Server:**  
**Function:**

**Service Configuration**  
 Service: Email Server  
**First Reference Document:** Email/LDAP Service Installation  
**Local Modifications:**  
 This server requires a specialized configuration that requires ...

**Service Layout**

Service Layout documentation is similar to Network Layout documentation but focuses exclusively on overall service use within a company. Systems managers use this type of documentation for tasks such as monitoring for required patches and determining which servers a possible service outage or upgrade will affect.

Unlike for Installation, Configuration, and Recovery documentation, a company typically has only a few Service Layout documents. Many organizations have only one master Service Layout document and one detail sheet that specifies service versions. Complex networks might require a master document and several detail layout documents for services that are related to one another (e.g., an e-commerce Service Layout document that describes Web, mail, DNS, and NFS services).

Figure 10 shows an example Service Layout diagram for a sales company’s e-commerce division. Notice that the diagram doesn’t include great detail. A systems manager would use this diagram to quickly identify areas of concern when making service changes and to determine which related services were affected.

**Figure 10**

*Example Service Layout*

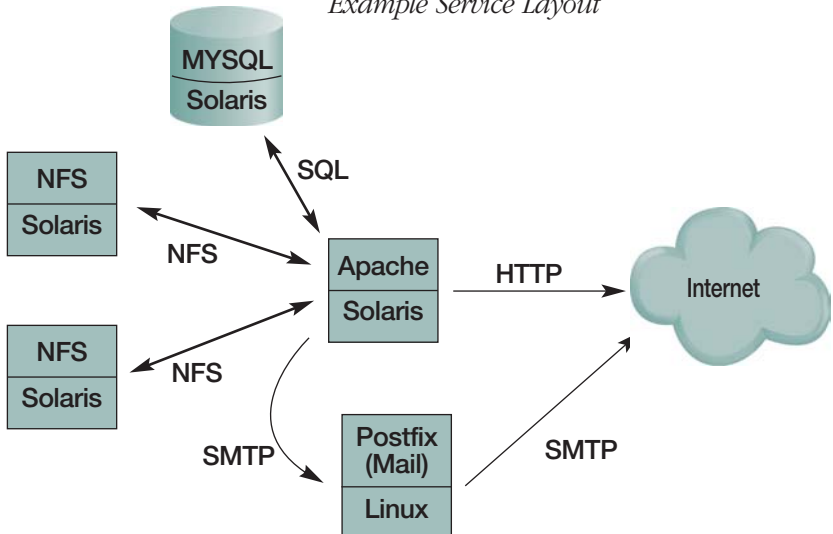


Table 2 shows the Service Layout detail document that would accompany the Service Layout diagram in Figure 10. This detail document identifies versions of service software, services' priorities, and services' responsible parties. The document lets you easily see who owns which service, which is necessary information in case you need to alert someone of a service outage. As an example, suppose you needed to immediately patch Apache because of a recently released exploit. You could consult Figure 10 to determine that no services rely on Apache, check Table 2 to determine who owns the service, and contact the E-commerce Division to schedule the upgrade.

**Table 2**  
*Service Layout Detail Document*

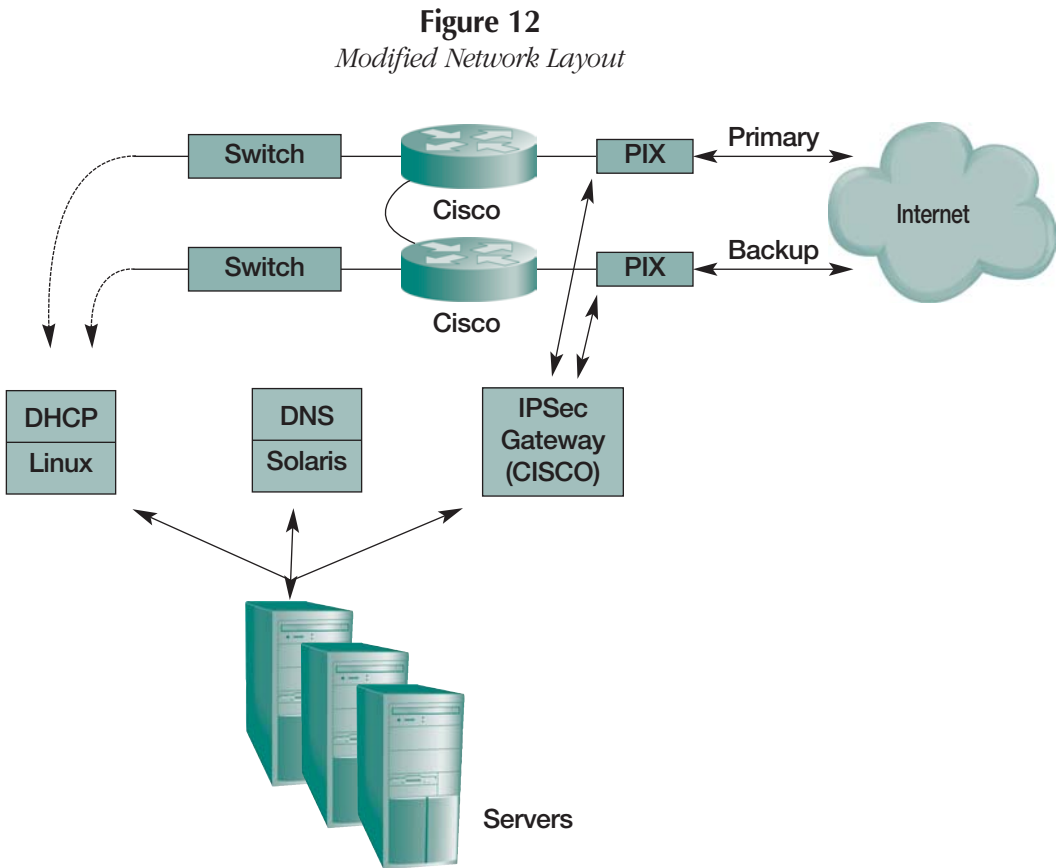
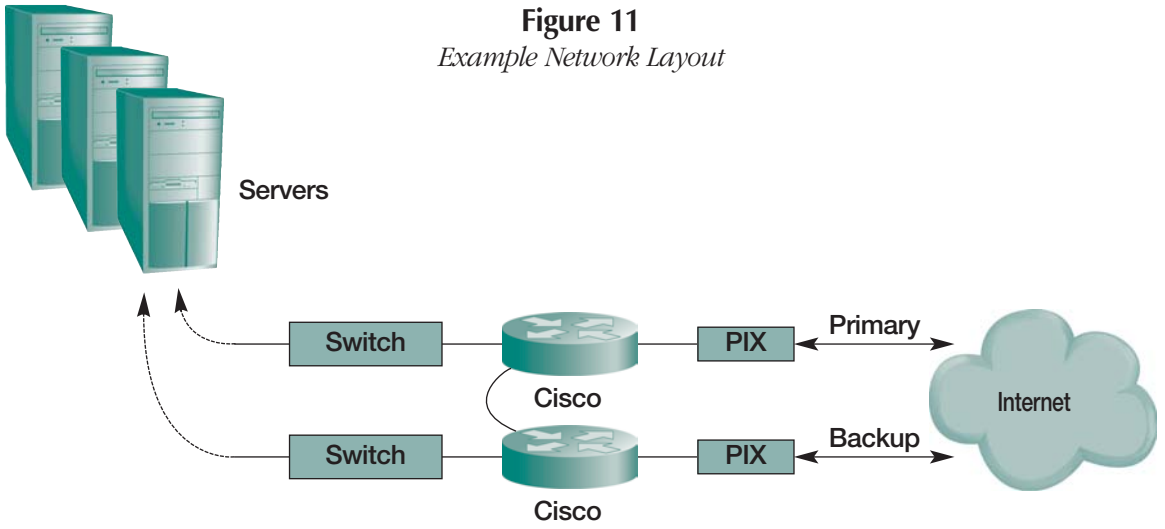
Server	Service	Version	Priority	Purpose	Owner
db1.example.com	Database	mysql-server-323-34	3-Critical	Stores customer data and orders	E-commerce Division
smtp.example.com	SMTP	apache-1.3.29	2-Important	Relays mail from Web servers to company email server	E-commerce Division
www1.example.com	HTTP	apache-1.3.29	3-Critical	Application server for e-commerce Web site	E-commerce Division
www2.example.com	HTTP	apache-1.3.29	3-Critical	Application server for e-commerce Web site	E-commerce Division

The systems manager chooses the format for a detail document. I recommend spreadsheets because of their built-in sorting and filtering capabilities, although a large organization might require a dedicated database application.

## Network Layout

Network Layout documentation is specific to network devices, physical and logical networks, and the integration of core services into the network. Like Installation, Configuration, and Recovery documentation and Service Layout documentation, Network Layout documentation is necessary in a well-documented and managed UNIX network.

Figure 11 shows a Network Layout diagram, which includes only the devices and networks in use on the network. This distinction separates Network Layout documents from Service Layout documents. A Service Layout document includes services such as DNS and DHCP, whereas a Network Layout document includes network devices and organization.



Although the diagram in Figure 11 is concise, it isn't as cohesive as a document of this type should be. Figure 12 shows a modified diagram that includes core network services; this diagram provides a clearer picture of the services and equipment the network is using. However, a disadvantage of this presentation is that you either must maintain two identical copies of services, one for the Network Layout and one for the Service Layout, or you must reference the Network Layout when you install services during a new or recovered server installation.

## Conclusion

This chapter covered a great deal of information. The topics I introduced here are merely a foundation for later chapters, which in turn provide even more in-depth information. All UNIX systems managers face similar problems in managing and maintaining their UNIX infrastructures. This ebook will help you discover the best practices and related software to ease your management burden and increase your environment's flexibility and resiliency.