

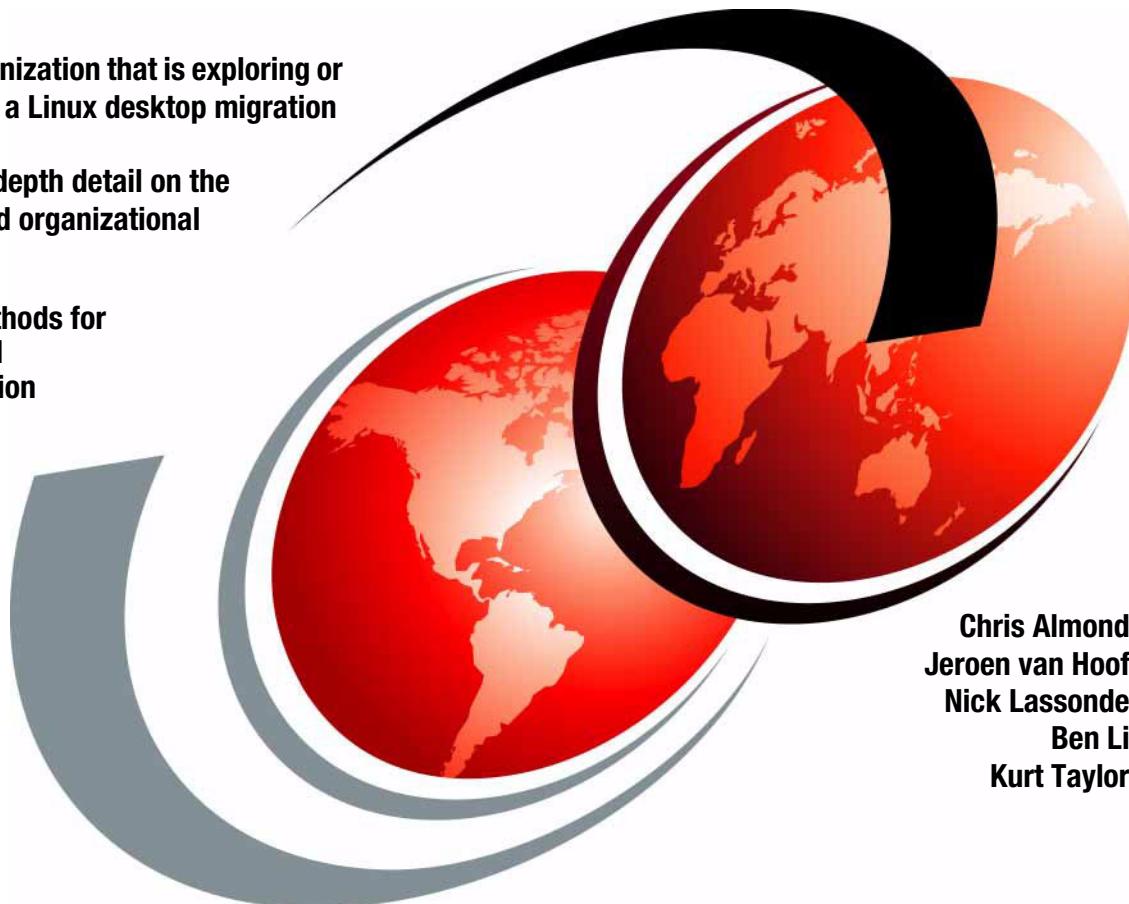
Linux Client Migration Cookbook, Version 2

A Practical Planning and Implementation Guide for Migrating to Desktop Linux

For any organization that is exploring or planning for a Linux desktop migration

Provides in-depth detail on the technical and organizational challenges

Includes methods for planning and implementation



Chris Almond
Jeroen van Hoof
Nick Lassonde
Ben Li
Kurt Taylor

Redbooks



International Technical Support Organization

**Linux Client Migration Cookbook, Version 2
A Practical Planning and Implementation
Guideline**

October 2006

Note: Before using this information and the product it supports, read the information in "Notices" on page xi.

Second Edition (October 2006)

© Copyright International Business Machines Corporation 2004, 2006. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Noticesxi
Trademarks	xii
Forward	xv
Bernard Golden, Navica	xv
Greg Kelleher, IBM	xvii
Preface	xix
The team that wrote this redbook	xx
Acknowledgements	xxi
Become a published author	xxii
Comments welcome	xxii
Part 1. Choosing Linux	1
Chapter 1. Introduction	1
1.1 The migration landscape today	2
1.2 Identifying suitable environments	2
1.3 Strategic context	3
1.4 Client environments	5
1.5 Why Linux	6
1.6 Linux overview and distribution choices	7
1.7 Desktop Linux futures	7
1.8 The rest of this book	23
Chapter 2. The case for migration	25
2.1 Why migrate	27
2.1.1 Desktop security	27
2.1.2 Costs related to Linux client	32
2.1.3 Manageability of the Linux client	34
2.1.4 Client customization	37
2.1.5 Free software and the open source philosophy	38
2.1.6 Ease of use and retraining	38
2.1.7 New economies of scale	38
2.2 When to migrate - Or not to migrate	39
2.2.1 Desktop Linux markets — the threshold of entry	39
2.2.2 Client roles fit thin and slim client model	40
2.2.3 High number of migratable applications	41
2.2.4 Organizational readiness	41

2.3 What makes Linux so different	42
2.3.1 The movements: free software and open source	42
2.4 Migration goals	43
2.4.1 Pilot migration	44
2.4.2 Full migration.	44
Part 2. Planning the pilot migration	47
 Chapter 3. Organizational and human factors planning	49
3.1 Assessing usage patterns	50
3.1.1 Functional segmentation - Fixed function to general office.	50
3.1.2 Surveying user data	52
3.1.3 User survey.	52
3.2 Establishing functional continuity	53
3.2.1 Bridging applications	53
3.2.2 Functionally equivalent utility applications	54
3.2.3 Web applications.	55
3.2.4 Building bridges to the server	55
3.3 Human factors	56
3.4 Retraining considerations	57
3.4.1 Bridging applications can separate retraining from migration	58
3.4.2 Learning a new look and feel	58
3.4.3 Familiar actions	58
3.4.4 File systems: Everything has been moved	58
3.4.5 Hands-on Linux prior to migration.	59
 Chapter 4. Technical planning	61
4.1 Assessing the client IT environment	63
4.1.1 Assessing the client hardware	63
4.1.2 Assessing the client software configuration	65
4.1.3 Assessing data dependencies	66
4.1.4 Assessing application equivalency	67
4.1.5 Assessing the infrastructure	67
4.1.6 Assessing the user	68
4.2 Integrating with existing network services.	69
4.2.1 Setting the environment	69
4.2.2 Authenticating within a Windows domain	70
4.2.3 File sharing using domain shares	72
4.2.4 Printing services in the domain.	73
4.2.5 DHCP and DNS configuration.	75
4.2.6 Web proxy interface	75
4.3 Standardizing the desktop.	75
4.3.1 Linux distributions	76
4.3.2 Linux desktop environments	76

4.3.3 Look and feel	79
4.3.4 User lockdown	83
4.3.5 Application choices	83
4.3.6 File systems and partitions	83
4.4 Migrating applications	84
4.4.1 Moving back to client/server computing	84
4.4.2 Logical segmentation - Thin, slim, or fat	85
4.5 Client administration planning	86
4.5.1 Operating system and vendor distribution updates	87
4.5.2 Application updates	88
4.5.3 Remote administration	88
4.5.4 Rollout of additional or replacement clients	89
4.5.5 Backup of clients	90
4.5.6 Virus mitigation	90
4.5.7 Examples of administration of enterprise distributions	91
4.6 Desktop versus notebook considerations	94
4.6.1 Hardware considerations	95
4.6.2 Peripheral extensions	96
4.6.3 Connectivity options	97
4.6.4 Offline mode	98
4.7 Unmigratable applications	99
4.7.1 What makes an application unmigratable	99
4.7.2 Terminal Server, Citrix Metaframe, or NoMachine NX solutions	100
4.7.3 Ericom Powerterm WebConnect for Workplace solution	101
4.7.4 VMware solutions	101
4.7.5 Dual boot solution	102
4.7.6 What to do if all else fails	102
4.8 Deploying the new client	103
4.8.1 Deployment method	103
4.8.2 Update deployed clients	104
4.8.3 Personalization of deployed clients	105
4.9 Post-migration troubleshooting and technical support	106
4.9.1 What to expect	106
4.9.2 How to handle the unexpected	107
4.9.3 When to contact vendor enterprise support	107
Chapter 5. Linux architecture and technical differences	109
5.1 What is Linux	110
5.1.1 Distributions	110
5.1.2 Standards	111
5.2 Technical differences	112
5.2.1 Kernel booting process	114
5.2.2 Communication, files, and services	115

5.2.3 Multi-user	117
5.2.4 Graphical and text-based environments	118
5.2.5 System runlevels	122
5.2.6 Drives, partitions, and file systems	123
5.2.7 Virtual memory	124
5.2.8 File links	125
5.2.9 Replaceable kernel	125
5.2.10 Device drivers and hardware support	126
5.2.11 Font support	126
5.2.12 64 bit and multi-core support	127
Part 3. Performing the pilot migration	129
 Chapter 6. Migration best practices	131
6.1 The transitional desktop	132
6.2 Choose an installation methodology	132
6.2.1 Wipe and Reload.	132
6.2.2 Dual boot.	132
6.2.3 Hardware refresh	133
6.2.4 Hardware round-robin	133
6.3 Centralize data locations	133
6.3.1 Central file server	133
6.3.2 Central mail server	134
6.4 Break down migration into manageable groups	134
6.5 Minimize impact of down time	135
6.6 Get user feedback	136
6.7 Automate the migration	136
6.8 Use a systems management tool	136
6.9 Do not migrate until you are ready	137
6.10 Do not just migrate, upgrade	138
 Chapter 7. Client deployment models	139
7.1 Restricting the desktop	140
7.1.1 KDE Kiosk framework	140
7.1.2 GNOME lockdown options	151
7.2 Remoting tools	154
7.2.1 Remote access	154
7.2.2 Thin client	155
7.2.3 Application forwarding.	155
7.2.4 Multi-station computing	155
7.3 Rich client	156
7.3.1 Eclipse and the Eclipse Rich Client Platform	157
7.3.2 IBM Workplace Client Technology	158
7.3.3 IBM Workplace Managed Client	158

7.4 Stateless client	160
7.4.1 Red Hat's Stateless Linux project	161
7.4.2 Custom implementation of stateless client	162
7.5 Multi-station client architecture	162
7.5.1 Multi-station computing and Useful Desktop Multiplier	163
7.5.2 What is multi-station computing	164
7.5.3 Approaches to desktop consolidation and deployment.....	166
7.5.4 Where and when to deploy multi-station systems.....	168
7.5.5 Advantages of deploying multi-station Linux systems.....	170
Chapter 8. Client migration scenario.....	173
8.1 Example client migration	174
8.1.1 Assess the client usage pattern	174
8.1.2 Identify the most important applications and infrastructure integration points	174
8.2 Migration plan details	175
8.2.1 Client approach.....	175
8.2.2 Graphical environment	175
8.2.3 Hardware.....	175
8.2.4 Application continuity	176
8.2.5 Windows networking	177
8.3 Performing the migration	178
8.3.1 Basic installation tasks	178
8.3.2 Integrating existing network services	179
8.3.3 Application configuration and installation	184
8.3.4 Screen captures: Client migrated to Linux	189
Chapter 9. Integration how-tos.....	195
9.1 How to join a Windows domain	196
9.1.1 Joining an NT4 domain	196
9.1.2 Joining an Active Directory domain.....	197
9.2 Using winbind to make domain users known locally	199
9.2.1 Common implementation of winbind	199
9.2.2 Alternate implementations of winbind	202
9.3 How to use LDAP to connect to Active Directory	204
9.4 Pluggable Authentication Modules and the domain	207
9.4.1 How to authenticate users using winbind and PAM	208
9.4.2 How to authenticate users using LDAP and PAM.....	209
9.4.3 PAM and home directories	212
9.5 How to mount a share on the Linux client	214
9.5.1 Mounting a share using smbfs	214
9.5.2 Mounting a share using CIFS	215
9.5.3 Use of smbclient	215

9.6 Automatically mounting home directories at logon	216
9.6.1 pam_mount on Red Hat Desktop	216
9.6.2 pam_mount on Novell Linux Desktop	218
9.7 How to use network printers in the domain	219
Part 4. Appendixes	231
Appendix A. Linux glossary for Windows users	233
What does it all mean.....	234
Common Linux Terms	234
Appendix B. Using enterprise management tools	255
Why use enterprise management tools	256
Internet standard technologies.....	256
Web-Based Enterprise Management (WBEM)	256
Simple Network Management Protocol (SNMP).....	257
Red Hat Satellite server and Red Hat Network (RHN).....	257
Architectural and functional overview	257
RHN Terminology	259
Sample update scenario	262
Novell ZENworks Linux Management	264
Architecture and Functionality.....	264
Usage examples	268
Webmin	269
Functionality	270
Usage examples.....	271
Other important tools	272
Appendix C. Automating desktop migration using Versora Progression Desktop	277
Benefits of an automated migration	278
What is Progression Desktop	278
How to migrate with Progression Desktop	279
GUI	280
Command line (with templates).....	282
Systems management tool	283
Progression Desktop architecture	284
PNP Files	284
Settings Packages	285
Plug-Ins.....	287
Enterprise Source License	287
Appendix D. Multi-station computing deep dive using Useful Desktop Multiplier	289

Deploying multi-station solutions on the IBM IntelliStation platform	290
Hardware requirements	290
Selecting your IBM IntelliStation model	296
Software requirements and installation considerations	296
Deployment considerations	298
Storage, printing, and external service considerations	298
Remote file systems	299
Network services	299
Network, electrical, and physical infrastructure considerations	299
Complementary technologies	302
Additional system management considerations	303
Software updates	303
Management tools	304
Software and support	304
System shutdown	304
Multi-language support	304
Security considerations	305
Privacy considerations	305
Case study one: General office desktops for a 25-user office	306
Requirement	306
Solution Design	307
Case study two: Transactional desktops: Public computers for a city library .	308
Requirement	308
Solution Design	308
Additional multi-station case studies	309
Understanding how multi-station computing works: Exploiting the flexibility of X	
Window System	309
The design of X	309
Appendix E. Client personalization	313
Microsoft Windows client personalization	314
Linux client personalization	314
Desktop personalization: KDE Desktop	315
Desktop personalization: GNOME Desktop	317
Appendix F. Desktop automation and scripting	321
Scripting languages	322
Shell scripting	322
Perl	322
Python	323
Embedded Scripting Languages	323
Kommander	325
Desktop interprocess communication	326

DCOP	327
Appendix G. Application porting	329
GTK+	330
Qt	330
REALBasic	330
wxWidgets	330
Mono and the .NET Framework	331
Related publications	333
IBM Redbooks	333
Other publications	333
Online resources	333
How to get IBM Redbooks	338
Help from IBM	338
Index	339

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®	Lotus Notes®	System z™
Redbooks (logo)  ™	Lotus®	Tivoli®
ibm.com®	Notes®	WebSphere®
AIX®	OS/2®	Workplace™
Domino®	Redbooks™	Workplace Client Technology™
DB2®	S/390®	Workplace Managed Client™
DFS™	Sametime®	1-2-3®
IntelliStation®	System i™	
IBM®	System p™	

The following terms are trademarks of other companies:

Java, JavaScript, JDK, JVM, NetBeans, Power Management, Sun, Sun Microsystems, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, ActiveX, Excel, Expression, Internet Explorer, Microsoft, MSN, Outlook, Visual Basic, Visual C#, Windows Media, Windows NT, Windows Server, Windows Vista, Windows, Win32, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Itanium, Pentium, Xeon, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

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

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

AMD and the AMD64 logo are trademarks of Advanced Micro Devices Corporation in the United States, other countries, or both.

SUSE and YaST are registered trademarks of SUSE AG.

AGFA is a trademark or registered trademark of Agfa Corporation, Agfa-Gevaert N.V., or Agfa-Gevaert AG depending on jurisdiction.

Nero and Nerovision are registered trademarks of Ahead Software AG.

Opera is a registered trademark of Opera AS.

Novell, iFolder, Mono, Nterprise, Red Carpet, Ximian, Novell Evolution, and ZENworks are trademarks or registered trademarks of Novell in the United States, other countries, or both.

Red Hat is a trademark of Red Hat Corporation in the United States, other countries, or both.

Tarantella is a trademark or registered trademark of Tarantella Corporation in the United States and other countries.

Debian is a registered trademark of Software in the Public Interest, Inc.

OpenLDAP is a registered trademark of the OpenLDAP Foundation, Inc.

Mandrake is a registered trademark of Mandrakesoft S. A. and Mandrakesoft Corporation.

REALbasic is a registered trademark of Real Software Corporation.

VMWare is a registered trademark of VMWare Corporation

KDE, K Desktop Environment, and Konqueror are registered trademarks of KDE e.V.

GNOME is a trademark of the GNOME Foundation.

Apple and Macintosh are trademarks of Apple Computer Corporation in the United States, other countries, or both.

Macromedia and Macromedia Flash are trademarks of Macromedia Corporation in the United States, other countries, or both.

Mozilla, Mozilla Firefox, and Mozilla Thunderbird is a trademark or registered trademark of The Mozilla Organization in the United States, other countries, or both.

Adaptec and Easy CD Creator are trademarks or registered trademarks of Adaptec Corporation in the United States, other countries, or both.

Adobe, Acrobat, Photoshop, FrameMaker, and Adobe Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Citrix and MetaFrame are either registered trademarks or trademarks of Citrix Systems Incorporated in the United States and/or other countries.

NoMachine and NoMachine NX are trademarks or registered trademarks of Medialogic S.p.A Corporation Italy.

SAP is a trademark or registered trademark of SAP AG in Germany and in several other countries.

RealPlayer is a registered trademark of RealNetworks Incorporated in the United States and/or other countries

WinZip is a registered trademark of Winzip Computing Incorporated.

Yahoo! is a registered trademark of Yahoo! Incorporated.

Big Brother is a trademark of Quest Software Corporation.

Nagios is a registered trademark of Ethan Galstad.

Netscape is a registered trademark of Netscape Communications Incorporated in the United States and/or other countries

Jasc and Jasc Software are registered trademarks of Jasc Software Corporation.

CUPS and Common UNIX Printing System are trademarks of Easy Software Products Co.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, or service names may be trademarks or service marks of others.

Forward

Bernard Golden, Navica

Open source software is a hotbed of innovation. Because the source code of a product is available for inspection and modification, users have the opportunity to change and extend the open source products they use.

Using the released version of a product as a foundation, open source developers can implement new functionality important to them and their organizations. Rather than being forced to live with the limitations of a product, open source users can improve it and make it better suited to their needs. They can even submit their code changes to the project managers for inclusion in the main code base of the product, ensuring that subsequent releases of the product will already include their changes, easing migration to newer versions of the product.

Because open source evolves so rapidly through user extension, it's critical for open source developers to adhere to standards for interoperability between applications. With the use of standards-based integration mechanisms, applications can evolve independently at their own pace, yet still be assured of being able to interoperate with other products.

A vibrant example of this open source characteristic can be seen in the Linux® desktop. Dozens of products are shipped in a Linux distribution, each progressing according to specific project goals and on project-specific timelines; however, products are able to interoperate based upon standards implementation.

The innovation and creativity of open source can be seen in the diversity of products. While Firefox is a very well-known open source browser released by the Mozilla Foundation, there are several other browsers included in a typical distribution, each focused on delivering an Internet experience in its own fashion.

This combination of rapid evolution, use of standards, and richness of alternatives is in direct contrast to the practices of the proprietary software world. In the proprietary world, vendors attempt to provide an integrated, sole source, single alternative software offering. Standards are often bypassed or only loosely implemented; since the vendor focuses on owning the entire stack; enabling other offerings conflicts with its business model.

While the convenience of an integrated, closed single offering can seem quite beneficial, it inevitably begins to fail under the stress of growth. As the vendor attempts to add more products to the integrated software stack, each of which must be integrated and delivered in a bundled offering, delivery dates begin to recede into the far distance, repeatedly delayed to allow each product in the

bundle to be completed. To draw an analogy, integrated offerings resemble a convoy of ships, with progress to market paced by the slowest vessel.

The benefits of open source development seem quite obvious in comparison to the slow-moving convoy that is integrated proprietary software.

However, the embarrassment of riches that is open source development can pose problems as well. The innovation and experimentation characteristic of open source leads to situations like the challenge of the current Linux desktop: two strong offerings — KDE and Gnome — are strongly represented in user usage. Each provides its own way of enabling software applications to interact with the desktop environment.

While users are well-served by being able to select which desktop best suits their needs, Independent Software Vendors (ISVs) are faced with a dilemma: support one or the other, or double invest to support both. In a world where the Linux desktop holds a small market share, this dilemma deters ISVs from supporting the Linux desktop at all. Clearly, if the Linux desktop is to achieve its potential, software applications are critical.

Based on this, the Portland Project¹ was born. The Portland Project represents an initiative by both the KDE and Gnome developer communities to present a common set of application interfaces — a de facto standard — that application writers can use to ensure that a single version of their application will run in both desktop environments.

The Portland Project provides a common set of tools that application writers can make use of to ensure application portability. Each of the desktops can continue to evolve, while adhering to a set of interfaces that enables application vendors to achieve low-cost portability. In this way, the Portland Project supports the open source characteristics of innovation and integration through standards.

Open source on the desktop faces a number of challenges, and the desktop development community is focused on working through them in a methodical way, with an end goal of enabling the Linux desktop to be a convenient, inexpensive, and easily-used computing environment.

— *Bernard Golden*
Chief Executive Officer, Navica

Bernard is a recognized authority and consultant on the use of open source software, particularly in enterprise settings. He is the author of "*Succeeding with Open Source*"², which is used as a course text in over a dozen university computer science and business programs throughout the world.

¹ <http://portland.freedesktop.org/wiki/>

² *Succeeding with Open Source*, by Bernard Golden (Addison-Wesley Press, 2005):
<http://www.navicasoft.com/pages/bkoverview.htm>

Greg Kelleher, IBM

The Open Desktop community is defined by the people involved. It is a world-wide community of individuals who share a deep passion for technology and frequently a great sense of humor as well. They are individuals who thrive on collaborating with their peers in the Open Source Software community, in accomplishing real progress toward creating and delivering technical innovations, and in freely sharing the value of those innovations with the rest of the world. Those shared innovations — coming from the highly cohesive efforts of the technology focused sub-groups within the Open Desktop community and their passion to create something new and highly flexible — are the source of the momentum that drives this migration book project, as well as every other Open Desktop project.

This second version of the Linux Client Migration Cookbook is intended to help the Open Desktop community continue to move forward — to enable more people to grow the community by leveraging the amazing work happening right now in Open Desktop development around the world. More than anything else, those individuals and sub-groups within the community — who are working diligently on printing, GUI design, power management, sound, multimedia, GNOME, KDE, x.org, the kernel, drivers, applications, internationalization, fonts, sound, accessibility, Laptop support, and so forth — are the heart and soul of the Open Desktop community. What exists is an Open Desktop meritocracy, a common passion to be creative, and a tremendously cohesive effort. An effort that is leading to a mature alternative client computing platform. There are no solo players or rock stars that I want to shout out to here. Instead, my deepest respect and thanks go out to all of the contributors within the Open Desktop community.

A special thanks goes to the hard core Linux technologists that authored both versions of this book, to the Linux Desktop Architects, the OSDL Desktop Linux working group, and to Chris Almond, the Project Leader and technical editor for this book, who leads by being open minded and creative.

— Greg Kelleher
*Senior Program Manager, WW Linux Client Strategy and Market Development
IBM® Corporate Linux & Open Source*

Preface

This second version of the Linux Client Migration Cookbook builds on the content strategy we started with in the first version. Although anyone interested in using Linux on the desktop could benefit from different portions of this book, our primary audience for this book is existing business IT environments that need to begin an evaluation of desktop Linux, or in a broader sense any organization whose strategy is to move toward greater adoption of open source software and open standards.

For this version, our goal was to complete an end-to-end refresh of existing content, and add as much new content as possible. Some key areas that we have focused on included the following:

- ▶ The future viability of desktop Linux strategies is starting to brighten considerably. A key reason for that is that the desktop developer communities have started collaborating on projects and standards in a way that is now achieving unprecedented levels of cooperation. We highlight that fact in the Forward section of this book, as well as a new section in Chapter 1: 1.7, “Desktop Linux futures” on page 7.
- ▶ We have added key sections to the book that focus on illustrating the differences between Linux and Windows®. See 2.3, “What makes Linux so different” on page 42, and this new chapter: Chapter 5, “Linux architecture and technical differences” on page 109. When coming from a Microsoft® Windows orientation, these sections will provide important technical background for readers in understanding why a strategy based on open software and open standards can yield so many new, different, and potentially more efficient ways to design, deploy, and manage client computing systems.
- ▶ Once you settle on an open client computing strategy you still have to deploy the new platform. Because of the extreme flexibility of the Linux software stack, IT organizations will have many options for how to design their deployment model. When migrating from Windows based environments, often the best way to approach this process will be to step back and reevaluate whether or not it makes sense to continue a strategy of providing users with dedicated high powered desktop computers, each running operating system images that by default have many more applications loaded than they actually need. We have focused on adding content that illustrates the flexibility you have in Linux for designing operating system images and deployment models for those images. The Multi-Station computing deep dive provided in section 7.5, “Multi-station client architecture” on page 162 and in Appendix D

provides a great example of how the flexibility of the desktop Linux software stack facilitates innovative deployment models that consolidate client computing systems.

- ▶ Finally, once you have settled on the design and configuration of your Linux based desktop operating system image, and you have designed and tested a cost-effective deployment model that optimizes hardware usage and projected management costs, then you still have to migrate your users to the new systems. One key challenge in minimizing the disruption to users is this: how do you efficiently capture and migrate the set of important desktop personalization data from existing Windows based clients to the target Linux based clients? Tools for automating migration of this data are now available from multiple vendors. We highlight the importance of using these tools in a medium-to-large enterprise, and provide a deep dive introduction to how you can use one of those tools, in Appendix C, “Automating desktop migration using Versora Progression Desktop” on page 277.

The team that wrote this redbook

Version 2 of this IBM Redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Chris Almond is a Project Leader and IT Architect based in the Austin, Texas ITSO center. He has a total of 15 years of IT industry experience, including the last five with IBM. His experience includes UNIX/Linux systems engineering, network engineering, Lotus® Domino®-based content management solutions, and WebSphere Portal-based solution design.

Jeroen van Hoof is a Senior IT Specialist with IBM Global Services in the Netherlands. He has over 15 years of experience with UNIX operating systems and over 10 years experience with Linux. Before joining IBM through PwC Consulting, he worked in high-performance computing at the Dutch National Supercomputing Centre as a parallelisation expert. His areas of expertise include UNIX/Linux, Windows, Virtualization, TCP/IP networking (routing, switching), parallelisation (MPI, PVM), firewall security, programming (Fortran, C, Pascal, Perl), SAP Basis consultancy, and Web technologies (HTTP, HTML, CGI). He is a Red Hat Certified Engineer, VMWare Certified Professional and SAP Certified Technology Consultant. He holds a PhD in Theoretical Physics from the University of Nijmegen.

Nick Lassonde is the Chief Software Architect at Versora, where he leads development activities on Progression Desktop, a leading automation tool for migrating personalization data between Windows and Linux desktop

environments. Prior to Versora, Nick was Senior Architect for Miramar Systems, where he focused primarily on development of tools for moving critical data and configurations between computer operating systems and for design and implementation of automated testing procedures. Prior to Miramar, Nick served as a Software Architect for Cadence Design Systems, where he focused on interoperability issues between Microsoft and Linux/UNIX® workstations.

Ben Li is the Vice-President of Innovation and Outreach at Userful Corporation, where he oversees development of integrated software and hardware solutions for multi-station Linux systems. His technical experience includes management roles in security and system administration on Linux, Windows and OS X operating systems in academic environments, developing and refining content management systems and workflows, as well as online and cross-media publishing. Ben is also a researcher for The Center for Innovation Studies, a Calgary-based research and advocacy group for the innovation and research commercialization community, and holds two degrees from the University of Calgary. As a consultant prior to joining Userful, Ben has developed a helpdesk notification system for a multi-national business services corporation, as well as a database-driven public relations tracking and reporting system for a leading Canadian energy company.

Kurt Taylor is a Senior Software Engineer at the IBM Linux Technology Center, located in Austin, Texas. Kurt is currently a Technical Team Leader for the Linux System Management Team. He has worked at IBM for 10 years, and has over 17 years of industry experience architecting, designing and developing distributed network and system management products. He is currently working with the OpenPegasus project and the OSDL Desktop Linux workgroup. Kurt graduated from the University of North Texas with a Bachelor of Science degree in Computer Science.

Acknowledgements

This is the second version of this IBM Redbook. The IBM Redbook team would like to acknowledge and thank the team of authors that wrote the first version of this book: **Chris Almond, Art Cannon, Jeroen van Hoof, Christian Patsch, Sekkar Vaddadi, Oliver Mark, and Thomas Schwaller**. Their efforts in the first version helped to create the original concept for this book, and laid down the structure and a significant amount of content that still appears in the second version.

In addition, the team would like to thank the following people for providing significant support for this project:

- ▶ **Greg Kelleher**, Senior Program Manager Worldwide Linux Desktop Strategy and Market Development: For sponsoring the project and helping to guide the development of the content outline and scope for this book.
- ▶ **Stephen Hochstettler**, ITSO Project and Linux Team Leader: For his vision and support.
- ▶ **John Bergland**, ITSO Project Leader, Lotus and WebSphere Portal, for his contributions.
- ▶ **Bernard Golden**, for his contributions.
- ▶ **Gerry Anderson**, **Jim Zemlin**, and **Dirk Hohndel**, Intel® Inc.
- ▶ **Mike Sheffey**, **Jon Walker**, Versora Inc., and **Timothy Griffin**, Userful Inc.
- ▶ **Gerry Riveros**, **Jonathan Blandford**, **Chris Blizzard**, **Dan Walsh**, **Seth Nickell**, and **Máirín Duffy**, Red Hat, Inc.
- ▶ IBM ITSO Professionals: **Erica Wazewski** (legal support), and **Leslie Parham** for her support in preparing the book for publication

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

<http://ibm.com/redbooks/residencies.html>

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review redbook form found at:

<http://www.redbooks.ibm.com/>

- ▶ Send your comments in an Internet note to:
redbook@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Bldg 905, 3D-004
11400 Burnet Road
Austin, Texas 78758-3493



Part 1

Choosing Linux

Part 1 of this book includes:

- ▶ Chapter 1, “Introduction” on page 1
- ▶ Chapter 2, “The case for migration” on page 25



Introduction

For several years now, many people involved with computing and the Internet have harbored hopes that Linux might become a viable user operating system choice for a broader segment of general purpose users. At the same time, there has been growing frustration with the problems and limitations of the current dominant commercial desktop operating system offerings from Microsoft. In turn, this frustration has fueled a greater need in the market for alternative desktop operating system choices. At the same time, Linux-based desktop-oriented distributions have improved tremendously as a result of the inclusive and open-ended dynamics of the open source development movement.

The goal of this book is to provide a technical planning reference for IT organizations large or small that are now considering a migration to Linux-based personal computers. For Linux, there is a tremendous amount of “how to” information available online that addresses specific and very technical operating system configuration issues, platform-specific installation methods, user interface customizations, and more. This redbook includes some technical “how to” as well, but the overall focus of the content in this book is to walk the reader through the important considerations and planning issues you could encounter during a migration project. Within the context of a pre-existing Microsoft Windows-based environment, we attempt to present a more holistic, end-to-end view of the technical challenges and methods necessary to complete a successful migration to Linux-based clients.

1.1 The migration landscape today

As this book was being written, the range of choices and capabilities of native Linux software was expanding rapidly. For those few nascent IT organizations that have the choice to grow an install base of Linux desktops organically (that is, they are starting a desktop IT infrastructure from scratch), then the Linux choice should provide a basis for all of their application needs today. But the majority of Linux desktop deployments will most certainly occur within the context of a migration. And, at an application's level, one of the most common and important migration challenges will likely be the feasibility of migrating users from the Microsoft Office productivity suite to a Linux-based equivalent.

Other high-profile migration challenges with heavy network infrastructure dependencies include messaging (Microsoft Outlook® does not run natively on Linux), and to a lesser extent interaction with enterprise directory and authentication services. To that end, this book includes sections that discuss and demonstrate migration methods for Linux client integration into an existing Active Directory® and Exchange-based network.

As for migration of office productivity suite applications, at this time we believe that the odds for migration success currently favor organizations or users that do not rely heavily on use of advanced functions in Microsoft Office, or customized applications that integrate with or extend Office. We believe that greater odds for success currently favor the “fixed function” or “technical/transactional” usage patterns, as defined in 2.2.1, “Desktop Linux markets — the threshold of entry” on page 39, and 3.1, “Assessing usage patterns” on page 50.

Note: In the near term, we still see successful Linux client migrations favoring “limited use” client scenarios (the fixed function, transactional, or technical user segments). As Linux-based office productivity suite applications (and the content migration tools supporting those applications) mature, we expect greater frequency and success for migration of advanced office users.

1.2 Identifying suitable environments

Some clients, such as large banking and insurance companies, public administrations, and the retail sector, are pushing toward an Open Source based-solution not only on servers, but also on their corporate desktops. As with all products, technologies, or solutions, a “one size fits all” approach to the open source desktop will not be feasible in all cases. Critical questions need to be asked:

- ▶ Is the client's employee population strictly dependent on a third-party application, plug-in, or devices that are only supported on Windows?

- ▶ Has the client intensively developed custom applications based on native Win32® APIs and programming environments, such as Visual Basic® or other Windows scripting languages?
- ▶ Is the client's entire employee population dependent on advanced Microsoft Office-based functions (for example, dependencies on complex macros)?

If the answer is yes to any of these questions, then a Linux-based solution might be a less-suitable alternative, or present a more complex (higher initial deployment cost) solution strategy that will require careful consideration.

1.3 Strategic context

From a migration point of view, Linux is only one piece in the puzzle. Clients are faced with the problem of simplifying and optimizing existing end-to-end IT infrastructures, including servers, databases, applications, networks, systems management processes, and clients. All of this has to be done while constantly maintaining focus on minimizing complexity, risk, and cost, while providing a stable and scalable foundation for growth and new solution deployment as business requirements dictate.

In business environments, a desktop computer is almost never an island. It needs to be integrated into a larger networked, services-oriented environment; it connects to various servers, storage devices, printers, and so on. It uses not only the operating system, but also a wide range of middleware products, application packages, and might even require custom application development. It has to be deployed, supported, and managed.

Note: For medium to large enterprises, you can expect that the business case for a migration cannot be entirely justified by the potential just for cost savings in operating system and application software licensing fees. In fact, in a client migration scenario, you should expect that the most important cost consideration that will come into play in medium to large enterprises will be the overall cost of *managing* the client computing infrastructure on an ongoing basis.

Be prepared to answer this question: “*How will migrating to desktop Linux affect our year-to-year overall cost associated with supporting and managing our client IT infrastructure?*”

The key to answering this question lies in understanding the following:

- ▶ How you can leverage the Linux software stack to create innovative, cost-effective deployment models.
- ▶ Recognition of the importance and value in using enterprise management tooling that is specifically designed to optimize efficient ongoing support of desktop Linux in the enterprise.
- ▶ Understand how migration strategies apply to the different functional segments, and how the threshold of entry for migrating each of those segments can be affected by both technical and business process dependencies.

There are many sections of this book that, collectively, will enable you to better understand how to create innovative, cost-effective desktop Linux deployment models. We focus on the importance of enterprise management tooling in 6.8, “Use a systems management tool” on page 136, and in Appendix B, “Using enterprise management tools” on page 255. You can find more information about functional segmentation strategies and threshold of entry analysis in 2.2.1, “Desktop Linux markets — the threshold of entry” on page 39, and 3.1.1, “Functional segmentation - Fixed function to general office” on page 50.

In this IBM Redbook, we take the approach that you will be migrating clients to Linux within an existing Microsoft Windows-based environment. In this IBM Redbook, we do not present an entire network migration strategy to Linux-based clients and servers. But, it is safe to assume that any Linux client migration strategy will likely be part of a larger strategic IT infrastructure migration plan, where Linux plays a role in supporting enterprise network and application services. To demonstrate this separation of concerns, in Figure 1-1 on page 5 we show an example infrastructure migration path for converting a Windows NT®-based infrastructure to Linux. This redbook mainly focuses on the portions of this migration path that are within the dotted line box in the figure.

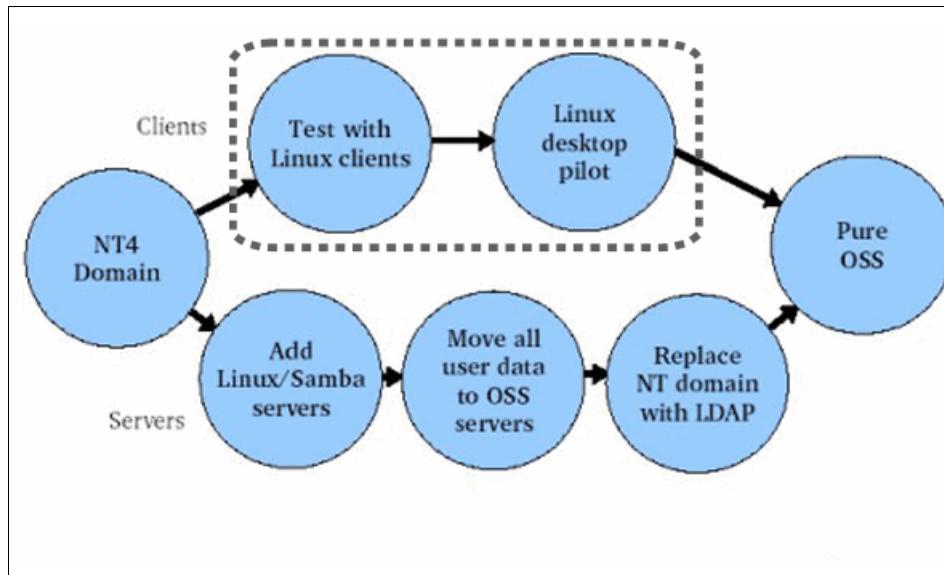


Figure 1-1 Migration route diagram¹

1.4 Client environments

The choice of an appropriate client platform for a particular set of users can depend on their functional role and the applications they must use to accomplish their objectives.

More and more line-of-business applications are being developed to depend less on the underlying operating system by taking advantage of open standards and pervasive technologies such as Web browsers. For economical reasons, many enterprises are quickly moving toward Service Oriented Architectures (SOAs) that allow them to compose applications out of existing services. This allows and encourages the reuse of application logic and data across the enterprise and even between enterprises.

SOAs are often implemented through Web services. Web services is an emerging set of standards that ensures interoperability by using such common technologies as Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Hypertext Transfer Protocol (HTTP), and others.

¹ Source "The IDA Open Source Migration Guidelines", netproject Ltd © European Communities 2003

Clients for applications based on Web services are often written in Java™ or based on Web browsers accessing portals. This makes the underlying operating system for clients transparent to the application and allows flexibility of choice, based on cost, support, flexibility, support for open standards, and so on.

It is important to understand what functions are required by a client platform to meet the needs of users today, while keeping an eye on the direction of technologies and enterprise architectures. This helps to ensure that the choices made today provide the capabilities that are required now and support the requirements of future architectures.

1.5 Why Linux

Linux has evolved into a powerful desktop operating system that can run on already existing hardware. In many cases, it requires less memory and processing power than other alternatives to provide similar performance on the client.

Because of its core design and open nature, Linux can be easily customized. Linux is available under the GNU General Public License² (GPL) agreement and can be obtained for free. However, most enterprises buy a Linux distribution to take advantage of the bundling features and support that accompanies them. The openness and flexibility of Linux, not the price, is becoming the driver for many organizations to migrate to this operating system. Its functionality, stability, scalability, and support have been key factors that have expanded the use of Linux from academic areas to the enterprise.

With support from such companies as IBM and others that deliver key client platforms, such as Lotus Notes®, the Mozilla Web browser, open office suites, and Java desktops, Linux is gaining momentum as a desktop operating platform.

Linux supports the Portable Operating System Interface (POSIX) standard that defines how a UNIX®-like system operates, specifying details such as system calls and interfaces. POSIX compliance has made it possible for developers to port many popular UNIX applications and utilities to Linux.

Linux also provides a complete implementation of the TCP/IP networking stack. A full range of clients and services are supported, including a standard socket programming interface so that programs that use TCP/IP can be easily ported to Linux.

² Copies of the GNU GPL Licenses can be found at <http://www.gnu.org/copyleft/gpl.html>
The GNU project is supported by the Free Software Foundation (FSF):
<http://www.gnu.org/fsf/fsf.html>

Linux supports the standard ISO-9660 file system for CD-ROMs, printing software, multi-media devices, and modems. In short, it provides the facilities to support the requirements of a wide range of client application types.

1.6 Linux overview and distribution choices

In 1984, the Free Software Foundation (FSF), started by Richard Stallman, began the GNU project to create a free version of the UNIX operating system. This system can be freely used, but even beyond that, the source code can be freely read, modified, and redistributed. A number of components were created, including compilers and text editors. However, it lacked a kernel. In 1991, Linus Torvalds began developing an operating system in a collaborative way. All information was made available for anyone on the Internet to improve the operating system that was called Linux. Linux was exactly the operating system kernel the FSF needed.

In the Linux community, different organizations have created different combinations of components built around the kernel and made them available as a bundle. These bundles are called *distributions*. Some of the most well-known distributions include Fedora and Red Hat Enterprise Linux, openSUSE and SUSE Linux Enterprise, Debian, and a derivative of Debian, Ubuntu Linux.

Linux is a UNIX-like, POSIX-compliant operating system distributed under the GNU software license. This means that the operating system can be distributed for free. Linux supports all the major window managers and all the Internet utilities, such as File Transfer Protocol (FTP), telnet, and Serial Line Internet Protocol (SLIP). It provides 32-bit and 64-bit multitasking, virtual memory, shared libraries, and TCP/IP networking. It is coupled to a native POSIX thread library for high-performance multithreading, symmetric multiprocessing (SMP) up to 16 logical CPUs or eight hyperthreaded CPU pairs, and massive parallel processing (MPP) up to 10000 AMD Opteron processors in a new Cray computer under development.

Linux has been developed to run on the x86, Itanium®, AMD64, and IBM System z™, System i™, and System p™ servers and S/390® architectures. A common source code base is used for all of them.

1.7 Desktop Linux futures

The future for desktop Linux looks promising. As this book was being prepared for publication, Novell released a ground-breaking version of their enterprise desktop Linux product: SUSE Enterprise Linux Desktop 10. At the same time, Red Hat is preparing for release of Version 5 of Red Hat Enterprise Linux, which

promises to include many of the cutting edge innovations from the Fedora community project in their desktop offering. Both Red Hat and Novell are providing desktop-oriented Linux products that demonstrate tremendous value, especially focused on business environments: from small business up to enterprise class. At the same time, the explosive popularity of the Ubuntu Linux distribution has aptly demonstrated the increasing interest in Linux as a viable desktop operating system alternative for the mainstream communities of casual home users and home computing enthusiasts.

Currently, one of the most important challenges facing the development communities that are focused on desktop Linux is compatibility of various application programming interfaces. For instance, how do you manage development of a single code stream for a popular multimedia player, while providing full functionality plus consistent look and feel when running in both the GNOME and KDE environments on Linux? This is possible today, but not without considerable effort on the part of the development teams. And the methods one team uses to accomplish this feat might not be at all similar to the same methods another team uses to reach the same goal.

A common set of standards is needed that allows developers to create applications which seamlessly integrate into various desktop Linux distributions, and the distributions need to support those standards. In support of this goal, there is a significant amount of community collaboration going on right now that will lead to further evolution and adoption of standards going forward.

To that end, the Open Source Development Labs (OSDL) sponsors a Desktop Linux Working Group. Also, freedesktop.org is actively supporting projects that are focused on interoperability and shared technologies between the various Linux desktop environments. More information about these projects can be found here:

- ▶ OSDL Desktop Linux Working Group:
http://www.osdl.org/lab_activities/desktop_linux
- ▶ freedesktop.org:
<http://freedesktop.org/wiki/Home>

Perhaps the single best point of reference for up-to-date information about Linux-focused standards development can be found at the Free Standards Group. Their tag line is: “Safeguarding the Future of Linux Through Standards”³. One of the key projects that the Free Standards Group supports is the Linux Standard Base (LSB). The LSB, and how it relates to supporting desktop Linux, is discussed in detail in the rest of this section.

³ About the Free Standards Group: <http://www.freestandards.org/en/About>

In the rest of this section, we reproduce the content of the following whitepaper:

“From Server to Desktop: Linux Standard Base Engages Application Developers Worldwide”^a

This is provided for publication in this book by Intel Corporation and the Free Standards Group. For more information, see:

<http://www.intel.com/softwareopensource>

<http://www.freestandards.org>

a. Reproduced with permission from Intel Corporation, and the Free Software Group.

From Server to Desktop: Linux Standard Base Engages Application Developers Worldwide

The opportunities and the promise of LSB as a stabilizing force to build greater acceptance for Linux at the enterprise level, particularly on the desktop, are substantial.

For many years, the open source development community has been engaged in the quest for a stable, unified, well-supported operating system (OS) — a reliable platform to provide the foundation for large-scale adoption at the enterprise level. While the Linux* OS has fulfilled many of the requirements of large-scale organizations as a cost-effective, reliable server OS, its adoption at the desktop level has been less common within enterprises. Part of the reason for this is the relative lack of full-featured desktop applications that meet the requirements of large organizations. The varied nature of non-standardized Linux OS distributions makes it difficult or impossible to achieve a global reach when developing applications for the desktop — the variations are too numerous and the user base is too fragmented.

The efforts of the Free Standards Group (FSG) and Intel, as well as other committed members of the ecosystem, promise to create a wealth of new opportunities for application developers. The fragmentation and disunity that inhibited the potential of UNIX* stands as a lesson to the industry. To avoid the mistakes of the past, many of those who were strongly involved with the evolution of UNIX are committed to achieving greater standardization and wider acceptance for Linux OS. A positive step in that direction has been the emergence of Linux Standard Base (LSB) as a worldwide standard ratified by the

International Standards Organization (ISO) and International Electrotechnical Commission (IEC). This landmark ratification also points toward the emergence of a thriving, competitive Linux OS distribution ecosystem that will encourage greater innovation and increased market opportunities. Both independent software vendors (ISVs) and users benefit from the expanded options and increased flexibility in the selection of operating systems for the desktop environment. Now is an excellent time for software companies and developers to become engaged in advancing and improving LSB and producing innovative applications for this platform. The Linux Desktop Project, initiated in 2005 and discussed later in this paper, focuses on standardizing the Linux desktop environment and establishing a commonality upon which developers can rely.

The current state of the Linux operating system desktop

The substantial costs of porting applications impose a burden on developers targeting applications for various operating systems. Particularly in the case of Linux OS, ISVs have been forced to grapple with multiple distributions of the operating system, often supporting multiple versions of the kernel, libraries, and applications to reach a worldwide audience. To expand the availability of applications for the Linux OS platform and minimize the development tasks in porting applications, standardization is vital. The Free Standards Group was founded to bring together industry participants and establish a working standard for the Linux OS that addresses binary portability and application development concerns. This purpose was achieved with the ISO/IEC ratification of LSB, which brings balance and order to the creation of desktop applications for Linux OS.

From a market perspective, the state of the Linux Desktop OS is relatively immature, but a number of indicators show that the arc of innovative development for the desktop is rapidly increasing.

The latest Linux OS desktop versions available today enable a very friendly user experience. Also, the addition of application suites, such as OpenOffice 2.0, provide a viable alternative to proprietary desktop office tools.

Jim Zemlin, executive director of the FSG, described LSB in simple terms as a unified set of application binary interfaces (ABIs), software libraries, and interoperability standards that will enable developers to target a single Linux operating system, avoiding the need to port applications for multiple distributions. “However, having said that,” Zemlin continued, let’s talk a bit about market adoption and the availability of a broader set of applications on the desktop. The value of a given environment is proportional to the network effect that is created by the total number of users within that desktop environment. In other words, the more people who can exchange information easily using a particular computing environment, the more valuable that computing environment is. ISVs want to

target a platform where, ideally, there will be millions and millions of users who will purchase and utilize their software.”

“Linux [OS] needs to create this kind of environment,” Zemlin said. “By using an open standard—such as Linux Standard Base—application vendors will not only be able to target the Linux desktop platform, but they will be able to target more than a single distribution—for example, Novell and Red Hat. They will be able to target a Linux Standard Base certified desktop distribution of their choice, thus creating an ecosystem of distribution vendors. It could be an Asian desktop, such as Red Flag in China, or Mandriva in South America, or Novell in Europe, or Red Hat in the U.S. You will get the kind of consistency required for mainstream uses in business, government, and education.”

Components of a Linux Standards Base

Built from a foundation of existing standards, LSB delineates the binary interface between an application and a run-time environment. Existing standards that LSB draws from include Single UNIX Specification (SUS), System V Interface Definition* (SVID), compilers for the Intel Itanium processor, C++ ABI, and System V Application Binary Interface* (ABI). At the same time, LSB builds on earlier efforts that attempted to prevent UNIX fragmentation, such as POSIX and SUS. In fact, it uses some POSIX source code standards and SUS interface definitions.

Although LSB has incorporated the durable aspects of these precursors, the FSG has learned from the UNIX experience, and, because of this, LSB has not adopted the limitations of POSIX and SUS. Notably, POSIX defined only programming interfaces and could not guarantee binary compatibility. At the other end of the spectrum, standards such as OSF/1, which aimed for binary compatibility, were found to be too restrictive. LSB strikes a balance between the two approaches—it includes a binary compatibility layer that splits the difference between the approaches taken with POSIX and OSF/1.

LSB formalizes the framework for interface availability within individual libraries and itemizes the data structures and constants associated with each interface. Figure 1-2 on page 12 illustrates the component organization in the LSB 3.1 environment. These components include shared libraries required by developers (including C++), file system hierarchies (defining where files are located in the system), specifications for the behavior of public interfaces, application packaging details, application behavior pre- and post-installation, and so on.

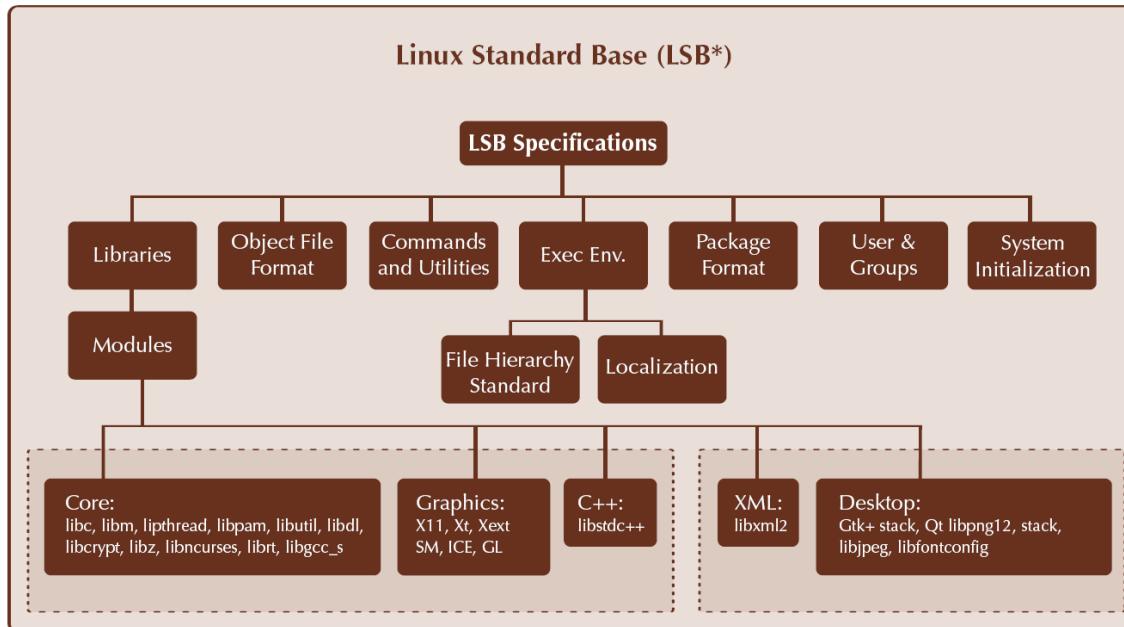


Figure 1-2 Component organization in the LSB 3.1 environment

The LSB Software Development Kit (LSB SDK), created and maintained by the LSB working group, consists of a build environment and the required tools to streamline porting software applications to conform with the LSB SDK. Tools within the LSB SDK also enable developers to validate the binaries and RPM packages to ensure LSB compliance and monitor the API usage by the application while the build is taking place so that conformance is assured.

Initial versions of LSB focused on server-side considerations—an approach that corresponded well with the early adoption of Linux OS at the server level within enterprises. The desktop initiative for LSB—a product of the Desktop Project for 2005 and 2006—completes the picture with specifications for application behaviors and libraries for Linux desktop applications.

The increasing availability of development tools to support efforts in this area is an essential component of building a unified Linux OS ecosystem and facilitating the development of compliant solution stacks.

Intel supports advances in this area with LSB-compliant compilers and other software development tools that contribute to stable, enterprise-caliber application development.

LSB 3.0 and the more desktop-oriented LSB 3.1 set the stage for LSB 4.0, which addresses many of the prominent issues that have proven problematic for developers targeting this platform. As discussed in the section titled Milestones on the Road to Large-scale Adoption, the goal is to allow developers to write an LSB-4.0-compliant desktop application that will be able to run across LSB-compliant distributions, integrating well with the two major desktop environments, GNOME* and KDE*.

More details on the progress of the LSB specifications can be found at:

<http://1sb.freestandards.org>

Strengthening the Linux Operating System Ecosystem

From an Intel standpoint, a standards-based Linux OS running on an open computing platform contributes to the overall health of the Linux OS ecosystem. Within a healthy Linux OS ecosystem, innovation and collaboration come together to provide fresh business models and solutions stacks — helping to extend the desktop market into areas of the world where demand for cost-effective PCs has been rising. Having clearly standardized interfaces creates more interest from ISVs to innovate and develop targeted solutions for the platform. In turn, this creates opportunities for hardware sales in areas of the market where inexpensive systems are required for large and small organizations.

Dirk Hohndel, director of Open Source Strategy in the Intel Software and Solutions Group, said, “Intel actively engages in Linux Standard Base development efforts to help meet the requirements of our customers who are asking for Linux [OS] solutions. We are working to provide a healthy and stable set of solutions for customers in different parts of the world. Obviously, whatever software stack our customers are choosing and for whatever reason, we want to make sure that Intel represents the platform of choice for them. Our primary goals are to respond to customer demand and contribute to the progress in developing a strong, international ecosystem.”

Incentives for independent software vendors

The nature of open-source development has created a climate in which new business models emerge around application development and distribution.

Within this open and flexible model, a new generation of solution stacks can combine the best characteristics of proprietary applications and free open-source software to challenge the marketplace, spur development, further design innovation, and extend market opportunities into new, previously untapped areas.

Difficulties in targeting the Linux Desktop Operating System

Ian Murdock, Chief Technology Officer of the Free Standards Group, has been involved in standards development since the evolution of UNIX in the early 1990s through the current effort to evangelize the benefits of LSB and bring coherence to Linux OS development. “Previously,” Murdock said, “developers would have to sit down and choose the distribution target— maybe Red Hat, Novell, and Debian—and then create individual ports for each of those distributions.”

Particularly on the desktop,” Murdock continued, “there is no standard way to do some pretty basic things—like create an icon on the desktop or interface with the printing subsystem. Just to do some of those basic things, you have to make assumptions about the environment that you are running in. And that includes not only which distribution you are running, in the desktop case, but also which desktop environment you are running. For example, there might be different ways to add an item to a menu, depending on whether the user is running GNOME or KDE. This is true even on the same version of the same distribution. The key issue is this: as a software developer writing an application for a platform, you want to pick the platform that gives you the greatest reach into your target market.”

“LSB essentially provides a single platform that an application developer can target. Applications will typically work across all distributions compliant with LSB. It is all about maximizing the addressable market.”

— Ian Murdock, Chief Technology Officer, Free Standards Group

In the current worldwide marketplace, distributions are often along geographical lines. What is popular on the desktop in the United States might be completely different than what is popular on the desktop in Asia. If the developer is targeting a specific geography, this makes it easier to choose the appropriate distribution. But it also immediately places limitations on the overall addressable market.

Prior to 3.1, LSB has been focused primarily on core components such as C and C++ run-time environments, areas typically associated with server applications. With the LSB core firmly established as an ISO standard, the LSB 3.1 release targets a level higher up the stack. LSB 3.1 starts at the lowest levels of desktop technologies, such as the popular graphical user interface toolkits GTK and Qt*.

Essentially,” Murdock said, “we have an ABI standard that covers both GTK and Qt and that allows application developers to write a desktop application against these toolkits that can run across LSB-3.1-compliant distributions.” LSB 4.0 further defines the desktop development environment, as discussed in the section Milestones on the Road to Large-scale Adoption, which explores the anticipated LSB roadmap.

Building success while expanding choice

As LSB provides a more stable, more consistent development platform with unified programming interfaces, the expanded market potential should strengthen the environment for commercial software sales. In all likelihood, developers will continue to take advantage of the agility and flexibility of open-source development to design new approaches to solving business challenges. With maturity and widespread adoption, LSB will enable software companies to experience enhanced opportunities and a broadened client base as usage of the platform expands.

“Proprietary software vendors are highly motivated to sell their software in the emerging Linux desktop market. They don’t care what platform it is — as long as there are plenty of users out there.”

— Jim Zemlin, Executive Director, Free Standards Group

“The key for the Linux [OS] desktop to succeed,” Zemlin said, “is to create a platform with consistent developer tools around a standard that enables many different system providers to distribute Linux desktop solutions—to be able to have applications run across all of those different distribution platforms.”

Dirk Hohndel, Director of Open Source Strategy in the Intel Software and Solutions Group, sees clear benefits for ISVs. “From the perspective of the ISV developer community—from the people who are trying to make a living with software that is running in this ecosystem—the LSB standard is tremendously helpful. It makes it so much easier to make the right decisions for your development environment, to make sure that the development you do (usually for one specific distribution) is easily accessible to customers who are running a different distribution. This means that it broadens your market without dramatically or exponentially increasing the effort on the development side.”

Any desktop platform by nature is more successful if it is localized and can be customized to a particular region or use case. When a developer customizes software, the most important consideration is to maintain enough consistency of standardization so that applications run on any of those particular regional or customized distributions.

The benefits of LSB are manifestly visible to ISVs, but the situation for distribution vendors is somewhat different. These vendors might be concerned that support for the standard could weaken their differentiation in the marketplace. However, despite the requirements of writing code to the new standard, most distribution members of the FSG see LSB as a way of furthering innovation, rather than stifling it. As Zemlin said, “A way to think of it is: innovation on top of shared invention. These companies are willing to standardize components of their open source invention in order to foster compatible innovation—on top of a shared platform.”

Ian Murdock also values the innovation sparked by the open-source community and believes that LSB will keep that alive while unifying the platform for commercial development efforts. “I think that the job of LSB,” Murdock said, “is almost akin to that of an integrator. We want to maximize the kind of Wild West mentality that prevails in the open-source community. The open-source community scales remarkably well because there is no central coordination or planning. We absolutely want to preserve that—that is what makes it tick. But, at the same time, we want to build this nice layer above it that makes the collective integrated whole on par with the commercial platforms. It is part of working with the stakeholders across the board—the upstreams, the distros, and the ISVs. And, it is also about doing a good job of building an abstraction layer above this Wild West ecosystem, so that it is more familiar to the commercial folks, in particular.”

While there is very little difficulty in targeting a single Linux OS desktop, the challenge increases exponentially if an application developer wants to support multiple Linux desktop distributions. For application vendors to stay competitive and effectively address the needs of their customers, they must target multiple platforms. In most cases, the time and cost associated with supporting multiple distributions is prohibitive. However, LSB and its unified ecosystem resolve this issue; developers can develop locally yet market to a large (and growing) international marketplace.

Reaching a global user base

A massive change has occurred in the software development model due to the Internet’s creation of a collaborative infrastructure. This new development model means commercial companies developing software need to proactively address the transformation and take advantage of the huge increase in the number of people using computer technology. Countries such as Brazil, India, and China are impacting the international market. Within these emerging markets, an increasing slate of new applications for Linux OS-based systems powered by Intel processors and technology can be found in numerous sectors—including government, education, and transactional applications, as well as small and mid-sized businesses. As the gap closes and the number of people with access

to computing technologies rises, there is great opportunity to sell both applications and hardware to a global market.

An industry call to action

“Our primary goals are to respond to the customer demand and contribute to the progress in developing a strong, international ecosystem.”

— Dirk Hohndel, Director of Open Source Strategy, Intel

Some of the areas where individual participants can work to advance the development of this ecosystem include:

LSB compliant operating systems:

Linux OS distribution vendors can help advance the acceptance of the standard by building LSB-compliant distributions and desktop tools. Compliance should be part of their standard development testing and QA at each release cycle. Most leading Linux distribution vendors have already achieved compliance. To achieve the goals discussed in this paper, all distribution vendors should participate in the LSB development process. The Client Linux Resource Center (www.intel.com/go/linux) provides solution-oriented information and resources about the operating system vendors participating with Intel.

LSB compliant applications:

ISVs can follow the LSB specification when developing applications for Linux OSs. They can also provide their feedback to the FSG to make sure their concerns are included in the next version of the standard.

User adoption:

Users now have a choice in their OSs and can include the selection of LSB-compliant systems in procurement policies and purchasing behavior. Large users of the technology can institute license and support agreements to include lifetime support, whether purchasing a Linux OS distribution or a Linux OS hardware system. This is something that is very easy for an user to accomplish and it effectively produces an insurance policy, so to speak, when purchasing this type of technology.

One extremely useful developer resource is freedesktop.org at:

<http://www.freedesktop.org>

Freedesktop.org is an open-source forum working toward greater interoperability and shared technology for X Window System desktops, including GNOME and KDE. This forum welcomes developers of all persuasions interested in furthering the graphical user interface technology associated with Linux OS and UNIX.

Growing the specification and corresponding tests for LSB provides extremely useful benefits to application developers and platform providers. This contributes to the growth of the overall marketplace and adds to the importance of LSB. Users and companies must also be educated so they can comply with and utilize the standard and develop solutions on top of it. As the number of libraries and interfaces included in the specification becomes more comprehensive and robust, the rate of adoption will grow and ultimately the market can thrive.

Milestones on the road to large-scale adoption

The approval of LSB by ISO, which took place on November 1, 2005 at the Open Source Business Conference, represented an important milestone clearly indicating the maturity and scope of LSB and the Linux OS. Ratified as a Publicly Available Specification (PAS) by ISO/IEC, the ISO standard will be published as International Standard 23360.

LSB Desktop Project

This ISO milestone dovetails with the October 18, 2005 announcement by the FSG of the formation of the LSB Desktop Project, which is supported by Intel and numerous other organizations. This project has already attracted an impressive contingent of industry participants including Adobe, IBM, HP, Linspire, Mandriva, Novell, RealNetworks, Trolltech, Red Hat, Xandros, and others. These companies are contributing time and resources in the effort to unify the common libraries and application behavior specified in LSB.

Once common application run-time and install time requirements are standardized and adopted by primary Linux OS distributions, developers will be freed from having to compile to multiple instances of Linux libraries and distributions. The LSB Desktop Project addresses the standardization of core pieces of the Linux OS desktop and provides clear guidelines for ISVs to follow in their development efforts.

Developers preparing for LSB 4.0 can begin now by building their applications for LSB 3.1. LSB 3.1 provides the foundational layer upon which LSB 4.0 will be built. As Ian Murdock said, "There is going to be some evolution beyond 3.1. For example, we can include additional functionality in some of the toolkits that is not in 3.1. Much of this depends on the ongoing discussion with the OSVs. In a sense, we are trying to capture what the OSVs want and need rather than

mandating what has to be done. On the desktop side, the big change is going to be integration of the standards produced by the freedesktop.org community, which will go a long way toward solving many of the issues involved in creating a common desktop framework.”

As a sub-project of the LSB, LSB Desktop Project follows the same modular framework that enables distribution vendors and ISVs to target the standard in a way that meets their requirements. This vision of a standard Linux OS has aligned a broad sector of industry leaders with the goal to build a robust, worldwide ecosystem that achieves a level of interoperability consistent with the requirements of users and ISVs.

The direction and focus of the Free Standards Group relies on active participation and collaboration among a deep and wide segment of industry participants including distribution vendors, ISVs, system vendors, open source developers, and independent software developers. Through ongoing development and clear standards that broaden the client base, the next generation of Linux OS-based desktop applications can be successfully created and released to a wide segment of both mature and developing markets.

Get more information about the activities of the Linux Desktop Project by visiting:

<http://lsb.freestandards.org>

Subscribe to the lsb-desktop list to stay engaged with others who are involved with shaping and refining the specifications for the Linux desktop at:

<http://lists.freestandards.org>

Discussion forums and the latest news about Linux desktop advances can be found at:

<http://www.desktoplinux.com>

Roadmap to LSB 4.0

Figure 1-3 on page 20 illustrates the unfolding roadmap for LSB. With LSB 3.0 now in place as an ISO standard and LSB 3.1 scheduled to be ratified in mid-2006, the focus is on application portability, as expressed by the LSB 4.0 enhancements now under discussion. These enhancements offer a practical solution to many of the desktop development issues discussed earlier in this paper — defining the interoperability framework, providing an update to the LSB core that includes GCC and other related items, and essentially establishing the unified development environment that will unlock opportunities for many ISVs. As shown in the Figure 1-3 on page 20, the estimated user base for Linux OS applications based on LSB 4.0 will be in the range of 75 million users.

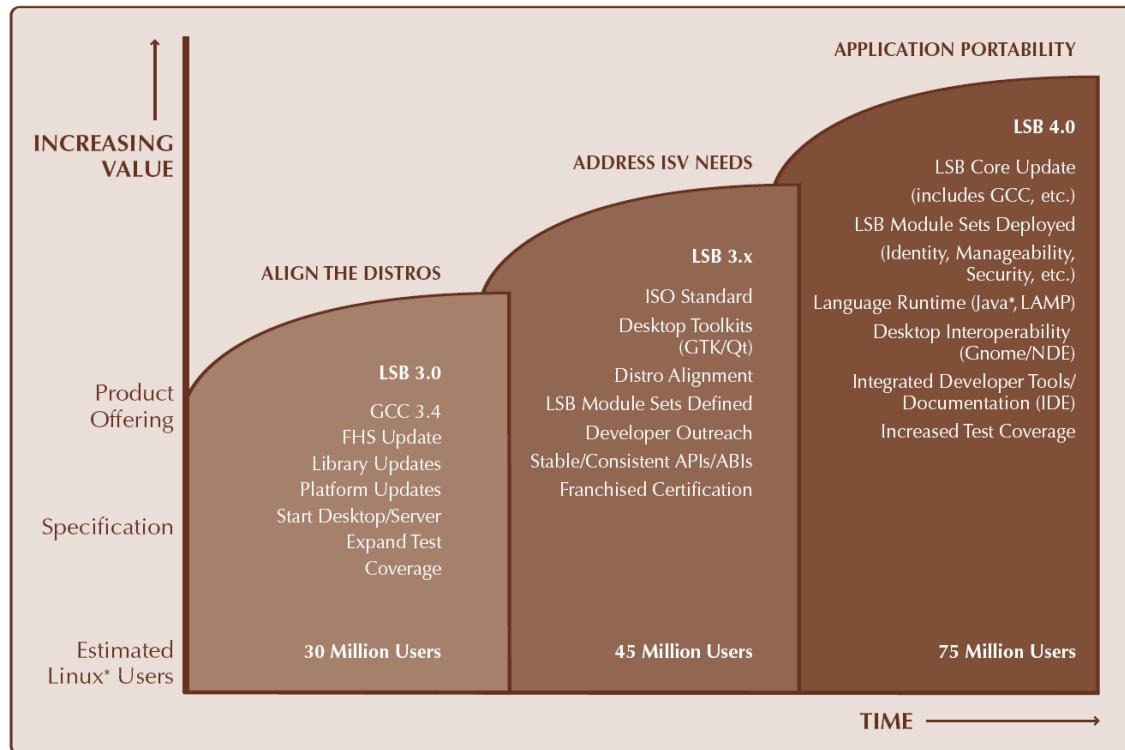


Figure 1-3 LSB Roadmap

LSB 3.1 Additions

LSB 3.1 includes the following:

- ▶ Integration of ISO standard LSB Core (ISO/IEC 23360).
- ▶ Readiness for GCC 4.1 and glibc 2.4. It should be possible for software built on LSB-compliant distributions based on GCC 4.1 or glibc 2.4 to build LSB-compliant applications.
- ▶ Addition of LSB Desktop, which initially covers the GUI toolkits (Gtk and Qt).
- ▶ Alignment of the LSB roadmap with the roadmaps of the major Linux distributions (Red Hat, Novell, Red Flag, Ubuntu, Mandriva, Xandros, and others). This will make it easier for software developers to correlate different versions of the LSB specification with the distributions that implement them. For example, targeting LSB 3.x provides support for Red Hat Enterprise Linux

5 and SLES 10, LSB 4.x provides support for Red Hat Enterprise Linux 6 and SLES 11, and so on.

- ▶ Greater participation of Linux distribution vendors and upstream maintainers in the LSB development process. This will make it easier to achieve roadmap synchronization and improve binary compatibility across major versions of the standard.
- ▶ Modularization of the LSB standard at both the compliance/certification level and the project management level. This will allow LSB to more rapidly incorporate emerging de facto standards with minimal disruption on the slower moving core functionality, as well as facilitate broader and more targeted participation in the LSB development process.
- ▶ Improved developer tools and documentation. This will broaden the Linux application market by making it easier for ISVs (particularly those coming from Microsoft Windows*) to target Linux.
- ▶ Franchised certification. This will allow third parties to integrate LSB certification into their own value-added certification and support offerings in a decoupled manner.

LSB 3.2 Additions

- ▶ Addition of freedesktop.org standards for cross-desktop interoperability (including menu entries, desktop icons, and so on), making LSB Desktop a complete desktop platform that allows ISVs to create applications that integrate effectively across GNOME and KDE.

LSB 4.0 Enhancements

The major additions and enhancements planned for LSB 4.0 include:

- ▶ Integration of updates to the compiler toolchain and core libraries that affect ABI compatibility (GCC, glibc, and so on).
- ▶ Binary compatibility with LSB 3. LSB 3 applications should run on LSB 4-certified distributions without being recompiled, as a step toward binary compatibility across major versions of the LSB standard.
- ▶ Addition of standardized language run times. Candidates are to be determined based on workgroup participation; the short list of desirable candidates includes Perl, Python, LAMP, and Java*.
- ▶ Addition of specialized LSB modules within the new modularized LSB framework. The actual candidates are to be determined based on workgroup participation; the short list of desirable candidates includes identity, manageability, multimedia, packaging, and security.

- ▶ Improvements in project infrastructure, including expanding the LSB database to track the major distributions in relation to LSB (useful for both internal project management and as an external developer resource), as well as continued improvements in the test frameworks.

Developers interested in learning the latest details about progress toward LSB 4.0 or who wish to participate in the discussion and definition of the standard should visit:

<http://www.freestandards.org>

This is a participatory, collaborative effort with room for diverse viewpoints and many opportunities to contribute to the crafting and improvement of LSB.

Summary

A standardized Linux operating system, as represented by the pending Linux OS desktop announcement in LSB 3.1, offers application developers a foundation for creating a new generation of desktop application and thus access to a share of a potential multi-billion dollar marketplace. Although standards can be a very esoteric part of the computing world, the ramifications of international standards have significant impact on global trade and affect many areas of the computer industry. For this reason, Intel and many other industry leaders are actively supporting the work of the Free Standards Group to refine and improve the Linux Standard Base to meet the requirements of the international community. The ultimate goal is the creation of a flourishing Linux OS ecosystem within which a new breed of desktop application can be developed to meet the needs of education, government, and business, and thus extend the benefits of reliable computing to the widest audience possible.

“Ten years from now, or maybe as little as five years from now, there will be a global ecosystem of Linux [OS] desktop solutions provided by profitable, competitive vendors based on this global standard. Our goals are to reduce computing costs, increase innovation, and lower support costs, as well as help produce an economy of human resources and human invention that will support this standard around the globe.”

— Jim Zemlin, Executive Director, Free Standards Group

The Free Standards Group is preparing to launch a Linux Developer Network to help developers acquire the skills and resources to port applications to Linux OS platforms. For current details and more information about the Free Standards Group, visit:

<http://www.freestandards.org>

For details about developing applications for standardized Linux OS, visit:

<http://www.intel.com/go/linux>

1.8 The rest of this book

The rest of this book consists of:

- ▶ Part 1, “Choosing Linux” on page 1
 - Chapter 1, “Introduction” on page 1

Introduction to the goals of this book, the strategic context, why you might want to migrate, and a deep dive on developer-oriented desktop Linux futures due to continuing development and adoption of the Linux Standard Base as well as other standards-oriented efforts.
 - Chapter 2, “The case for migration” on page 25

This chapter presents a detailed discussion about when and why you might want to migrate to Linux on the desktop. In addition, we introduce the topic of differences between Linux and Windows.
- ▶ Part 2, “Planning the pilot migration” on page 47
 - Chapter 3, “Organizational and human factors planning” on page 49

This chapter provides a discussion of human and organizational factors that are important considerations for transition management, and the importance of gathering non-technical data about the “as is” client environment.
 - Chapter 4, “Technical planning” on page 61

This chapter provides background, analysis, and planning guidance for various technical topics that need to be considered as part of planning for a Linux client migration.
 - Chapter 5, “Linux architecture and technical differences” on page 109

This chapter illustrates differences by presenting a top to bottom view of the desktop Linux software stack, the dynamics of the free software and open source movements, the definition of a “distribution”, and the importance of standards.
- ▶ Part 3, “Performing the pilot migration” on page 129
 - Chapter 6, “Migration best practices” on page 131

This chapter describes best practice methods you can use in your own Linux client migration projects. The topics covered include situations, which can make a migration to Linux easier, and use of third-party tools to automate specific migration tasks.

- Chapter 7, “Client deployment models” on page 139

This chapter concentrates on several client deployment models that can make desktop Linux clients easier to manage. We present general ideas that have been realized in several client projects, and several tools that are available as stand-alone open source projects, commercial products, or can be integrated in future versions of mainstream Linux distributions.

- Chapter 8, “Client migration scenario” on page 173

This chapter documents steps necessary to complete our own sample migration.

- Chapter 9, “Integration how-tos” on page 195

This chapter demonstrates additional methods for integrating Linux clients into an existing Windows domain (NT4 or Active Directory). We cover many integration issues, including mounting home directories from SMB shares at logon.

- ▶ Part 4, “Appendices” on page 231

- Appendix A, “Linux glossary for Windows users” on page 233

- Appendix B, “Using enterprise management tools” on page 255

- Appendix G, “Application porting” on page 329

- Appendix F, “Desktop automation and scripting” on page 321

- Appendix E, “Client personalization” on page 313

- Appendix C, “Automating desktop migration using Versora Progression Desktop” on page 277

- Appendix D, “Multi-station computing deep dive using Useful Desktop Multiplier” on page 289



The case for migration

This chapter presents a detailed discussion about when and why you might want to migration to Linux on the desktop. A lot of that discussion is triggered by understanding why Linux is different than Windows, and how those differences can justify a potentially costly migration process. So in addition, we introduce the topic of differences between Linux and Windows. Those differences are discussed at a high level here, and in much more detail later in the book (see Chapter 5, “Linux architecture and technical differences” on page 109).

We do not go into detail about total cost of ownership (TCO) or return on investment (ROI) models in this chapter. We also do not try to “sell” a migration; although, this chapter does present arguments as to why you might want to migrate to desktop Linux.

Since this book describes a migration to Linux clients from Windows clients, a comparison with the Windows platform is unavoidable. However, we primarily want to introduce the reasons for migrating on their own merits. A comparison with the Windows platform has been presented in various articles, papers, and reports many times, from various sources that are available on the Internet. And most of those sources tend to contradict each other anyway since they naturally reflect the bias of the authors writing them.

The sections in this chapter are:

- ▶ 2.1, “Why migrate” on page 27

Reasons for considering a migration.

- ▶ 2.2, “When to migrate - Or not to migrate” on page 39
Deciding factors about actually performing a migration.
- ▶ 2.3, “What makes Linux so different” on page 42
Understanding where Linux comes from and the dynamics of the open source and free software movements.
- ▶ 2.4, “Migration goals” on page 43
Pilot and full migration goals.

2.1 Why migrate

For many business environments today, Linux-based client computers can already provide a fully functional and cost-effective alternative to Microsoft Windows. The decision whether or not to migrate in your environment depends on many factors. It is important that you carefully consider all of the technical and organizational challenges before making any decisions. There are many situations where, on technical merits alone, a Linux-based client computing strategy is hard to beat. But in the broader analysis, a migration in those environments might still not be justifiable because of the high cost of overcoming organizational lock-in around an existing Windows-based environment.

In this section, we focus on some of the differentiating aspects of a Linux-based client computing strategy, presented as follows:

- ▶ Desktop security (2.1.1, “Desktop security” on page 27)
- ▶ Costs: direct & indirect (2.1.2, “Costs related to Linux client” on page 32)
- ▶ Manageability of the Linux client solution (2.1.3, “Manageability of the Linux client” on page 34)
- ▶ Client customization (2.1.4, “Client customization” on page 37)
- ▶ OSS philosophy (2.1.5, “Free software and the open source philosophy” on page 38)
- ▶ Ease of use (2.1.6, “Ease of use and retraining” on page 38)
- ▶ Economies of scale (2.1.7, “New economies of scale” on page 38)

2.1.1 Desktop security

You can easily write an entire book about the topic of comparing methods of securing Windows-based and Linux-based client systems. And, the technical landscape for comparing Windows with Linux is currently evolving rapidly:

- ▶ The upcoming release of Windows Vista™ will significantly change how the Windows platform manages many security-related concerns from a user point of view.
- ▶ The continual improvement and wider adoption of SELinux. All versions of Red Hat Enterprise Linux Version 5 and Red Hat Desktop Version 5 will include SELinux as an optional security services layer, thus providing the latest methods for multi-level security controls (MLS)¹ that are needed in many environments where highly sensitive data is managed. New “targeted policies” are now evolving that will allow for desktop-oriented security

¹ For more information, see: *The Path to Multi-Level Security in Red Hat Enterprise Linux*:
http://www.redhat.com/f/pdf/sec/path_to_msec.pdf

management policies that focus on a “compartmentalizing the breach” strategy. The basic idea in this strategy is to protect user space from non-user space processes. This strategy is supported by the fact that modern desktops are adding more and more “services” by default (that is, client applications that need network connectivity and awareness in relation to other client and server applications in the domain). It would be onerous to support application level compartmentalization for every desktop application needing it. So instead, because of the fact that desktop systems do not need to meet stringent availability requirements, you focus on a strategy that affords high levels of security (compartmentalization) to just specific file system partitions on the client that are used to store the most sensitive data. “In order to achieve this goal, the community and independent software vendors (ISVs) must work with the SELinux developers to produce the necessary policies.”²

- ▶ Novell’s AppArmor³ provides a Linux network security framework. AppArmor originally was developed by Immunix, Inc. Novell acquired Immunix in 2005, and AppArmor is now included in the SUSE Linux Enterprise 10 products. AppArmor is an open source project.⁴ The approach that AppArmor takes is different than SELinux, and they are currently seen as competing products. Novell maintains an extensive FAQ for AppArmor at this site:

http://developer.novell.com/wiki/index.php/Apparmor_FAQ

Because of its dominant share in the desktop computing market, coverage of Windows security concerns (the latest exploits, viruses, patches, and so on) receives a lot of attention in the IT press relative to those same sets of concerns on Linux. That does not mean that security concerns disappear when you migrate to Linux. But when you consider the open “UNIX-like” architecture of Linux and the fast response dynamics of open source community development, Linux does present a very compelling option when the goal is to lower the total cost involved in providing secure client computing platforms.

Even though Linux has many security-related features or properties, we concentrate on the ones relevant to a Linux client. Because on a Linux client there is interactive user activity, the security issues we look at reflect that. We focus on the features most related to this interactive user activity and some others that are also relevant:

- ▶ Browser security
- ▶ Messaging-client security
- ▶ User fencing and security
- ▶ Bugfix response time
- ▶ Modularity of the operating system
- ▶ Firewalling

² From the Fedora Core 5 SELinux FAQ: <http://fedora.redhat.com/docs/selinux-faq-fc5/>

³ <http://www.novell.com/linux/security/apparmor/overview.html>

⁴ <http://en.opensuse.org/AppArmor>

- ▶ Cultural differences in the developer communities
- ▶ System level access
- ▶ File system controls

Browser security

The browsers used with a Linux client are mostly open source software. Some of these browsers are also available for other operating systems. In a Linux client, the browser is not part of the operating system—it is a stand-alone application hosted by the operating system. For Linux platforms, there are a number of browsers to choose from. The most commonly used are:

- ▶ Mozilla and Mozilla Firefox
- ▶ Opera
- ▶ Konqueror

Because the browser is not closely built into the operating system, it is more difficult to use the browser as an entry point for circumventing operating system security. All of the browser software runs in user space and not in kernel space. This also means that the browser software can be installed as a non-root user.

Apart from the fact that security exploits of the browser do not immediately affect the operating system, the bug-fixing process for the most commonly used browsers is usually very fast, most notably within days (and sometimes hours) after discovery of the security problem. This “speed to fix” is a result of the active involvement of a large and growing community of open source developers, and again because only the application is being patched and tested to provide the fix, not the host operating system as well.

Another temporary advantage of open source browsers is their small market share relative to Microsoft Internet Explorer®, thus making them smaller targets for exploitation. This advantage would diminish in the long run as more clients begin using alternative open source browsers.

Messaging-client security

A messaging client in this context is defined as an application that is used to communicate messages with another user of another computer over a TCP/IP network, including the Internet. This includes both e-mail applications as well as instant messaging applications.

Like browser applications, messaging applications for Linux are open source software, and they are stand-alone applications that are hosted by the operating system. This means that all advantages listed in the section about browser security are valid for messaging applications as well.

The open source messaging client can generally handle more than one messaging protocol. This means that a choice of a messaging application for the Linux client can still provide some flexibility in the choice of which server-side messaging application or protocol is being used. The choice of client application can actually be influenced by the choice of server-side application. Thus, security considerations can be an influence when designing messaging system services in a Linux-based environment.

User fencing and security

User security has been an important part of UNIX operating systems from their early beginnings. Since Linux, like UNIX, is inherently a multi-user operating system, it is possible to use this core feature to separate different security roles on the client. Also, the fact that there is only one user with all administrative rights (the root user) by default helps in keeping the client secure.

The security options in Linux that can be applied to users and groups are typically applied to “fence off” the login environment of the individual user. The user does not, by default, have administrative access to the operating system on the client.

Bugfix response time

A large factor in responding to security exploits is the time to fix. The risk related to an exploit is directly related to the time the security “hole” is available to people who want to exploit it. Since most if not all components in a Linux client are open source software, fixes either come from the open source community or from an enterprise vendor that offers this kind of support. It has been shown that the time to fix security exploits in the Linux kernel is quite short.

Also, since the source is open, it is possible for an organization to develop a bugfix on its own, test it and deploy it internally, and even share the modified source code with the OSS development community for consideration.

Firewall support

The Linux kernel has firewall services built into the kernel. As of Version 2.4 of the kernel, this firewall is called *iptables*. Configuring iptables correctly enables the administrator to implement a network security policy on the Linux client.

Iptables firewalling can be used to secure the Linux client in the following areas:

- ▶ Protect against worm viruses.
- ▶ Prevent individual users from using protocols not allowed by policies.
- ▶ Secure against known exploits.

Community development dynamics

The cultural differences in the developer ecosystems between Windows and Linux is quite significant. First, the open nature of Linux and other services that run on Linux allow anyone (with the proper skills) to be able to perform a process or source code level security audit of the system whenever necessary. The size and vigilance of the developer community facilitates a “many eyes” approach that usually yields a very quick and peer-reviewed response to security problems.⁵ Also, with open source packages any problems could potentially be fixed by the same person who found the flaw. In contrast, if a security flaw is identified in a closed-source product, then you are dependent on the ability of the software vendor to provide a timely fix. In an open-source product, fixes could come from the official repository, or could be developed in-house if the service is extremely crucial and needs to be fixed immediately.

System level access

Another major cultural difference is permissions for a standard user. Due to backwards compatibility concerns, many Windows users run with Administrator level access. Many older applications require Administrator access to run properly, and most applications still require Administrator access to install. On Linux systems, running as the root user is discouraged. Some distributions even display warning boxes or special warning wallpaper for the root user, to demonstrate the security risk to an individual user. Instead, in many cases users only need to have administrator levels of access to their home directories. When users wish to perform system maintenance, historically they would start a console as a root user and perform the configuration from there. On modern distributions, most graphical system configuration tools prompt for the root user password, thus granting root access only to that configuration tool. This main difference means that even if a user were to launch some malware, then it would not have complete control of the machine. Because the malware would still be able to modify or delete personal settings, it is still a problem for users. However, by not compromising the entire operating system, an infected Linux system in this scenario could be significantly easier to recover compared to a Windows-based system.

File system controls

Considering file system security, both the Windows and Linux base OS support grant and revoke permissions to read, modify, and execute files and folders. On Windows NT-based operating systems with NTFS partitions, Windows uses Access Control Lists (ACLs) to define each user or group who has permission to any given file. Most Linux distributions default to a less granular method of permissions, which breaks down access levels into three groups: the owner of a

⁵ Linus's Law states: “give enough eyeballs, all bugs are shallow”. Coined by Eric S. Raymond. See: http://en.wikipedia.org/wiki/Linus%27s_law

file, the group of a file, and everyone else. Therefore, if you want to grant read and write permission to two given users but not to anyone else, then there must be a group with just those two users in it, and then the file can be associated with that group. While it is possible to create countless groups in order to provide granular file security, the option is not realistic. While most environments do not have such specific requirements, there is a solution if necessary. Many modern file systems now include support for ACLs, and support from other tools (such as backup tools and file system browsers like Konqueror and Nautilus) is starting to gain momentum.

2.1.2 Costs related to Linux client

The different cost factors related to the client are fairly general and independent of the operating system, but the relative impact and size of these cost factors can be highly dependent on the client operating system. Cost factors to consider for a Linux migration include:

- ▶ License and support cost for the Linux distribution
- ▶ Hardware cost
- ▶ Application cost for the base desktop client
- ▶ Management and support cost
- ▶ Migration cost

Note: In an actual situation in which you are presenting a justification model for a large scale migration to desktop Linux, you undoubtedly talk about overall cost to the company and the return on investment models that justify that cost. But within that situation, it is important to emphasize that you would not be talking about Linux on that day, in that context, if it were not for the game changing industry dynamics of the free software and open source movements.

Free software and open source are discussed later in this chapter (2.1.5, “Free software and the open source philosophy” on page 38), and in more detail in 2.3.1, “The movements: free software and open source” on page 42.

License and support cost for the Linux distribution

The Linux kernel and most applications included in Linux distributions are open source and licensed under the GNU Public License (GPL). This means that the software is freely distributable. Therefore, there are no license costs independently related to the Linux client.

However, distributions packaged by enterprise distributors are not free. There is usually a per-seat pricing model for enterprise distributions. And this fee usually includes a support mechanism for the installed machine for one year. It might

also be possible to purchase extra levels of support from the enterprise distribution vendors.

Given the open source nature of the Linux operating system and related software, it is also possible to use it completely free of license and support costs. For support, the organization is then completely dependent on the open source community and well-trained personnel within the organization.

Hardware cost

Most Linux distributions can run well on older hardware. Depending on your client application requirements, it is even possible to reuse hardware that has already been retired because it cannot support the latest Microsoft Windows performance requirements.

However, the newer enterprise distribution offerings for the desktop have minimum memory requirements that approach the same requirements of Microsoft Windows. Because this is a memory requirement only, you might still be able to use existing or even retired systems to support these distributions.

Application cost for the base desktop client

The Linux distributions include a large number of basic applications such as editors, imaging software, browsers, e-mail applications, instant messaging applications, and even some office productivity tools such as word processing, presentation, and spreadsheet applications. This means that the cost for these basic applications when using a Linux client is small to none.

Even if an application needed is not included in the distribution, chances are that there is an OSS equivalent that can be installed free of cost.

Management and support costs related to the client

In any medium to large enterprise, keeping any production desktop client operational and free of bugs and security exploits is usually one of the largest overall cost factors. This does not change in a Linux-based client strategy. But, the fact that the operating system is UNIX-like introduces a lot of innovative cost-saving strategies for consideration. For example, because the client can be remotely connected to and managed using telnet or SSH protocols, it is possible to install scripts on the client that can easily be remotely executed.

Using remote scripts, it is possible to monitor the clients for problems and to execute a task on all clients from a central server. For example, it is possible that a bugfix can be implemented on all clients by remotely executing a script on the client that is able to fetch a patch and install it, without user interruption.

A key consideration here is how the organization plans to use distributed system management tools. These management tool suites are available as additional add-on tools, or from enterprise Linux vendors, included as part of the operating system packaging. Examples are Red Hat Network from Red Hat and the ZENworks Suite from Novell.

Note: For medium to large enterprises, you can expect that the business case for a migration cannot be entirely justified by just a potential cost savings in operating system and application software licensing fees. In fact, in a client migration scenario, you should expect that the most important cost consideration that comes into play is the overall cost of *managing* the client computing infrastructure on an ongoing basis.

As the size of the supported client base scales up, these costs (IT staffing, support and help desk, system upgrade and replacement, repair and protection from viruses and other attacks, and so on) greatly overshadow the costs of recurring operating system and application software licensing.

This is why we strongly recommend that the approach for building a cost justification model for a client migration to Linux should emphasize the potential for introducing innovative ways to achieve savings in how enterprise client platforms are managed. The integrated management tooling suites provided by IBM strategic Linux business partners (Novell and their ZENWorks Suite; Red Hat and their Red Hat Network) provide important value-added features that any enterprise must consider implementing as part of a Linux client migration plan.

Enterprise management tooling is discussed further in Best Practices, 6.8, “Use a systems management tool” on page 136, and details about several tooling platforms are included in Appendix B, “Using enterprise management tools” on page 255.

2.1.3 Manageability of the Linux client

As stated in the previous section, one of the major considerations related to client choice is the way the client can be managed. Some of the inherent properties of the Linux operating system, as well as tools developed in either the open source community or by vendors, make the Linux client a very manageable client alternative.

The properties and tools we discuss in this section are:

- ▶ Modular structure of the operating system
- ▶ Update mechanism
- ▶ Inherent remote access

- ▶ Remote management and provisioning tools

Modular structure of the operating system

The Linux kernel has, by design, a modular structure. This means that the kernel is not one monolithic binary. Instead, it consists of a central smaller kernel binary together with various kernel modules. The kernel modules can be loaded when needed. Some of the modules have to be loaded at bootup, because these are needed to read file systems or other peripheral hardware.

Not only is the kernel modular, but the application framework around the kernel has a modular construction as well. The applications such as scripting engines (Perl, PHP, and Python) or editors (gedit, and vi) are not integrated into the operating system and can even be replaced by others or new versions more or less independently.

The modular nature of Linux means that updates or patches only involve a small part of the operating system, for example, either a single application binary or a library within the application framework or a single kernel module. This modularity is the main reason that an update or a patch almost always does not require a reboot on Linux.

Update mechanism

The update mechanism for Linux does not need to be central to the system. What we mean by this is that since updating or patching is not destructive to the system state (in other words, leading to a reboot), it can be done while other applications are running. Unless the update or patch impacts the core kernel or a running application, it can be done online.

This means that the mechanism that checks for new updates can be scheduled regularly and can be allowed to implement the changes automatically.

Inherent support for remote access

The Linux operating system has inherent remote access through TCP/IP, like all UNIX operating systems. This is a really powerful tool for systems management, because almost all management tasks can be done remotely, and since the advent of the secure socket layer can be done securely using SSH.

Remote access to the client is useful for two types of administrative tasks:

- ▶ Remote execution of administrative programs or monitoring scripts
- ▶ Access to the client from a central location to give support on the client

Remote execution

Remote execution is used to execute applications or scripts from a central location, for example, to force all clients to update a virus scanner or fetch a

critical security patch. This can also be used to execute monitoring scripts that enable the administrator to do preventive work, for example, to clear a temporary directory before a file system fills up.

Remote support

Almost all support on the client can be provided by accessing the client from a remote location. The only types of problems that still require hands-on interaction at the client are network problems. All other problems reported by the user can be handled by logging onto the client remotely and studying the system configuration and the log files. This capability could improve the effectiveness of helpdesk staff significantly.

Remote management and provisioning tools

There are a lot of tools available in the open source community as well as from commercial parties for remote management or monitoring and provisioning.

Some of the most popular tools are:

- ▶ Webmin⁶
- ▶ Big Brother/Big Sister⁷
- ▶ Nagios⁸

Webmin is mainly used for remote management of a single machine. Big Brother and Nagios are mostly used to monitor run-time properties of a network of machines, then to apply preventive administrative tasks when system properties go outside of established limits.

Commercial tools are available for these tasks as well. For example, IBM provides Tivoli® Configuration Manager and Tivoli Monitoring software for automation of monitoring and remote management tasks.

Red Hat Enterprise Linux distributions can use the Red Hat Network (RHN) to remotely manage individual systems or groups of systems. The RHN also offers automatic provisioning services as described in “Administration of Red Hat Desktop” on page 91.

Novell Linux Desktop and SUSE Linux distributions use ZENworks software to remotely manage or provision systems, as described in “Administration of Novell Linux Desktop” on page 93. After Novell acquired Ximian software, the SUSE Linux distributions adopted Ximian’s Red Carpet product (now under the ZENworks name) as the standard tool for their enterprise distributions. Before

⁶ <http://www.sourceforge.net/projects/webadmin>; <http://www.webmin.com>

⁷ <http://www.sourceforge.net/projects/big-brother>;

<http://www.sourceforge.net/projects/bigsister>; <http://www.bb4.org>

⁸ <http://sourceforge.net/projects/nagios>; <http://www.nagios.org/>

that, SUSE Linux used Yast online Update (YoU) as the tool to remotely manage systems.

2.1.4 Client customization

Sometimes it is useful to match the client to the role of the user using it. If a standard client build contains every possible application provided by the source distribution, then it becomes not only large but also more difficult to maintain. Therefore, client customization becomes an important goal in a Linux desktop deployment strategy.

In this section, we look at the following features of the Linux client:

- ▶ Flexibility to add or remove components of the Linux installation
- ▶ Ability to prevent “bloating” of the client by maintaining package control
- ▶ Availability of task-oriented distributions
- ▶ Flexibility of desktop design and session manager

Flexibility to add or remove components

Starting during the installation, it is fairly simple to add or remove components to the installation. Most installers supply a default installation, with the option to change which components to add or remove. This is again a result of the modular structure of the Linux operating system. This flexibility enables the construction of customized installations for the Linux client.

Ability to prevent bloating of the client

Using the flexibility to add or remove components, it becomes possible to prevent the client from “bloating”. Because customized Linux clients can be constructed for different user roles, it is not necessary to put all applications in one client image. This ensures that the client size does not grow out of control.

Linux distributions have been constructed as small as a single floppy disk (1.4 Mb). The enterprise distributions will install to several Gb in a default installation.

Availability of task-oriented distributions

Because of the freedom to include and remove components and the absolute freedom to add other applications from the open source community, it becomes possible to create special task-oriented distributions. It is possible to create a distribution (and a Linux client based on it) for audio/video work or for signal processing. These task-oriented distributions are useful when there are very specialized desktop client requirements in an organization.

Flexibility of desktop design and session manager

One of the most obvious differences between the design of a Microsoft Windows client and a Linux-based client is the freedom of choice you have in selecting desktop session managers and window managers in Linux. Also, most session managers (certainly KDE and GNOME) also provide extensive sets of theme options that can be used to further customize the look and feel of the desktop user interface. These themes are essentially a collection of icons, colors, window frames, backgrounds, and widgets, which combine to provide the selected look and feel.

2.1.5 Free software and the open source philosophy

For some organizations, the fact that Linux is open source software can be reason enough for justifying a migration. This is mainly because those organizations want to avoid vendor lock-in. Another reason that organizations might want to use open source software is that the source code is available and can be studied or adapted to their specific needs when necessary.

Note: Even if these factors are not influencing your migration decision, it is still very important that you understand the dynamics of the free software and open source software movements. The Linux alternative as it exists today and the rate of innovation occurring right now around the Linux platform would not be possible without the support of the developer communities. (See 2.3.1, “The movements: free software and open source” on page 42)

2.1.6 Ease of use and retraining

Modern Linux-based user interfaces (UI), such as those based on GNOME and KDE, provide the familiarity of a windowing desktop application environment as well as very sophisticated application development platforms. In a properly designed Linux-based desktop UI, you should easily be able to provide acceptable levels of both familiarity (in the mechanics of controlling the UI components and in application behavior) and ease of use. Further on in this book, we discuss key strategies for easing the “on-the-glass” user transition in a Windows to Linux migration scenario (for example, see 3.2.1, “Bridging applications” on page 53).

2.1.7 New economies of scale

A Linux-based desktop computing strategy introduces many new options for designing client deployment models, options that introduce new economies of scale in the design of client architectures. Thin client architectures, and the hybrid strategy aptly demonstrated by the “Multi-Station” approach pioneered by

Userful Corporation⁹, are excellent examples of new economies of scale in the design of client architecture. See 7.2, “Remoting tools” on page 154 and 7.5, “Multi-station client architecture” on page 162 for more details.

2.2 When to migrate - Or not to migrate

This book focuses on methods for migrating Microsoft Windows-based clients to Linux-based clients within a mainly Windows-based enterprise. But in general, the client migration is almost always part of a larger migration to open source software within the enterprise. This has to be taken into account when planning a client migration.

Even though the new Linux desktop might have properties (as indicated in 2.1, “Why migrate” on page 27) that are in favor of a migration, the total end result of the migration must have advantages as well. The total end result of the migration is not just a number of clients running Linux. Because of infrastructure and application environments, the architecture after the migration is usually more complex.

In the rest of this section, we look at several circumstances that favor a decision to migrate to a Linux client. These circumstances can mostly be described as an absence of a complicating factor. We describe some in detail.

2.2.1 Desktop Linux markets — the threshold of entry

The Open Source Development Labs (OSDL) is at:

<http://www.osdl.org>

OSDL has developed a useful chart that illustrates the relative threshold of entry levels for each type of user segmentation in an organization. For more information about user and role segmentation and how it can affect your migration strategy, see 3.1.1, “Functional segmentation - Fixed function to general office” on page 50. The threshold chart is shown in Figure 2-1 on page 40 below.

⁹ <http://www.userful.com>

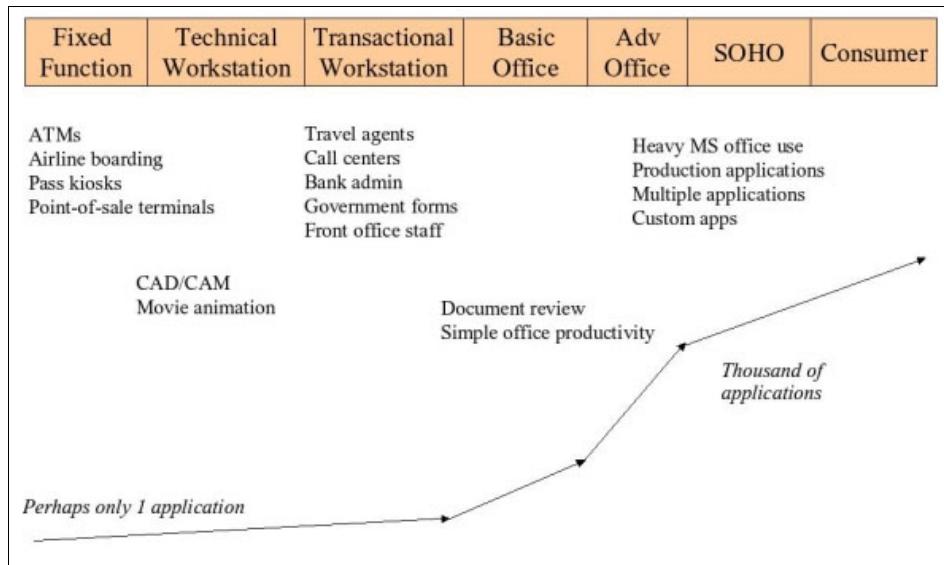


Figure 2-1 Desktop Linux markets — threshold of entry¹⁰

You can see from Figure 2-1 that, as your user segments move toward increasingly complex business application requirements, the threshold of entry increases.

2.2.2 Client roles fit thin and slim client model

Clients within a client/server model are generally called either fat, slim, or thin. This indicates where the majority of the application used actually runs. For example, in a thin client only the presentation part of the application runs locally (for example, in a browser), and the rest of the application runs on the server. These different roles are described in more detail in 4.4.2, “Logical segmentation - Thin, slim, or fat” on page 85.

One of the complicating factors in the migration is a large number of fat clients. A *fat client* is a client with a large number of applications locally installed. The larger the number of applications, then the greater the chance that some of those applications will be difficult to migrate to the Linux client. Some of them might even be unmigratable, which leads to a more complex infrastructure integration, as described in 4.7, “Unmigratable applications” on page 99.

If the current client is thin, this means that migrating the client to Linux does not involve the complicating factor of a complex infrastructure integration. In the

¹⁰ From http://www.osdl.org/lab_activities/desktop_linux (reproduced with permission)

idealized case of the user only using a browser to connect to a portal for all of their application needs, then migration becomes almost trivial.

The more the client fits a thin or slim client model, the easier it is to migrate to a Linux client. This leads to most early migrations only including the thin clients in an organization and leaving fat clients in their present state. It can also bring organizations to move to a thinner client prior to migrating to an alternative client operating system altogether.

2.2.3 High number of migratable applications

An application is termed *migratable* when there is a direct way to migrate the application to a Linux client, either by using a Linux version of the application or using a Linux-based alternative.

As described in more detail in 4.7, “Unmigratable applications” on page 99, applications that do not migrate well to Linux complicate the migration to a large extent. This also means that more migratable applications lead to easier migration.

Before performing a migration and handling the unmigratables, some organizations try to move away from applications that do not migrate easily. The best way to do this is to move the application to a portal-based application. This not only facilitates moving to a Linux client, but then the application also is not a problem in any future migrations.

2.2.4 Organizational readiness

Even if all other factors in the migration are favorable, an organization that is not ready for a migration will still be unsuccessful. An organization that is ready can be defined as:

- ▶ Having users that are ready and able to move to another client
- ▶ Having administrators and support staff that are enthusiastic about the migration and knowledgeable about the technology you are migrating to
- ▶ Having procedures in place that will streamline the handling of problems during and after the migration

These factors are discussed further in Chapter 3, “Organizational and human factors planning” on page 49.

One of the most important factors is that the migration has to be carried by the administrators. The people who have to manage the clients and infrastructure after the move can make or break the migration. Getting system administrators

and support staff behind the migration is one factor that can become very important in the decision process.

2.3 What makes Linux so different

We must begin any discussion about why Linux is so different by first understanding where it came from. Linux would not exist as it is today if it were not for the cultural dynamics of the free software and open source movements.

Note: We begin defining how Linux is different by discussing the cultural dynamics of the free software and open source movements here. We continue this discussion by illustrating top-to-bottom technical differences of the Linux desktop software stack in Chapter 5, “Linux architecture and technical differences” on page 109.

2.3.1 The movements: free software and open source

Two critical forces have shaped the development of Linux so far:

- ▶ The free software movement¹¹
- ▶ The open source movement¹²

These two forces have created the growth medium for the grass-roots community dynamics that have led to the development of Linux as it is today (and, to a large extent, also led to the graphical desktop environments and applications that you run on top of it).

In casual conversations, it is easy to lump the goals of the free software and the open source movements together. To an outside observer, the differences between them are subtle and rarely recognized. To members of the community, the differences are very important.¹³ The free software movement is a social movement. And “free software” in this case is not free by their definition unless the source code is available. In other words, free software implies open source. In contrast, the open source movement embodies a development methodology. The fact that a software product is “open source” does not necessarily mean it is free. So, “free software” implies open source, while defining a project as “open source” does not necessarily guarantee all of the freedoms that are promoted by the free software movement.

¹¹ Definition of free software: <http://gnu.open-mirror.com/philosophy/free-sw.html>

¹² Definition of open source: http://en.wikipedia.org/wiki/Open_source

¹³ For more about the relationship between free software and open source, see:

<http://www.gnu.org/philosophy/free-software-for-freedom.html>

Note: “Free” in this context has a different meaning than simply not having to pay for something. A common refrain in the development community is to define freedom in this context using the following phrase: “freedom as in speech, not as in beer”. Freedom in this sense means that you have been granted certain liberties in how you may use the product, not that it has a certain price.

An excellent essay is available online that explains in detail the relationship between the free software and open source movements:

Live and let License, by Joe Barr, in linuxworld.com, published 5/23/2001

<http://www.itworld.com/AppDev/350/LWD010523vcontrol4/>

The community development dynamics that have been guided by the goals of the free software and open source movements have led to top-to-bottom transparency in the Linux software stack: from low-level kernel all the way up to user productivity applications in the UI.

It is this characteristic, the potential for top-to-bottom transparency of the entire software stack, that should be viewed as the single most important differentiator between Linux and Windows. And note too that this characteristic does not derive from purely technical differences. It is rooted in the freedoms promoted by both the free software movement and the community development dynamics promoted by the open source movement.

The following is stated earlier in this book, but it is worth repeating again in this context. In an actual setting in which you are presenting a justification model for a large migration to desktop Linux, you undoubtedly talk about overall cost to the company and the return on investment models that justify that cost. But within that setting, it is important to emphasize that you would not be talking about Linux on that day, in that context, if it were not for the game changing industry dynamics of the free software and open source movements.

Any IT organization that makes a strategic investment in Linux is also making an investment in these movements. And the long-term viability of a Linux-based strategy is going to be affected by the continuing progress of these movements.

2.4 Migration goals

Not all migrations are the same. While eventually migrations will have a goal of replacing all desktop clients in an organization with Linux clients, most early migrations serve as pilot projects and only migrate a part of the desktops.

In this section, we discuss both goals in more detail. We indicate how the goals of both types of migration are different. Details for planning either a partial migration, pilot migration, or full migration are found in the rest of this book.

2.4.1 Pilot migration

The actual goal of the pilot migration is not that some client will be running the Linux operating system. The main goal of a pilot migration is answering the following question: How can we deploy across the organization with confidence?

A pilot migration has to target all types of usage models that will be included in an eventual full migration. All applications that are going to be considered in the migration have to be included in the pilot as well. This way, the pilot can be seen as an information-gathering exercise for the full migration.

The pilot migration should give answers to most of the following questions:

- ▶ Can all applications still be used after migrating?
- ▶ Are users able to use the infrastructure of the organization in the way they did before the migration?
- ▶ Are administrators still able to perform all their tasks after the migration?
- ▶ Do the new infrastructure components, such as terminal servers or consolidated Windows desktops, function as planned?

2.4.2 Full migration

In a full migration, the migration methods that were tested in the pilot phase are used to successfully complete migration of all targeted desktops to the new configuration.

Goals of a full migration could include the following:

- ▶ Increase the level of desktop client security.
- ▶ Improve manageability of desktop clients.
- ▶ Lower the overall total cost of ownership (TCO) of desktop clients.
- ▶ Decrease dependency on a single software vendor.
- ▶ Improve the lifecycle of client hardware.
- ▶ Comply with governmental regulations or strategies (for example, China has declared use of OSS as a strategic imperative).
- ▶ Extend usage of Linux from servers to desktop to leverage existing experience and skills.

- ▶ An excuse for change (for example, to clean up existing problems and start new with a standardized client implementation strategy).

Good planning is a major part of the migration. If a migration is started from a technical perspective, it is very easy to start with the technical challenges. That is certainly a part of a successful migration. But to complete a successful migration, management support on all levels is certainly just as important.

In the remainder of this book, we discuss how to plan a migration, both human factors and from a technical point of view. We then document an example of migration, including technical *how to* steps for solving some of the issues involved.



Part 2

Planning the pilot migration

Part 2 of this book includes:

- ▶ Chapter 3, “Organizational and human factors planning” on page 49
- ▶ Chapter 4, “Technical planning” on page 61
- ▶ Chapter 5, “Linux architecture and technical differences” on page 109



Organizational and human factors planning

This chapter provides useful information regarding non-technical issues that correlate with a client migration. In contrast to migrations in a data center, you have to consider many more human and organizational factors—justified by the fact that a migration affects the daily work of the users.

The sections in this chapter are:

- ▶ 3.1, “Assessing usage patterns” on page 50
 - Considerations and sample segmentation model for assessing usage patterns
- ▶ 3.2, “Establishing functional continuity” on page 53
 - Defines “functional continuity” and stresses the importance of taking advantage of this when possible
- ▶ 3.3, “Human factors” on page 56
 - Considerations for planning how to manage transitions and expectations
- ▶ 3.4, “Retraining considerations” on page 57
 - Methods to consider for minimizing the cost and time associated with retraining users

3.1 Assessing usage patterns

When planning a pilot migration, one of the most important steps is to perform an as-is analysis of your current IT environment that focuses on client application usage patterns. From this analysis, you should be able to derive a segmentation of role-based usage patterns. A very helpful tool for gathering this information is a user survey. Practical methods for performing this survey are provided in 4.1, “Assessing the client IT environment” on page 63.

3.1.1 Functional segmentation - Fixed function to general office

When considering the range of applications hosted by a client (that is, what type of work is done at a workstation), desktops in an enterprise or corporate environment can be roughly segmented into five distinct types, as shown in the top row of Figure 3-1.

Fixed Function	Technical Workstation	Transactional Workstation	Basic Office	General Office
Limited use of business applications	Applications which drive business processes			
Limited office productivity	Simple office productivity		Advanced office productivity	
No e-mail	Simple e-mail		Advanced e-mail	
No instant messaging	Instant messaging			
	Simple browser access to the intranet and portals			Advanced browser access to the Internet
	File/print, systems management, network access, host emulation			

Figure 3-1 Client functional segmentation

The types are:

- ▶ Fixed function

Users of these client machines run only a fixed and limited set of designated applications. Applications are customized for specific usage. For example, a kiosk or point-of-sale terminal.

- ▶ Technical Workstation

Users of these client machines work on industry-specific applications. It might require specific software packages, tailored to sector or problem domain, for example, engineering applications, such as CAD and CAM applications, or entertainment applications, such as movie animation.

- ▶ **Transactional Workstation**

A client designed to run form-based applications to support transaction processing. Often additional functions required include access to an intranet or defined Internet sites, and simple e-mail (though no attachments).

Examples of transactional clients include travel agency workstations, bank teller workstations, and front office workstations in insurance agencies.

- ▶ **Basic Office Workstation**

A client designed to run applications in support of a company's business processes. Support is required for ERP and CRM GUIs, intranet browsing, access to Internet sites, instant messaging, e-mail (with attachments), and the creation and viewing of simple documents (memos, letters, and spreadsheets) within the company only. The level of Windows interoperability required depends on the number of Windows clients deployed in the organization. The applications could create files in portable formats (for example, ODF, PDF, or HTML). Examples of basic office clients include a loan officer workstation or an office administrator in a small business.

- ▶ **General Office Workstation**

A client designed to run a broad suite of applications, including business process applications (ERP and CRM GUIs), complex compound document creation (such as word processing, presentation graphics, and desktop publishing), and collaboration (instant messaging, file sharing, workflow, and advanced calendaring). The level of Windows interoperability required depends on the number of Windows clients deployed in the organization and the number of external organizations that a user interacts with. Browsing intranet and Internet sites is required, with support for a broad range of multimedia (such as streaming audio and video or Macromedia Flash).

Examples of an advanced office client include workstations to support sales and technical professionals, finance planners, and executive assistants.

Planning tip: You should expect that the migration of clients that fit into the more advanced functional segments as defined in the right side of Figure 3-1 on page 50 requires more intensive application and data migration steps. In considering the overall cost of migration, it might be appropriate to identify and migrate only those workstation segments that fit into the left side of this figure. As native Linux applications for the desktop mature and become more readily available, migration of the more advanced client functional segments, as shown in the right side of Figure 3-1 on page 50, should become more common.

For more discussion on this topic, see 2.2.1, “Desktop Linux markets — the threshold of entry” on page 39. Also, see the discussion about “segmented perspectives” on the Open Source Development Laps “Desktop Linux” page:

http://www.osdl.org/lab_activities/desktop_linux

3.1.2 Surveying user data

Assuming that a group or role-based segment of users is identified for migration, then the next step is to collect data (a technical survey) that captures all aspects of the workstation usage profile. This profiling data should include all applications that are being used, all application dependencies (for example, database servers), and all required file types that are used, as well as client hardware data.

As part of this survey, you need to pay careful attention to evaluating hardware and peripheral device compatibility. Another topic that needs special attention is the range of file types that are used company-wide. In order to avoid the possibility of inaccessible files or content, it is reasonable to make a list of all types and then identify applications or migration paths that guarantee the usability on the new system.

3.1.3 User survey

The task of an user survey can take a lot of time, but it is also extremely important, because along with information about users’ application usage patterns, you can gain insight into the user’s point of view. This can help you discover how users use the systems and what is important to them.

Another important issue you can learn about by this survey is the existence of applications or other items that are not listed in the software catalog of a company. Sometimes users have built templates or databases on their own,

which are useful to them, or they have installed applications that are necessary for their work requirements.

3.2 Establishing functional continuity

This chapter includes methods that help to retain the functions of existing applications. After the migration, users in most cases will have to switch to different but functionally equivalent applications. In order to bridge this gap, which can result in a loss of productivity, it is useful to develop a strategy in which users get accustomed to the new applications.

3.2.1 Bridging applications

Some applications that run natively on Linux are also available natively for Windows. These applications provide an opportunity to minimize the transition effects and retraining requirements that are triggered by an operating system migration to Linux. Thus, it is possible to migrate to applications that will be supported on the Linux platform prior to actual migration of the operating system itself.

Important: The method described in this section can provide an extremely important way to help ease user interruption and frustration, as well as minimize the amount of application retraining necessary to complete the migration.

The benefit of such pre-migration changes is that the users are allowed to get accustomed to the new applications before the actual migration of the operating system is done.

After the new operating system is installed, the users will not experience any change at all as far the applications are concerned when making the switch with respect to those applications.

Table 3-1 on page 54 provides several examples of applications that you can use as “bridging” applications between Microsoft Windows and Linux. The Linux-based equivalents listed in the second column are examples of applications that provide versions, which can run natively on both Microsoft Windows *and* most major distributions of Linux. Thus, they allow application migration to occur prior to operating system migration.

Table 3-1 Example bridging applications

Application used in Windows	Bridging application
Microsoft Internet Explorer	Mozilla or Mozilla FireFox
Microsoft Outlook or Outlook Express	Mozilla, Mozilla Thunderbird, or Evolution
Microsoft Word	OpenOffice.org ^a Writer
Microsoft Excel®	OpenOffice.org Spreadsheet
Microsoft Powerpoint	OpenOffice.org Impress
Jasc Paint Shop Pro, Adobe Photoshop	The GIMP ^b
Messaging Client (MSN®, Yahoo, ICQ, AIM)	GAIM ^c
IBM Lotus Sametime® 7.5 ^{d,e}	(Native Windows and Linux support)
IBM Lotus Notes 7.x ^{d,f}	(Native Windows and Linux support)

a. <http://www.openoffice.org>

b. <http://www.gimp.org>

c. <http://gaim.sourceforge.net>

d. IBM Lotus Sametime and IBM Lotus Notes will begin offering native versions for both Windows and Linux beginning with these versions.

e. <http://www.ibm.com/software/sw-lotus/products/product3.nsf/wdocs/st75home>

f. <http://www.ibm.com/software/sw-lotus/products/product4.nsf/wdocs/linux>

3.2.2 Functionally equivalent utility applications

The functions provided by utility applications such as file system browsers, archivers, and viewers force the design of these tools to be more closely tied to the host operating system. They cannot be considered “bridging applications”, in the sense that we described for the applications listed in the previous section. One of the reasons Linux is considered to be approaching equivalency with Windows is the availability of many choices for utility applications. In many cases, these applications can in fact have more powerful feature sets than their equivalent utility applications used in Windows today.

Table 3-2 on page 55 provides examples of functionally equivalent utility applications available in Linux.

Table 3-2 Examples of equivalent utility applications in Linux

File managers (Windows Explorer)	Konqueror (KDE) Nautilus (GNOME)
Archivers (like WinZip)	KArchiver, Ark (KDE) FileRoller (GNOME)
Viewers (like Adobe Reader)	Konqueror, KView (KDE) Nautilus, Evince (GNOME) Adobe Reader for Linux
Multimedia players (like Windows Media® Player)	xmms, xine, RealPlayer, totem, amaroK, JuK (KDE) Rhythmbox, Banshee (GNOME)

3.2.3 Web applications

Unfortunately, it is not likely that you can find bridging applications (the ideal case) or functionally equivalent applications to meet every requirement. For instance, applications providing ERP or CRM are especially likely to have thick client implementations for which there are not cross-platform equivalents between Windows and Linux. Enterprise application vendors are responding to this not by developing separate thick client implementations for each operating system, but by focusing on the Web browser as the container for extending their client applications to alternative client platforms, such as Linux.

If a browser-based solution is not feasible, the approach of bridging applications to the new desktop by first transitioning to a cross-platform Web client interface cannot be used. In this case, you might have to migrate the application to a newer version that does support multi-platform clients, or you might have to consider switching to another software vendor that meets cross-platform client requirements.

3.2.4 Building bridges to the server

In the last three sections, we showed ways to make the changeover easier for the users by switching to ported applications or Web clients. Another possibility to bridge gaps regarding functionalities is moving the application from the client back to the server as the primary processing host.

Server-based computing (for example, terminal services-based remote hosting of client applications) can yield many advantages. An introduction to some methods for this are discussed in more detail in 7.2, “Remoting tools” on page 154. Regardless of which operating system is installed or which sort of client is in use, you should always be able to insure that the same application user interface is delivered in your terminal application. The only requirement is

the availability of client-based applications that facilitate interaction with the remotely hosted application on the server.

In the case of Windows Terminal Server or Citrix Metaframe, the requirements are met—clients are available for both Windows and Linux. Native Windows terminal servers can be accessed from Linux using rdesktop, an open source client for the Remote Display Protocol (RDP). The URL for this useful tool is:

<http://sourceforge.net/projects/rdesktop/>

Citrix delivers ICA clients for several operating systems with its MetaFrame Suite, including an ICA Client for Linux.

Using this technique, many applications can be moved to the server-based model. Your migration path in this case introduces the possibility of moving from a fat to a thin client. This approach is especially worth considering for clients that fit into the Fixed Function and Transactional Workstation roles as defined in Figure 3-1 on page 50.

A new approach to server-based computing is given by the developers of NoMachine NX, using a connection protocol that has been shown to reduce network traffic on both X and RDP sessions significantly. More information about the NX protocol and NoMachine can be found here:

http://en.wikipedia.org/wiki/NX_technology
<http://www.nomachine.com>

3.3 Human factors

Because a desktop client migration affects users in a very direct way, considering human factors in the transition management strategy is extremely important.

You can expect that a radical change in the desktop interface from what users are accustomed to will cause different kinds of reactions: From enthusiastic acceptance to outright rebellion. Therefore, it is important to keep the users informed about the new developments in a clear and understandable way. A communications plan is key.

Figure 3-2 on page 57 provides an example of a new technology acceptance curve in an organization.

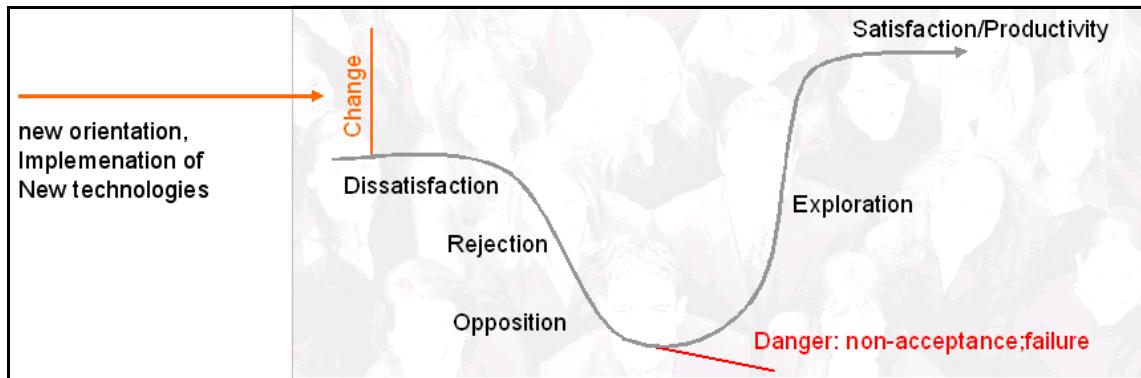


Figure 3-2 Impact of changes

The effect of a good communications plan is to flatten out the negative aspects of the acceptance curve in Figure 3-2. A communications plan coupled with proper training on the new desktop applications should minimize the number of users that fall into the rejection and opposition mode, cause users to start out with acceptance instead of dissatisfaction as the initial response, and lead to a quick transition into exploration and productive use.

Regarding the IT support staff, these same issues have even more importance. A strategic decision to switch the operating system and the way the clients are managed can cause the impression of disapproval of the work they have done so far. It could give staff the impression that their current skills are being devalued. It probably will not be easy to convince an administrator of Windows systems that migrating to Linux clients is a good strategy unless you can also convince them that the organization is prepared to make a significant investment in upgrading their skills as well.

Thus, two very important areas to focus on prior to starting a pilot migration are:

- ▶ Develop a communications plan.
- ▶ Select a pilot group that is best suited for the task.

3.4 Retraining considerations

In the course of the migration, retraining for users will be necessary in many cases. Because classes are always cost-intensive due to the payment for a trainer and the non-productive time of the employees, you have to figure out ways that can reduce this amount.

3.4.1 Bridging applications can separate retraining from migration

Referring to the bridging applications mentioned in 3.2.1, “Bridging applications” on page 53, the strategy of replacing current applications with OSS equivalents that are available on both Windows and Linux could reduce training costs. The migration could be done in a smoother way, because users have had the chance to acclimate themselves to the same applications that will be used on a Linux-based client, prior to actual migration.

3.4.2 Learning a new look and feel

Another strategy to save costs is the effort to retain the look and feel of the current applications and the desktop. It is possible to customize certain aspects of the GNOME and KDE desktops to emulate the look and feel of the Windows Desktop and Windows-based applications. Many freely available themes are available for download and further customization. For both KDE and Gnome, examples can be found at:

<http://www.kde-look.org>
<http://art.gnome.org>

3.4.3 Familiar actions

Emulating actions is also a good idea. A good example is enforcing double-click instead of single-click as the open action for desktop icons in the window manager.

3.4.4 File systems: Everything has been moved

Windows users are used to a hierarchical file system based on partition mount points such as C:\ or D:\. The hierarchical file systems in Linux differ from this convention. Some conventional mount points for file systems in linux include /usr, /home, and so forth. Migrated users could encounter much confusion when trying to understand the new Linux file system hierarchy. To smooth this transition, one recommended method is to migrate the entire contents of the user's existing My Documents folder into a similarly named folder in the user's Linux home directory. Inside of /home/user/My Documents, the content and structure will appear exactly the same as was present in the original folder in Windows.

3.4.5 Hands-on Linux prior to migration

Most Linux distribution vendors are now providing live or bootable CD-ROM-based distributions. One of the pioneers in creating live CD distributions was Knoppix:

<http://www.knoppix.com>

The following description from Knoppix further explains what a live CD provides:

KNOPPIX is a bootable CD with a collection off GNU/Linux software, automatic hardware detection, and support for many graphics cards, sound cards, SCSI and USB devices, and other peripherals. KNOPPIX can be used as a Linux demo, educational CD, rescue system, or adapted and used as a platform for commercial software product demos. It is not necessary to install anything on a hard disk. Due to on-the-fly decompression, the CD can have up to 2 GB of executable software installed on it.

A live CD can be used to provide users with the ability to run a bootable Linux system on their desktop. They can use it to test and evaluate the UI, applications, and other facets of the Linux client, before an actual client migration occurs. And this can be done without harming the host operating system install on the local hard disk. Another benefit of using a live CD as part of a migration plan is for detection of hardware and device problems. If you are able to customize the driver modules loaded by a live CD image, then you should be able to also help validate proper hardware detection and device support on your target platforms prior to actual client migration.



Technical planning

So far, we have focused on organizational planning considerations and strategy. In contrast to this higher-level view, in this chapter we take the discussion to a more technical level. What are some of the technical challenges that need to be considered and planned for when migrating from a Windows-based client to a Linux-based client?

Sections in this chapter include:

- ▶ **4.1, “Assessing the client IT environment” on page 63**

Before starting to look into technical details of a future Linux and Open Source-based environment, it is critical that you thoroughly assess the current status of IT infrastructure and the supporting processes.

- ▶ **4.2, “Integrating with existing network services” on page 69**

This section describes how to plan for integrating Linux clients in an existing network. We specifically look at how to incorporate Linux clients in a predominantly Microsoft Windows-based network.

- ▶ **4.3, “Standardizing the desktop” on page 75**

This section discusses the topics that should be considering when building a standard desktop client for an organization. We discuss distributions, the different desktop environments with their underlying toolkits, and also more general topics, such as corporate identity guidelines, look and feel, application menu design, user restrictions, and file system setup.

- ▶ **4.4, “Migrating applications” on page 84**
A migration path needs to be determined for each application that will be migrated to a Linux-based equivalent. Different migration paths will be necessary because not all Microsoft Windows-based applications have Linux-based equivalents. This section demonstrates methods for defining functional and logical client groupings based on what types of applications they run.
- ▶ **4.5, “Client administration planning” on page 86**
This section describes several methods for efficient administration of the Linux client.
- ▶ **4.6, “Desktop versus notebook considerations” on page 94**
This section describes additional considerations necessary for planning migration of mobile (notebook-based) clients.
- ▶ **4.7, “Unmigratable applications” on page 99**
This section covers issues related to applications that cannot be migrated to run directly on a Linux client. We propose alternate ways to support these applications.
- ▶ **4.9, “Post-migration troubleshooting and technical support” on page 106**
This section discusses considerations for supporting a Linux client after migration.

4.1 Assessing the client IT environment

Before starting to look into technical details of a future Linux and open source-based environment, it is critical that you thoroughly assess the current status of the IT infrastructure and the supporting processes.

For the IT Infrastructure, many components will come to play, not only the core installation of a desktop environment, but also any other infrastructure components around it. This might be servers providing file and print functions, databases, directory services, application servers (both native and Web-based) or even mainframes which are accessed via host emulation or API-based communication. Some of the key considerations and methods in this area are discussed in more detail in 4.2, “Integrating with existing network services” on page 69.

Because we are focusing on migrating client workstations, two key areas in the client IT environment need to be surveyed:

- ▶ Human interactions: The user application usage patterns
- ▶ Physical systems: Desktop system properties, special devices, and peripheral integration requirements

4.1.1 Assessing the client hardware

You should complete a physical systems survey of all client systems targeted for migration. The results of this survey will enable you to identify any hardware support issues, and define rules for buying and replacing systems in the future.

This survey should ideally be done through an automated process on a regular basis to develop historical data. This survey can also be done as a one-time scan for migration purposes. IBM eGatherer technology and the Asset Depot offering can help automate the survey. For more details, see the IBM Redpaper *Using Asset Depot for Inventory Management*:

<http://www.redbooks.ibm.com/abstracts/redp3763.html?Open>

Questions to ask in this area are:

- ▶ What hardware is in use (vendor, type, and model)? If this results in a large, diversified list, then consolidation is something to consider, regardless of the platform you are using.
- ▶ Is the hardware standardized? If all the machines are the same, then driver support and operating system deployment should be significantly simpler.

- ▶ What local attached devices are currently installed and required by users? This includes any type of printers, scanners, and special function keyboards required for business types of transactions.
- ▶ Is support for Linux included in current requests or proposals to hardware vendors when buying hardware?
- ▶ Which key components of your hardware are currently required by users? For example, machines might have sound cards installed, but drivers are not installed because users are not supposed to use sound. Therefore, sound support on Linux would not be required.
- ▶ What types of attached devices need to be supported? For instance, synchronizing calendaring and other data with PDAs or smartphones might be required. Also, USB sticks, Bluetooth devices, and external FireWire hard drives have become very popular, though some organizations disallow such devices due to security reasons. It might be necessary to support such hardware or to explicitly *not* include support for some of these devices. Likewise, many users might be synchronizing their desktops with an MP3 player. While supporting these MP3 players in Linux could help users enjoy the Linux experience, it can also increase the complexity of supporting a Linux desktop installation.

Sometimes hardware providers provide explicit support for Linux, while other devices are maintained by a third party. IBM, for example, provides information about supported desktop hardware for their current models, so customers can check on both officially supported hardware as well as available drivers.

Information can be found at:

<http://www-306.ibm.com/pc/support/site.wss/MIGR-48NT8D.html>

A sample client hardware survey table might look like Table 4-1 on page 65.

Table 4-1 Sample hardware survey results

Model	Type	RAM in GB	Disk in GB	Total	Attachments
2373MU4	T40	2	80	120	
23668AU	T30	1	60	42	
2366MU1	T30	1	48	1023	
2366MU9	T30	0.5	48	311	
26473U5	T23	1	48	278	
26476RU	T23	0.5	30	67	
26478EU	T22	1	30	44	
264757U	T21	0.5	24	1	
2682MU3	R40	1	48	99	

4.1.2 Assessing the client software configuration

You need to complete a client software configuration survey of all client systems targeted for migration. The results of this survey will enable you to identify all applications, services, special configurations, and component support issues that need to be considered in the migration plan. This survey can shed light on specific technical support issues, such as hidden user IDs or scheduled tasks.

Key survey questions:

- ▶ What third-party applications are installed and used? This will result in a list of ISV applications including versions used and potential fixes applied.
- ▶ What in-house applications are installed and used? This will result in a list of applications developed and maintained within the company that might need to be ported to Linux or a platform-neutral environment.
- ▶ What applications require access to data external to the client? This will result in a list of applications accessing file servers, application servers, Web servers, databases, mainframes, and other implementations of data processing. See 4.4, “Migrating applications” on page 84, for more details.
- ▶ Are groups of users defined? How are they characterized? This should give you an overview as to whether there are some typical groups or users and, if so, how they are grouped. Grouping can be done by departments, applications used, type of work, or business responsibility. If a questionnaire is used when interacting with users, then this should become a kind of self-assessment.

- ▶ What security-related applications, processes, and regulations are being enforced? This gives you an overview of products used for securing the client, such as anti-virus, desktop safeguarding, port scanning, as well as rules and regulations of how those applications are installed, maintained, updated, and how the user is forced to use them. It also includes policies for security patches of any operating system components and installed applications.

4.1.3 Assessing data dependencies

For most client/server applications, the only requirement is the availability of a functional replacement client-side application that runs natively on Linux. An example could be an application which uses a Web interface to access data stored on the server. As long as the Web interface can be run in a Linux-based browser, then the client-side migration of that application becomes a non-issue.

For some applications (mainly local and native applications), data can be stored in a proprietary format that will require a data conversion process. Applications in this category can include mail systems (such as Lotus Notes) or productivity suites (such as Lotus Smartsuite or Microsoft Office). Without discussing the actual technical migration, during the assessment you should explore the current use of such applications.

Some example questions to ask in this area are:

- ▶ Do you use Microsoft Office? If so, which components and how often?
- ▶ Do you use macros in Microsoft Office? If so, what type of macros and for which components and how often?
- ▶ Do you use Microsoft Outlook? If so, which components and how often?
- ▶ Do you use Microsoft Project? If so, which components and how often?
- ▶ Do you use Visual Basic to automate activities within or across applications?
- ▶ Do you use Lotus Smartsuite? If so, which components and how often?
- ▶ Do you use Lotus Notes? If so, which components and how often?
- ▶ Do you share files with external organizations? If so, which formats and how often?

Some answers to the above questions will require further reviewing of the underlying infrastructure. For instance, using Microsoft Outlook on the client most often leads into Microsoft Exchange on the server, while Lotus Notes on the client usually indicates Lotus Domino on the server. For scenarios such as these, server installations have to be taken into consideration when designing a new client stack, and migration of user accounts has to be planned. If there is no Linux client for the back-end server, the back-end server might need to be

migrated to a Linux-compatible solution before the Linux client migration can begin.

4.1.4 Assessing application equivalency

We expect most, but not all applications, to have functional equivalents available that run natively on the Linux client. And, there are special (ideal) cases where the functional equivalents are also what we call “bridging applications” (discussed in 3.2.1, “Bridging applications” on page 53). In general, once you have assembled a list of all the supported (and required) applications that require functional equivalents on the target Linux platform, then you will need to select applications that provide the same functions on Linux. There are many online sources to help guide you in this process. Here are three:

- ▶ “The table of equivalents/replacements/analytics of Windows software in Linux”:
<http://www.linuxrsp.ru/win-lin-soft/table-eng.html>
- ▶ Linuxquestions.org: “Software equivalents to Windows Software”:
http://wiki.linuxquestions.org/wiki/Linux_software_equivalent_to_Windows_software
- ▶ www.novell.com: “Novell Linux Desktop Equivalents of Windows Software”:
<http://www.novell.com/coololutions/feature/11684.html>

4.1.5 Assessing the infrastructure

Survey considerations in this area include:

- ▶ What is the network infrastructure that clients connect to?
- ▶ What is the topology of the network infrastructure? This includes a architectural overview of all local and remote connections including bandwidth and protocol conversion.
- ▶ What type of network protocols are installed and configured on the client in order to access any type of infrastructure components?
- ▶ What servers do the clients connect to and which services do they use? The list of servers should include physical and logical instances. Services can include file, print, DHCP, Web content, dynamic content, applications, and others.
- ▶ What databases are accessed and how is this access implemented? From this, all connections to databases can be derived as well as the type of implementation, such as a native client (such as IBM DB2® UDB client), an API connection (perhaps over named pipes), message queuing (such as IBM WebSphere® MQ), or direct application access (such as an SAP client).

- ▶ What mainframes are accessed, how do the workstations connect to them? Accessing mainframes of different kinds can be done through native clients (such as IBM Personal Communications), Web interfaces (such as IBM WebSphere Host on Demand and IBM WebSphere Host Publisher), or API connections (such as through High Level Language Application Program Interface (HLLAPI)).

4.1.6 Assessing the user

Another key element of a migration is the user. In some cases, especially when talking about success and acceptance of a migration, this is the most important aspect at all. If users are committed to the change, looking forward to working in the new environment, or actively involved in development or deployment of the solution, then user expectation management issues can be minimized.

Questions to ask in this area are:

- ▶ What are the most often performed tasks for a user or group of users?
- ▶ Are there definitions of user roles? If so, how are they defined?
- ▶ Are there any defined exemptions from these user roles or rules?
- ▶ Are there user-specific settings that need to be migrated, such as browser bookmarks or wallpaper?
- ▶ Do any of the existing applications use user-customizable plug-ins for operation?
- ▶ Are users involved in the software selection process, and how?
- ▶ Do users have the option to pull software from an installation repository?
- ▶ Is there a process to request, install, or delete software?
- ▶ How often do users call the help desk and what are the average values of key measurements such as response time, recurring calls, and level of support involved?
- ▶ What is the skill level of users for the base operating system?
- ▶ What is the skill level of users for productivity applications (that is, word processing)?
- ▶ What is the skill level of users for business applications?
- ▶ What type of devices (local and remote) do users require access to?

4.2 Integrating with existing network services

In this section, we describe how to plan for integrating Linux clients in a existing network. We specifically look at how to incorporate Linux clients in a predominantly Microsoft Windows-based network.

4.2.1 Setting the environment

Usually a Linux *client* will not be the first Linux machine added in an environment. Instead, back-end servers are typically the first machines that are migrated to Linux. This means that most descriptions of how to incorporate Linux clients in an existing network will describe the servers as being Samba servers running on Linux. Since this book is focused on client migration only, we decided to consider the scenario where Linux client pilot migration is occurring in an environment where back-end servers have not yet been migrated.

A Linux client migration project could occur within any of the following environments:

- ▶ NT4 domain with Microsoft Windows NT 4.0 PDC (and possibly BDCs)
- ▶ Active Directory (AD) domain with Microsoft Windows 2000 or 2003 servers
- ▶ NT4 domain with Linux running Samba PDC and BDCs
- ▶ Other non-Microsoft Windows domain-based environment

The majority of environments will be the second option, because the first is becoming less common now that NT4 is end-of-life. The third option is becoming more widespread, and most descriptions of client migration already assume the server back-end to be migrated to a Samba domain.

In this book, we concentrate on the first two types of environments (that is, a pure Windows environment) for two reasons:

- ▶ Most domains still fit this model.
- ▶ There are already a lot of descriptions about how to put Linux clients in a Samba domain.

Planning tip: If a server-side upgrade of the domain to either AD or Samba is already planned, take this into account in the client migration planning. Try to avoid integrating twice.

Within the Linux community, the most widely used tool for integrating Linux clients with Windows is Samba. Samba originates from being a successful UNIX application that provides network services based on the Server Message Block (SMB) protocol. This is the same protocol used by Microsoft Windows to provide client/server networking services for file and printer sharing. Thus, Linux systems

running with Samba can integrate seamlessly within a Windows domain for tasks such as accessing network file shares and printing services.

For integration examples using Samba, see Chapter 9, “Integration how-tos” on page 195. Here we indicate technical issues that have to be taken into account when planning the migration.

4.2.2 Authenticating within a Windows domain

In this section, we look at all technical issues that need to be considered when planning for authentication of Linux clients within an existing Windows domain.

Reasons for authenticating a Linux client in an existing Windows domain include:

- ▶ Network services that require domain authentication need to be accessed from the Linux client (such as network file servers, printers).
- ▶ Users will have only a single username and password combination (network services single sign-on).
- ▶ Administrators will only need to administer a single user collection.

There are several ways to authenticate to a Windows domain:

- ▶ Using Samba/winbind without changing the existing infrastructure
- ▶ Using LDAP connecting to Active Directory directly, which means changing the AD schema
- ▶ Using LDAP connecting to a separate directory, which means synchronizing the extra directory with Active Directory

Some of the advantages and disadvantages of these methods are discussed in the next section.

Samba/winbind connect to unchanged Windows domain

Advantages of this option are:

- ▶ No changes needed to domain.
- ▶ Users do not need to be created locally.

Disadvantages are:

- ▶ Winbind does not scale well.
- ▶ Some implementations require creation of a local mapping for users, which might be different on each client.
- ▶ When using the winbind separator, this can impact most applications being used on the client.

The winbind separator

Using winbind will lead to a choice for what is used as the winbind-separator. This is the character that will separate the domain name from the username in the Linux username. For example, AD6380+Administrator is the Linux username of the user Administrator in domain AD6380 when the winbind-separator is a plus sign (+). The impact of the chosen character has to be studied in all applications and network services being used. Using the plus (+) character for separation generally is the best choice for most Linux shells and applications.

Planning tip: Plan and test winbind and the winbind separator extensively to validate the setting prior to migrating clients.

LDAP connect to changed Active Directory

Advantages:

- ▶ Mapping users to uid and gid is done within the AD, that is, centrally.
- ▶ LDAP is a general protocol to connect to AD.

Disadvantages:

- ▶ The AD schema needs to be changed.

Using LDAP means that it might be necessary to change the Active Directory schema to include UNIX-like information such as a uid and gid for users and groups. Microsoft's Service for UNIX can be used to do this.

Planning tip: When planning to use LDAP with Active Directory to authenticate, check whether it is possible to extend the Active Directory schema in your organization.

LDAP connect to directory synchronized with Active Directory

Advantages:

- ▶ Mapping users to uid and gid is done within a central directory.
- ▶ LDAP is a general protocol to connect to directories.

Disadvantages:

- ▶ The two different directories have to be synchronized.

Synchronizing two directories, each containing different objects with partially different definitions, is not a trivial exercise. This is one of the reasons that this method is not discussed in Chapter 9, “Integration how-tos” on page 195.

Choosing to create users locally on the client means extra administrative overhead. In this case when a user is added to the domain, the user ID has to also be added to any of the Linux clients that the user will be using to connect with that domain. Even though this process could be automated, it is really not necessary when using winbind.

In the case of authenticating natively with an Active Directory domain, Kerberos has to be configured as well as Samba.

In summary, the applications that are going to enable us to authenticate with a Windows domain are Samba, Kerberos, winbind, and LDAP.

4.2.3 File sharing using domain shares

Let us assume that some Linux clients are joining a domain and the majority of clients in the domain will still be Windows clients. In this scenario, the best way to share files is through Windows shares.

Mounting shares on a Linux client using the correct `mount` command (`smbmount` or `mount -t cifs`) gives the user access to those file systems on the Linux client, in a way that is much like any other file system they are used to accessing. However, if the share is not open for everybody, a username and password combination is needed, just like under Windows. When mounting a share manually, the login information can be input without problem. However, this might lead to several other problems:

- ▶ Manually mounting shares using the `mount` command might not be an acceptable practice for all users. Users are not used to having to mount shares manually, because under Windows this is done automatically when logging on.
- ▶ A way must be found to enable automatic mounting on login just like on Windows. In Linux, automatic mounting of SMB shares can be accomplished using entries in the `/etc/fstab` file.

A simple solution would be to mount shares in a logon profile. However, this needs a password, and the user is used to shares being mounted using the password supplied upon logon.

A Pluggable Authentication Module (PAM) exists to enable automatic mounting at logon, called `pam_mount`. Since this module is not completely mature yet, care has to be taken to include this in the planning. Currently, it is the preferred way to incorporate some sort of single sign-on-like functionality, such as under Windows, so that all the shares are mounted automatically on login to the Linux client.

When planning for this, extra time should be included to test the pam_mount module on the Linux client chosen for the project. When it works, it is a very powerful method of mounting shares automatically.

Planning tip: Test and plan use of pam_mount extensively.

Home directories and shares

During planning, it might be very tempting to push user home directories on Linux clients to Windows shares. This will enable a “thinner” client.

Putting the user’s home directory on a share would indeed enable logging on to any client and have the same home directory integrated. Using a graphical logon, this would mean getting the same desktop and environment on any client used.

However, care has to be taken. Some session managers (most notably KDE) need to create symbolic links or sockets in the user’s home directory as part of normal operation. One of the shortcomings of the SMB file system is that symbolic links or sockets cannot be created in such a file system. This means that any KDE-based application would have problems when running on the desktop machine in a scenario where home directories are remotely mounted on SMB shares.

Planning tip: A strategy using remote file shares for mounting of home directories will need to be carefully tested.

4.2.4 Printing services in the domain

Of course, it is possible to add printers directly to Linux clients. But this creates extra administrative overhead in a scenario where you are integrating Linux clients into an existing Windows domain that provides network printing services already. Since almost all Linux distributions now include the Common UNIX Printing System (CUPS), it is possible to use CUPS in conjunction with Samba on the client to enable printing to the existing domain printers.

If you plan to use CUPS on the Linux clients to print to existing printers, some issues have to be taken into account:

- ▶ Is CUPS and Samba integration handled correctly?
- ▶ Do the printers need authentication? Domain passwords for shared printer access could be exposed.
- ▶ What are the advantages of using print servers versus direct printing to the printers network interface?
- ▶ Are drivers available for CUPS for each printer model in use?

CUPS and Samba integration

In most distributions, CUPS and Samba will be integrated properly. It needs to be checked, however, and this has to be taken into account when planning. Most importantly, you have to check if Samba is part of the CUPS back-end. How to do this is described in 9.7, “How to use network printers in the domain” on page 219.

Printers and authentication

Printers in the domain can require domain authentication to be able to use them. This is possible using CUPS, by providing the username and password in the URI of the printer. The consequence of this is that the password will be exposed in several CUPS configuration files. The only current workarounds to exposing passwords are either making printers unauthenticated (that is, available to everyone) or creating a special printer user for each printer and only incorporating this special user on those clients that need to print to the server.

Print server versus direct printing

Using CUPS, it is possible to use the domain print servers or print directly to the network interface of the printer (if available). Usually if all clients already in the domain use print servers, it is good to follow this principle for the Linux clients as well.

The advantages of using the print servers are:

- ▶ There is no difference between print jobs coming from the Windows or Linux clients.
- ▶ The printer is controlled from the print server, so when it needs to be rerouted or disabled this is done for all clients.

CUPS drivers

There is quite a big difference in the models and drivers included in CUPS in each of the different distributions. So this is one of the most important steps to check.

If your printer is old, or used by very few people, then decisions have to be made about how much to invest to get it working on Linux clients. The investment for a new printer that works “out-of-the-box” might be a lot less than to get older models to work.

4.2.5 DHCP and DNS configuration

In almost all cases, using DHCP and DNS from a Windows domain will work without problems. This is completely transparent to the user. Once configured correctly for the Linux client, both protocols should work without any problems.

However, some care has to be taken when using DHCP and X11. The Linux client generally accepts host names from DHCP. If this occurs after X11 has started and the user is logged on, then new applications might have trouble accessing authorization information handled by Xauth.

Planning tip: Make sure a DHCP host name is accepted by the Linux client before starting X11 or logging on graphically.

4.2.6 Web proxy interface

The protocol that is used to talk to a Web proxy is independent of the operating system. This means that talking to a Windows Web proxy is as easy as setting the correct settings in the Web browser.

This means that interfacing the Linux client with an existing Web proxy is one of the easiest tasks of the migration.

4.3 Standardizing the desktop

When deploying a Linux desktop solution, you will most likely want a standardized environment used across your organization. While one of Linux's advantages is its freedom of choice, passing this choice on to users in your organization will most likely increase the technical support load. Instead, a standardized desktop environment should be created for your organization, with only slight variations depending on the needs of the user.

Important: Linux's greatest advantages include freedom of choice and 100% top-down client software stack design flexibility. Use these advantages to create simplified and task-focused desktop user interfaces that are designed to limit excessive, non-business-oriented complexity.

Compared to building a standard Windows desktop environment, there can be far more design decisions to be made when building a standard Linux desktop environment. These choices can include the distribution to base your client platform on, the desktop environment users will work with, the look and feel of the environment, permissions available to the user, applications that the users

will use, and even which type of file system to use and a partitioning and mount point strategy for those file systems.

Planning tip: Assuming that you are migrating to a Linux desktop distribution purchased from an enterprise vendor, then you will have the option of using the default predesigned desktop environment as the basis for your clients. In that case, you might not need to design your standard desktop from the ground up, as discussed in this section. In fact, your selection criteria for choosing which enterprise vendor distribution to use could be based in large part on how closely the default desktop provided by that distribution meets your client-side functional requirements.

4.3.1 Linux distributions

Because Linux, and most of the software surrounding it, is open source, this means that anyone can legally create a new distribution of Linux. This empowerment has led to hundreds of various distributions, each with a niche it is trying to fill. Some distributions are made to be easy-to-use, others are targeted toward enterprise users, while still others are based on the idea of extreme customizability. The two distributions that we focus on in this Redbook are Red Hat Enterprise Linux and Novell Linux Desktop (which is based on Novell's SUSE Linux).

There are many reasons to choose one distribution over another. When choosing a distribution, you must consider some or all of the following:

- ▶ Vendor support options and costs
- ▶ Applications bundled with the distribution
- ▶ Administration tools offered with the distribution
- ▶ Availability of third-party software which targets this distribution

4.3.2 Linux desktop environments

While there are numerous desktop environments available for desktop Linux (as seen in 5.2.4, “Graphical and text-based environments” on page 118), the majority of desktop Linux installations feature either a KDE or GNOME-based desktop. These two environments provide a complete desktop environment. This means providing not only a graphical toolkit and window manager, but also a standard environment with desktop icons, a menu panel, and applications such as a file browser, CD burner, and other basic tools. More advanced applications have been written for each environment, such as office productivity suites, music players, photo management tools, and advanced e-mail and groupware clients. Applications that are written specifically for either KDE or GNOME will look similar to other applications written for the same environment and will typically follow global settings that have been configured for that environment. For

instance, if you change the default GNOME color scheme, the GNOME-based Evolution will use the new color scheme but the KDE-based Kontact will not. This disparate look-and-feel, and not any inherent technical problem, typically prevents users from using a KDE-based application on a GNOME desktop and vice versa.

Note: Applications written for a KDE-based desktop will work on a GNOME desktop and vice versa, though they might have a different look-and-feel than the current desktop environment.

Usability tests have shown that modern Linux desktops such as GNOME or KDE are very competitive with Windows desktops. Users with no former computer experience can perform equally on Windows and Linux desktops. Though users who have used Windows desktops will need to spend some time to relearn tasks, the majority of users are able to make the transition with only a small amount of training. More information about Linux usability can be found at:

<http://www.betterdesktop.org>
<http://www.openusability.org>

A lot of work in many different areas had to be done to achieve these results:

- ▶ Accessibility (for users with physical disabilities)
 - Support for assisting technologies such as:
 - Screen magnifiers
 - On-screen keyboards
 - Keyboard enhancement utilities
 - Screen readers
 - Braille displays
 - Text-to-speech programs
 - Speech recognition
- ▶ Programming standards
 - GNOME Human Interface Guidelines¹ (HIG)
 - KDE User Interface Guidelines²
- ▶ Internationalization (UTF-8 support, Pango rendering)

¹ <http://developer.gnome.org/projects/gup/hig/>

² <http://developer.kde.org/documentation/design/ui/>

KDE Desktop

KDE is at:

<http://www.kde.org>

KDE provides a full-featured desktop and application environment for Linux and other UNIX-like operating systems. KDE is based on the cross-platform Qt programming toolkit from Trolltech, and uses the KParts component model together with the DCOP IPC/RCP mechanism, which we describe in Appendix F, “Desktop automation and scripting” on page 321.

With KDE, you can place icons and folders on your desktop and start applications from the KDE panel (Kicker). The panel can also contain KDE applets (such as a clock, battery and wireless status, or a virtual desktop chooser). SuperKaramba³ provides options for even more fine tuning of KDE. More information about KDE applications can be found at:

<http://www.kde-apps.org>

Significant applications for the KDE desktop include:

- ▶ *Kontact* PIM application with *Kolab* or *eGroupware* as calendaring server
- ▶ *KOffice* suite (including *KWord*, *KSpread*, *KPresenter*, *Kivio*, *Karbon*, and *KChart*)
- ▶ *K3b* CD and DVD burning application
- ▶ *Digikam* digital photo management application
- ▶ *Scribus* desktop publishing system
- ▶ *Konqueror* Web and file browser

Note: Apple is using and extending the rendering engine from Konqueror (KHTML) in its Safari Web browser.

GNOME desktop

GNOME is at:

<http://www.gnome.org>

GNOME desktop is an open source desktop environment based on CORBA (ORBit2) and the GTK+ GUI toolkit. It is the default desktop for Red Hat, and is also used by some traditional UNIX vendors including Sun™. More information about GNOME applications can be found at:

<http://www.gnomefiles.org>

³ <http://netdragon.sourceforge.net>

Significant GNOME desktop applications include:

- ▶ *AbiWord* word processor
- ▶ *Gnumeric* spreadsheet program
- ▶ *Evolution* e-mail and calendaring client
- ▶ *Ekiga* VOIP and Video Conferencing Tool⁴
- ▶ *GNUCash*
- ▶ *GTKam* (GUI for gphoto2)
- ▶ *Inkscape* scalable vector graphics (SVG) editor
- ▶ *Nautilus* file manager

As with KDE, you can place icons and folders on your desktop and start applications from the GNOME panel. On this panel, you can also freely arrange GNOME applets (such as a clock, battery and wireless status, or a virtual desktop chooser). For even more fine tuning of the GNOME desktop, *GDesklets*⁵ provides many options to add custom mini-applications (“desklets”) to the GNOME desktop.

4.3.3 Look and feel

After choosing a desktop environment, you might wish to customize the look and feel of the desktop. Choices include choosing the icons that will be used for custom applications or even replacing icons for built-in applications, choosing a theme to make applications look consistent across the environment, and setting up a simple, unified menu structure.

Custom Icons

You probably want a company logo and other brand graphics to be part of your standardized desktop. You might wish to customize desktop icons, such as replacing the e-mail application icon with one fitting the corporate color scheme. Or, you might wish to change the default menu launcher (the equivalent of the Windows Start Menu) to an icon representing your organization. Also, you need to create or find suitable replacement icons for any custom applications that you have deployed.

Simple bitmap graphics are not really state of the art anymore and should be kept to a minimum to eliminate display resolution problems. KDE and GNOME have very good Scalable Vector Graphics (SVG) support in the most current releases. You should use the W3 standard SVG⁶ when designing vector graphics

⁴ formerly known as GnomeMeeting, see <http://www.gnomemeeting.org>

⁵ <http://gdesklets.org>; <http://en.wikipedia.org/wiki/GDesklets>

⁶ <http://www.w3.org/Graphics/SVG/>

logos and other graphics elements for your desktop. SVG images typically look better than bitmaps, are easily resizable, use a standardized format viewable by Web browsers, and are part of a general desktop Linux trend towards SVG-based themes.

Table 4-2 SVG editing programs

Editing program	URL
Inkscape	http://www.inkscape.org
Skencil	http://www.skencil.org
Karbon	http://www.koffice.org/karbon
OpenOffice.org	http://graphics.openoffice.org/svg/svg.htm

Themes

Images and logos are part of the bigger topic of themes. The standard choices the Linux distributors or desktop projects make here (for example, Novell Linux Desktop's Industrial, Red Hat's BlueCurve, or KDE's Plastik theme) are quite reasonable and most users are comfortable with them.

Designing a new theme from scratch is probably not worth the effort and could be confusing to users reading standard documentation from the Linux distributors. If you really feel the need to design a corporate theme yourself, then first visit the following Web sites to check if a theme that meets your needs is already designed and available:

- ▶ <http://themes.freshmeat.net>
- ▶ <http://www.customize.org>
- ▶ <http://www.kde-look.org>
- ▶ <http://art.gnome.org>
- ▶ <http://www.crystalgnome.org>

Even when you decide to use a standard theme, you can make a lot of small changes to refine the look even further.

Some applications include their own theming engines, such as Mozilla-based projects or OpenOffice.org. Also, when running a GNOME-based application on a KDE desktop, the GNOME-based application will use the selected GNOME theme, which might not look appropriate (Figure 4-1 on page 81). Thus when choosing a theme, you might wish to choose one which has variants for all the applications you wish to use. For instance, Red Hat's default BlueCurve theme is available for both KDE and GNOME. Also, there is a Firefox theme⁷ which matches Novell Linux Desktop's default Industrial theme.

While using a Windows XP theme with KDE or GNOME might seem like it would ease the transition to Linux, it probably is not a good idea, because users will expect exactly the same behavior they experienced with Windows, which could be quite confusing.

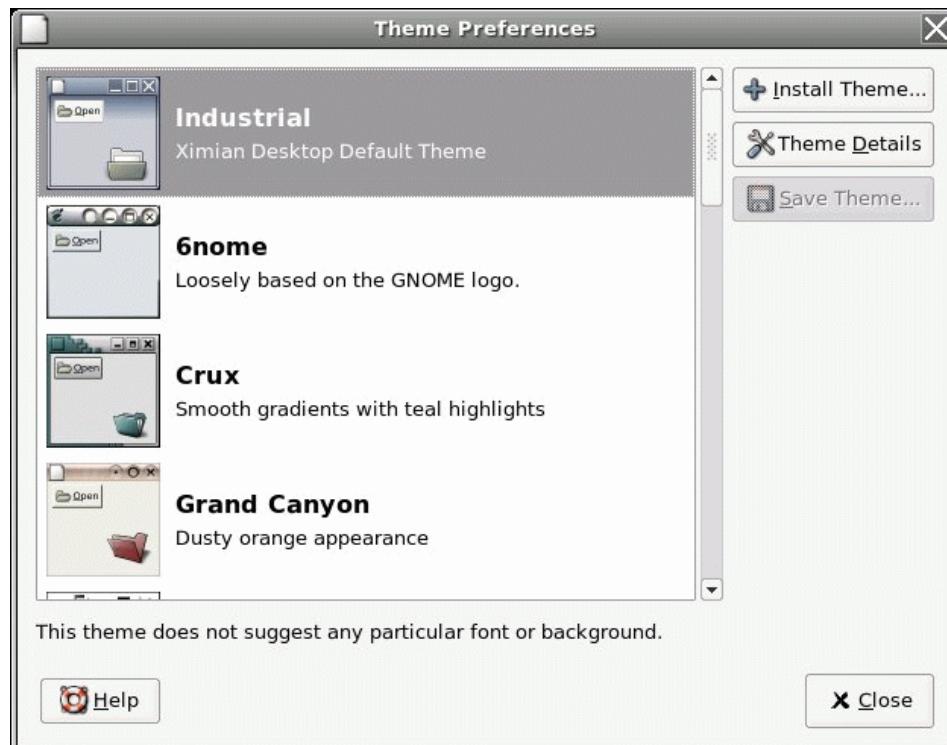


Figure 4-1 *Theme Preferences dialog in GNOME*

Application menu design

Another desktop standardization topic to consider is the application menu design. In enterprise environments, users do not need five different applications appearing in their menus that do basically the same thing. SUSE Linux Enterprise Desktop 10 takes this approach further, by limiting the total complexity in the root "Computer" menu to a few specific productivity applications plus a few entry points to some other frequent tasks. An example screen capture of the default SLED 10 menu structure is shown in Figure 4-2 on page 82.

⁷ available at <http://primates.ximian.com/~glesage/stuff/firefox/>



Figure 4-2 Simplified, task-oriented menu design (SUSE Linux Enterprise Desktop 10)

In fully controlled environments, you might want to reduce menu selections even further and lock them down so that users cannot make changes to their configuration (such as adding or changing panel applets, desktop launchers, or menu entries). KDE provides the *Kiosk* framework to restrict the capabilities of the KDE environment. Although primarily designed for unattended operation of kiosk terminals, these features can also be very valuable in enterprise settings where a more controlled environment is desired. We will discuss the Kiosk framework in 7.1.1, “KDE Kiosk framework” on page 140.

Important: Changing the ownership and read-write flags of configuration files is not necessarily a foolproof lockdown mechanism, because the user can potentially rename the files and create new ones with the correct permissions.

If your desktop environment needs to support both KDE and GNOME desktops, then you will want to ensure that both desktop environments have similar menu structures. Until recently, keeping a unified menu structure was difficult because both KDE and GNOME used different format for menus. However, as of KDE 3.3 and GNOME 2.12, both environments now support the freedesktop.org Menu Specification, available at:

<http://standards.freedesktop.org/menu-spec/latest>

4.3.4 User lockdown

When setting up a Linux environment, there are several options for locking down workstations. While in a technical workstation, users can be given complete control over their workstations (by giving them the root password), most desktop environments should limit what a user can do.

On many Linux distributions, some hardware and software features are limited based on group membership. For instance, there is often an `audio` group, and only members of this group have access to the sound card. There can also be a `games` group, where only members of this group can play any games installed on the system. Similar restrictions can be made to any hardware device or application on the system.

You might want to limit what menu items are shown to the user, and what settings can be configured. More information about this topic can be found in 7.1, “Restricting the desktop” on page 140.

4.3.5 Application choices

One of the main benefits of open source software is that it provides choices. However, sometimes these choices can be overwhelming. For instance, some distributions install four text editors, five terminals, and six e-mail clients. A user that is new to Linux will not know which application to choose. Depending on the technical knowledge of the user, the number of applications available for each task should be limited. The majority of installations should only include one e-mail client, one Web browser, and probably only one desktop environment. However, if the users are used to more control over their desktops, they might resent some of the choices selected. This is another situation where user desires need to be balanced against technical support considerations.

4.3.6 File systems and partitions

As discussed in 5.2.6, “Drives, partitions, and file systems” on page 123, Linux provides many more options of file systems and drive partitioning schemes. When configuring the standard desktop environment, drive partitioning should be

homogeneous across the desktops. Also, locations for network share points should be standardized, for ease of technical support among other reasons. Because the new mount point configurations can confuse users, you might wish to add the old drive names in brackets when configuring desktop folder icons, so that users will recognize their old environment.

Also, you will need to decide if you wish to use advanced ACLs or the standard access levels as provided by the file system. Depending on your environment, the granularity of control provided by the standard permission set might be sufficient. If ACLs are required, then you will need to make sure that all tools dealing with security (such as the graphical file manager or backup tools) have complete support for ACLs.

4.4 Migrating applications

A migration path needs to be determined for each application that will be migrated to a Linux-based equivalent. Different migration paths will be necessary, because not all Microsoft Windows-based applications have Linux-based equivalents. Some example scenarios include:

- ▶ Bridging applications: These have native equivalents for both Microsoft Windows and Linux. See 3.2.1, “Bridging applications” on page 53, for the definition and importance of using bridging applications if possible.
- ▶ Similar applications: Applications that provide the same functionality and usually data import capabilities. For instance, OpenOffice.org provides word processing, spreadsheet, and presentation capabilities and can import Microsoft Office files. See 3.2.2, “Functionally equivalent utility applications” on page 54.
- ▶ Server-based applications: For an application that has no Linux-based equivalent. In this case, application servers provide some type of remote terminal service, and the Linux client then runs the application using a remote desktop application interface. See 3.2.4, “Building bridges to the server” on page 55.
- ▶ Web-based functional equivalent: Provide a platform-independent application interface that is browser-based. See 3.2.3, “Web applications” on page 55.

4.4.1 Moving back to client/server computing

When planning for a Linux-based client migration, it is possible that you might find compelling reasons to consider moving to a client/server computing architecture for certain application services. This consideration becomes especially evident when the Linux migration coincides with an application

migration to a Web services-based model (for example, moving from a PC-based application to an equivalent Web portal-based solution).

Thus, we need to consider patterns for logical segmentation of workstation types, as discussed in the following section.

4.4.2 Logical segmentation - Thin, slim, or fat

Generally speaking, a major shift from an existing client computing environment can also trigger an expansion in workstation types that are used by the organization.

For the purpose of this logical segmentation discussion, we define three major “types” of workstations, as follows:

- ▶ **Thin:** Always connected; no local applications besides a displaying component, which could be either a Web browser or any other client component of a terminal services environment.
- ▶ **Slim:** Intermittently connected; some applications running locally (that is, not requiring synchronous server connectivity). This could be any type of implementation of such components ranging from self-written client/server to Eclipse-based Java components in a portal-based solution.
- ▶ **Fat:** Off-line capable; most applications are locally installed and run locally on the client.

Within a client/server environment, Figure 4-3 on page 86 shows the progression from thin to fat, where the important difference is in how many layers of the application logic are hosted locally on the client.

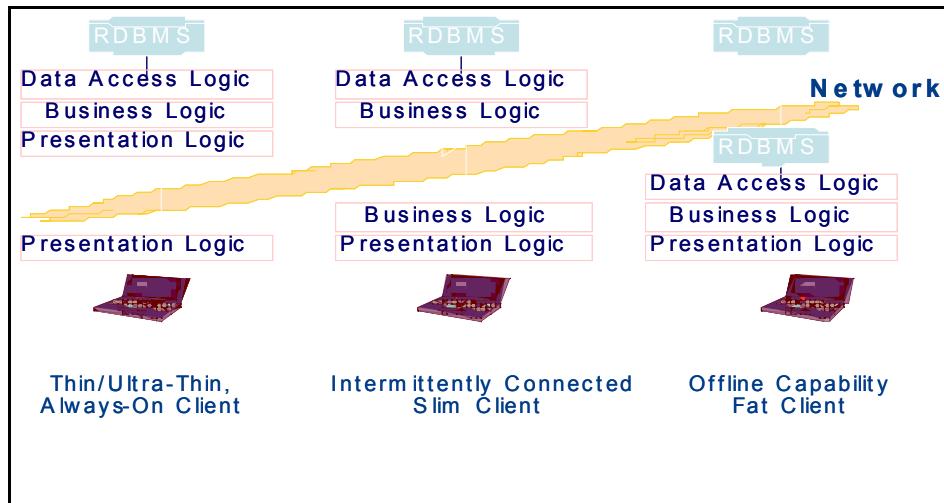


Figure 4-3 Thin, slim, or fat client

4.5 Client administration planning

This section describes several methods for efficient administration of Linux clients. For a corporate network with more than twenty clients, it becomes impractical to install and maintain clients individually. An added advantage for maintenance of the Linux client is that remote logon is possible, either through SSH, telnet, or even an xterm connection.

The topics discussed in this section have to do with administering installed Linux clients after a migration. We do not discuss initial installation or rollout of the Linux clients in this section. Topics in this section are relevant to migration planning, because it is important to consider the impact of post-migration administrative tasks.

At the end of the section, we highlight two enterprise desktop distributions to show how the administration issues are solved using tools provided by those products.

Planning tip: Even though this section provides details about daily administration tasks, after a migration, these topics are still relevant. It is extremely important to consider the impact of post-migration administrative tasks during migration planning.

4.5.1 Operating system and vendor distribution updates

Keeping up with security concerns, and a constant stream of enhancements and bug fixes that are submitted by the OSS community as well as distribution vendors, means that there is an essentially continuous stream of updates and patches that are available. Methods for managing the update process include:

- ▶ Automate all operating system updates and patches.
- ▶ Facilitate the operating system update process for the user.
- ▶ Force critical operating system updates and patches when necessary.

In the automatic update process, the operating system will be updated at regular intervals. This means that a client will never be “out-of-date”. The process is completely out of the user’s control.

When facilitating operating system updates for the user, the user needs to perform some action to update. One possibility is to put updates on the corporate intranet and notify users to fetch and install them. This process can lead to clients with varied status, some up-to-date and some out-of-date.

In both cases, a mechanism has to be put in place that forces critical updates and patches. This can be an administrator running a script centrally, which contacts all clients remotely and forces an update using either the automated facility (casing it to run immediately instead of once a week) or the facilitated fetch and install method. This mechanism is needed in case of severe security problems where patches need to be applied immediately.

A typical Linux distribution consists of a lot of parts that all have independent updates. This means that there can be many different states of a client. Certainly a lot more than under Windows, which generally aggregates fixes in service packs (even Windows updates are generally small service packs and include changes for multiple parts of the operating system). To be able to administer a larger pool of Linux clients, it is important to keep all at more or less the same level and version. This means that the automated update process is preferred over the user driven process.

Most distributions have tools to do this update automatically. Examples are:

- ▶ Red Hat Network (up2date) tools on Red Hat Enterprise Linux; see “Administration of Red Hat Desktop” on page 91.
- ▶ YaST online Update on SUSE Linux at:
<http://www.novell.com/coololutions/feature/11823.html>
- ▶ Novell ZENworks Linux Management.
See: “Administration of Novell Linux Desktop” on page 93.
- ▶ OSS alternative update management applications include apt and yum.

Planning tip: Automate operating system updates and patching to keep all clients at the same level. If possible, as a best practice you should maintain a private enterprise accessible repository of authorized updates.

4.5.2 Application updates

If the application is included with and maintained as part of the distribution, then the tools mentioned in the previous section can be used.

When the application is not part of the distribution, there are several ways to approach the update:

- ▶ Use scripts to fetch updated files and configurations from a network server.
- ▶ For third-party applications, an update executable or script might be provided.
- ▶ Build a package that can be handled by OSS tools such as apt and yum.
- ▶ If the change is extensive and the client is thin, replace the entire client image with a new one.

In early proof-of-concept tests and pilot migrations, most of the changes can (and will) be done manually. When the size of the pool of Linux clients grows, this quickly becomes unfeasible, and more automated methods need to be put in place.

Planning tip: Plan to create an update mechanism during early proof-of-concept or pilot migration of the Linux client.

Special care should be taken to make sure that the desktop and menu structure are not adversely impacted when updating (or maybe even removing) applications. The menu should keep pointing to existing applications to prevent stale menu items. Icons on the desktop should point to applications still available, and to the right version if multiple versions exist simultaneously.

Important: Always include desktop and menu structure in changes related to application updates. A successful update can appear failed, because the users are left with stale icons or menu items.

4.5.3 Remote administration

The Linux client supports various remote administration methods without having to install extra software. An administrator can log on remotely through one of the standard mechanisms, for example, by using SSH. This enables the

administrator to analyze and fix problems remotely (except networking problems that prevent remote access). It also enables the administrator to remotely run scripts to monitor clients and gain data about the state of the clients.

Monitoring of the clients is used to proactively prevent problems instead of reacting to users not being able to work. It is used to detect problems with disks, memory, and CPU usage, or even with certain applications. There are several products or solutions available to monitor systems, from commercial products such as IBM Tivoli Software to OSS solutions such as Nagios and Big Brother.

4.5.4 Rollout of additional or replacement clients

In a steady state situation (that is, after the initial rollout of Linux clients), there will still be requirements for on-going replacement and addition of client systems. For both tasks, there should always be an up-to-date system image that can be used to build the client. Some extra updates after installing the initial client should not be a problem, but building a new client should not mean applying very many updates after the initial image installation to bring it fully up-to-date with the current supported image.

When replacing a client, there are several things that have to be taken into account:

- ▶ In the case of a thick client, personalization data has to be transferred from the old replaced client or restored from backup.
- ▶ Configuration files need to be copied over.
- ▶ Care has to be taken that using the old name for the system will not lead to extra unnecessary problems.
- ▶ When using a new name on a replacement, make sure that the old name is removed from server-side configurations.

Important: In the case where winbind is used, take care to copy the winbind idmap file from the replaced client. Because this keeps track of domain user to uid and gid matching, this is very important for file ownership.

On Red Hat Enterprise Linux systems the winbind idmap file is:

`/var/cache/samba/winbindd_idmap.tdb`

On Novell Linux Desktop and SUSE Linux systems the winbind idmap file is:

`/var/lib/samba/winbindd_idmap.tdb`

4.5.5 Backup of clients

Backup of the client is only important if there is data stored locally on the client. If all important data is kept on a reliably backed up server, then replacing a broken client just means resetting with a clean initial client image or install.

If the client contains important application data, this has to be backed up. This is generally implemented in a client/server configuration where a client of the backup software is installed on the client that sends its data to the backup server. The backup server will then write the data to offline storage (such as a tape drive). One such product that supports Linux clients for this purpose is IBM Tivoli Storage Manager.

However, as stated before, some configuration and cache files are important and do contain data even if no applications write data to local file systems. For example, users can be allowed to make some changes to their desktop. Restoring these changes after a replacement of the client can be seen as a service to the user. Otherwise, the user might have to spend several hours getting it just right again.

We could make sure all the necessary files are kept by making an inventory of all these needed files during early migration or proof-of-concept and implementing a script that will collect all files in a zip or tar file and place them on a server before that server is backed up. Because these files are generally not that large, this method does not have a large impact on the backup.

Do remember to update the list of files as applications change or names of configuration files change with new versions.

By remotely mounting user's home directories, as described in "Home directories and shares" on page 73, you might be able to greatly minimize the amount of client specific backup tasks that need to be supported.

Planning tip: Implement a method to back up key configuration and cache files from the client, to be used after client restore.

4.5.6 Virus mitigation

It is important to start protecting Linux clients from virus infection as soon as the clients become operational. Even though there are few viruses targeted at Linux, the number is expected to grow as the number of Linux clients grows.

Virus or worm protection is done in three ways:

- ▶ E-mail virus protection on the mail server
- ▶ Regular virus scanning on the client (perhaps once a day or once a week)

- ▶ Configuration of the client firewall to protect against worms

To scan for viruses on the mail server, several solutions are available, both OSS and commercial.

Virus scanning on the client should include all parts of the local file systems. Several solutions, both OSS and commercial, are available. The virus definition files of the tool chosen should be regularly updated automatically. The user should not be allowed to stop the scanning operation.

The Linux client firewall should be configured to defend the client against infection by worms. The firewall configuration should not be changeable by the user and should be regularly, centrally updated.

Planning tip: Plan to start virus mitigation procedures on Linux clients early in the migration process. Linux viruses will appear eventually. Be ready.

4.5.7 Examples of administration of enterprise distributions

In this section, we highlight how Red Hat Enterprise Desktop and SUSE Linux Enterprise Desktop can be managed using vendor integrated system management frameworks.

Administration of Red Hat Desktop

The Red Hat Desktop distribution is based on Red Hat Enterprise Linux. Like all other products in this range, the mechanism for administration and update is Red Hat Network (RHN).

Red Hat offers three architectures for connecting to RHN: Hosted, Proxy, and Satellite. Usually a client will connect directly to the RHN servers at Red Hat. This is called the *Hosted architecture*. Included in the Red Hat Desktop offering are the Proxy and Satellite architectures, where you can install a RHN proxy or RHN satellite server locally. The RHN proxy server caches traffic and content between the RHN servers at Red Hat and the local client. Using a satellite server, it is possible to replicate a RHN server locally.

The proxy and satellite servers are useful when the number of clients increases and the amount of traffic to Red Hat becomes large. Also, these solutions will increase security because there is a limited connectivity to the Internet. An added advantage of the satellite server is that the RHN solution can be taken offline (disconnected from the Internet) if you choose.

Updating and patching the desktop client is done using **up2date**. The RHN alert notification tool is used to alert the user that an update or patch is available. The update process can also be automated.

Using RHN, there are three types of modules:

- ▶ Update
- ▶ Management
- ▶ Provisioning

These modules determine the service level you get from RHN.

The Update module only gives you basic update capabilities such as:

- ▶ Access to a Web interface to manage the systems.
- ▶ Priority e-mail notification of new updates and errata.
- ▶ Errata information provides a list of all available errata updates for each system.
- ▶ RPM dependency checking makes sure that every applied update gets all necessary dependencies.
- ▶ Auto update allows a system to download and install errata and updates autonomically.

The Management module provides all capabilities of the Update module plus extended management capabilities:

- ▶ Systems can be managed as a group.
- ▶ System permissions can be assigned to administrators and groups of systems.
- ▶ Actions such as updates can be scheduled for a system or group.
- ▶ System search allows for searching systems and groups by package, errata, or system specifications.
- ▶ Detailed comparisons of packages can be produced between systems or against a pre-built package profile.

The Provisioning module is really useful when RHN is used to create new clients. Along with everything in the Update and Management module, Provisioning also provides:

- ▶ Bare metal provisioning - A tool to automatically provision a system using kickstart to deploy operating system, packages, and activation keys.
- ▶ Existing state provisioning - Provision a system to take the state of an existing system of a predefined installation.
- ▶ A rollback functionality to roll back updates.

- ▶ Configuration management that can combine with kickstart actions to enable a complex provisioning action.
- ▶ Provision applications based on RPMs.
- ▶ Kickstart configuration writer.

All these options are available in the Hosted architecture. The Proxy and Satellite architectures add extras such as custom channels, integrated network installs, and much more.

Using RHN, it is possible to remote manage the Red Hat Desktop clients. For the Hosted architecture, the management is limited to the operating system and applications included in the distribution. Using the other architectures, it is possible to create channels for third-party applications.

More information about RHN is available at:

<http://www.redhat.com/software/rhn>

Note: More information is also in “Red Hat Satellite server and Red Hat Network (RHN)” on page 257.

Administration of Novell Linux Desktop

The preferred method to update the Novell Linux Desktop is through the use of Novell ZENworks Linux Management. ZENworks Linux Management is based on Red Carpet Enterprise.

ZENworks offers a central system where client systems can be configured and where updates and patches can be pushed to the client. The central system resides inside the firewalls of the organization and can get its content (packages) from a variety of sources including Red Carpet servers, YaST online Update, and even Red Hat Network.

The clients connect to the server to update and patch using the **rug** or **red-carpet** programs. This connection uses the HTTP protocol. Administration is done on the ZENworks server using either a Web interface or command line commands.

The activation key that is used to activate the client actually specifies which administrators have access, which channels are available, and to which groups the client belongs.

Some of the features included in the ZENworks Linux Management are:

- ▶ Enhanced ACL - Closely related to Activation keys. Administrators, channels, groups, or machine notifications can be associated with an activation key.

- ▶ Check pending updates - Administrators can see which clients have updates pending and the importance of these.
- ▶ Package sets - Software packages can be grouped to a set. The sets behave as packages and can have a version and dependencies.
- ▶ Ad-hoc groups - Groups can be created whenever an Administrator needs one; for example, for all machines with a certain update pending.
- ▶ Enhanced transaction - Immediate or future execution scripts can be added or rolled back.
- ▶ Set client permissions - Client configuration files can be controlled from the server.
- ▶ Server-side dry run - A transaction can be tested on the server to check dependency information without taxing the client.
- ▶ Machine comparisons - Tool lists the differences between two or more clients.
- ▶ Choice of database back-end - PostgreSQL or Oracle.
- ▶ Caching server - Server that caches software on different subnets.
- ▶ Enhanced mirroring tool - Tool generates package sets from product descriptions provided by mirrored servers.

The client has the following additional features:

- ▶ Multiple server support - The client can access multiple ZENworks Linux Management servers.
- ▶ Time-based rollback - Can be invoked locally and from the server.
- ▶ Expanded system information - The client can send hardware, software, and system information to the server.

More information about ZENworks Linux Management can be found at:

<http://www.novell.com/products/zenworks/linuxmanagement/>

Note: More information can also be found in “Novell ZENworks Linux Management” on page 264.

4.6 Desktop versus notebook considerations

While looking at differences between desktops and notebooks, not only the included hardware pieces differ, but the way of working might be different, too. Users of notebooks often work offline and therefore need to be able to synchronize their data once they are back online. Another topic in this relationship is remote connectivity through wired or wireless networks.

Not only should the hardware support working in disconnected mode; the software should also have mechanisms necessary to adapt to the current connection type if necessary.

Planning tip: Notebook computer users are extremely likely to fall into the advanced end of the client functional segmentation shown in Figure 3-1 on page 50. Because of this, migrating notebook users will always require an extra level of technical planning, both for hardware compatibility issues as well as application migration.

4.6.1 Hardware considerations

When discussing hardware support in Linux, it is important to first state that support has improved significantly in the last few years. More manufacturers see the rising distribution and use of Linux both in private and in corporate environments, and therefore have started to focus more resources toward delivering compatible device drivers for Linux. Unfortunately, there is still a lot to be done, especially in areas such as wireless LAN device support, or specialities such as Winmodems or GDI-printers. Also, the open source community runs several projects that develop device drivers, such as the WLan driver for the Intel Centrino chipset. A problem in this context is the firmware files, which are mostly closed source and so open source developers have to sometimes reverse engineer the device interfaces to achieve some level of functionality for native Linux device drivers.

Developments such as this have even led to ideas such as the ndiswrapper project, which enables the loading of Windows ndis drivers in order to run network cards under Linux. NDIS is the Windows network driver API, and ndiswrapper allows use of this API as kernel modules in Linux. The URL of this project is:

<http://ndiswrapper.sourceforge.net>

It is very important that the hardware types are determined during the planning stage of a client migration (see 4.1.1, “Assessing the client hardware” on page 63). Existing hardware could need more effort to get it running properly on Linux than the hardware is currently worth.

This rule applies especially for the selection of PC models. Almost every PC vendor provides a model type that is suitable for Linux. IBM provides a Web site with information about supported models, which can be found at:

<http://www.ibm.com/pc/support/site.wss/MIGR-48NT8D.html>

Regarding desktop computers, it is quite easy to determine if Linux will run without problems; all the components, such as the graphics card, are known and

can be checked for Linux support. Onboard components need special consideration because they likely cause problems; but even if this happens, it is possible to work around this problem, such as by inserting a separate AGP card with a supported graphics chipset.

When considering notebook computers, such an exchange cannot be made, because the components cannot be removed from the mainboard. Thus, it is necessary to check Linux support for notebook models before purchasing them.

Of special concern for mobile Linux-based computers is support for power management functions. The support of the standards APM and ACPI in Linux is still lacking in some areas. ACPI has been supported in the kernel since Version 2.6; but the implementation in hardware drivers varies, resulting in problems for some users. For example, it is frequently more reliable to use APM for systems than it is to use ACPI.

Note: One method to test hardware compatibility is to use a bootable CD-ROM image of the Linux distribution that you are considering migrating to. These live CDs are designed for this purpose. See 3.4.5, “Hands-on Linux prior to migration” on page 59 for more details. We highly recommend that you use this method to verify existing laptop system support.

In conclusion, it is very important to pay special attention to the hardware that will be in use after the migration. Additional work might be required to have a customized version of a distribution that supports all components, if it is even possible to support every component.

4.6.2 Peripheral extensions

When you look into the client landscape, you will recognize that many users not only have a desktop computer, but also different kinds of peripherals. Examples include:

- ▶ Locally connected printers
- ▶ Scanners
- ▶ Plotters
- ▶ Card readers
- ▶ Digital cameras

While planning the migration, it is important to assess peripheral hardware in the same way that you assess the client hardware. Considerations about that are given in 4.1.1, “Assessing the client hardware” on page 63.

Especially in cases of cheap peripherals, we recommend building a list of approved and standardized devices, and aligning this with your inventory. If some

existing peripherals are hard to support in Linux, it is probably less expensive to buy new devices that are known to run under Linux.

Information about open source projects that support peripheral extensions can be found on the following Web sites:

- ▶ Common UNIX Printing System (CUPS), see 9.7, “How to use network printers in the domain” on page 219, too):

<http://www.cups.org>

- ▶ Scanner Access Now Easy (SANE):

<http://www.sane-project.org>

- ▶ UNIX SmartCard Driver Project:

<http://smartcard.sourceforge.net>

- ▶ gphoto2 - Digital camera support:

<http://www.gphoto.org>

4.6.3 Connectivity options

Now that the Internet is seemingly everywhere and many services are “on demand”, connectivity becomes one of the most important features of a client. The Internet was created on UNIX-based servers, and because Linux was created as a UNIX-like operating system for the x86 architecture originally, it should come as no surprise that Linux easily fulfills network connectivity requirements.

The majority of Ethernet network cards are supported. Because DSL and Cable modem Internet connections operate over an Ethernet network interface, support in Linux for those devices is also common. For DSL connections, PPP and PPPoE Protocols are supported. Cable modem connections are easy, too, because they usually provide IP addresses via DHCP.

Wireless network device compatibility is where you will need to be most careful in testing your target client systems and devices. In this situation, we recommend that you choose devices from larger manufacturers; for example, Cisco offers Linux drivers for its wireless network cards. Support for wireless devices has improved significantly, so there should be no problem finding a solution to give your users wireless connectivity.

Supporting dial-up connectivity can be more problematic, especially in the case of winmodems, which are extremely common in notebook computers. These are modems that use software to emulate the hardware components, and therefore, you need special software in order to use it.

For winmodems, some Linux drivers are available for some models. Up-to-date information about this can be found at:

<http://linmodems.org>

The support of these devices is dependent upon the distribution. Sometimes, it is easier to use an external modem with a supported chipset than to spend a lot of effort to get the internal winmodem working.

4.6.4 Offline mode

In this section, we discuss working in offline mode.

Offline messaging

There are many full-featured desktop e-mail client applications available (such as Microsoft Outlook or IBM Lotus Notes) that provide for offline access to a user's e-mail. Frequently when planning a Linux client migration for mobile (laptop-based) users, supporting a Linux-based messaging client with offline capabilities will be the single most important design decision in a mobile desktop application migration strategy. There are Linux-based messaging client options that provide for offline access modes. Mozilla Thunderbird is one example; Thunderbird supports both POP and IMAP protocols. More information about Thunderbird can be found at:

<http://www.mozilla.org>

Although Thunderbird will support offline messaging modes, it does not provide a way to integrate calendaring and scheduling functionality into the messaging client that is provided by systems such as Microsoft Outlook with Microsoft Exchange back-end and IBM Lotus Notes with Lotus Domino back-end. For this reason, we choose to demonstrate use of Novell's Exchange Connector combined with the Novell Evolution e-mail client in Chapter 8, "Client migration scenario" on page 173. Extending the example migration scenario in Chapter 9, "Integration how-tos" on page 195, for the notebook user means that you will also have to test and evaluate the offline capabilities of that solution.

Offline files

Because every user to some extent creates and manages files and folders, it follows that the notebook user should be able to access copies of their files in offline mode. However, the primary storage location for these files should be a regularly backed up and highly available server.

For example, Microsoft provides methods for automatic synchronization of files between servers and mobile clients. One of these methods uses the *Briefcase*, a special folder whose contents are synchronized at specific intervals. A second

method was provided with the launch of Windows 2000, where a new function called offline files was implemented. This mechanism is able to synchronize whole folders. While the notebook is offline, the files are fully accessible, and changes are retransmitted automatically after connecting to the network.

In Linux, this kind of server-to-client file synchronization is not built into the operating system itself. Instead, the task of developing and supporting Linux-based file synchronization services between client and server is left to ISVs. For example, Novell offers a solution called iFolder for taking files offline and sharing them with other users. There are two servers available for iFolder, one is open source and the other is a commercial product. More information can be found at:

<http://www.novell.com/products/ifolder>

Another possibility is to use rsync for this purpose. *rsync* is a powerful open source Linux utility that performs incremental file transfers. Used with the right options, it can fulfill the demand for synchronizing local and remote folders. Because it builds checksums of the files and transfers only the missing parts, rsync is extremely effective over small-bandwidth connections. The Web site for rsync is:

<http://samba.anu.edu.au/rsync/>

By using rsync, you can develop a client/server file synchronization tool that could be run as both a scheduled or user-triggered service on notebook computers. We recommend that you develop a user-friendly wrapper that controls this process, and, thus, hides complexity from the user.

4.7 Unmigratable applications

This section discusses issues related to applications that cannot be migrated to run directly on a Linux client. We propose alternate ways to support these applications for access from a Linux client.

We assume that the number of unmigratable applications is small and that a small percentage of users are using these applications. If this is not the case, then the case for the client migration has to be reconsidered.

4.7.1 What makes an application unmigratable

We define an application as unmigratable when one or more of the following statements about the application are true:

- ▶ A Linux version of the application or an alternative application does not exist.
- ▶ Porting the application to Linux is not feasible.

- ▶ License issues make a move to Linux impossible or highly expensive.

Once an application is designated as unmigratable, there are several ways to migrate to a Linux client and solve the issues around this application:

- ▶ Investigate whether the application can run on a Windows server and be used through remote access mechanisms such as a Terminal Server, Citrix Metaframe, or NoMachine NX.
- ▶ When using Workplace™ Managed Client on the Linux client, investigate the use of Ericom Powerterm WebConnect for Workplace to connect to Windows servers containing unmigratable applications.
- ▶ Examine whether it is possible from a cost perspective to run VMware workstation on the client to create virtual Windows machines on those clients that still need to run the application natively in Windows.
- ▶ Create dual-boot clients if the application is not used very often.
- ▶ Leave some clients out of the migration and consolidate all unmigratable applications on those shared clients, for use by all users who need to use those applications.

4.7.2 Terminal Server, Citrix Metaframe, or NoMachine NX solutions

If the application can run as a centralized application on a server, solutions such as Windows Terminal Server, Citrix Metaframe, or NoMachine NX can be used to access them remotely. Before an application can be moved to a central server, certain conditions have to be met:

- ▶ The application has to be able to run on a multi-user environment. A multi-user environment has several consequences:
 - Settings must be stored in a per-user location in the registry and file system.
 - More than one user can run the application simultaneously.
- ▶ The application license must allow you to run it in a multi-user environment.
- ▶ The servers must have enough resources to allow you to concurrently run multiple instances of the application or other non-migratable applications.

Client applications are available for Linux for Windows Terminal Server, Citrix Metaframe, and NoMachine NX. The client for Windows Terminal Server, rdesktop, is open source; the Citrix client is available when you purchase Metaframe solutions; and NoMachine has released the NX client under the GPL.

4.7.3 Ericom Powerterm WebConnect for Workplace solution

In case the client is already running Workplace Managed client to enable Lotus Notes or productivity tools, you can also use this to enable some unmigratable applications.

Using Ericom Powerterm WebConnect makes it possible to connect to applications running on UNIX, a mainframe, and Windows servers. You can use this last capability to connect to applications that cannot be migrated to the Linux client, but that can be centralized on a server.

Using Ericom Powerterm WebConnect for Workplace, you can centralize the management of these applications. The release of Windows applications through Powerterm WebConnect for Workplace can be controlled and managed centrally.

More information about this can be found in Appendix B. of *IBM Workplace Managed Client 2.6 on Linux*, SG24-7208:

<http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247208.html?Open>

4.7.4 VMware solutions

Using VMware (or any similar application that runs on Linux), it is possible to create a virtual Windows machine that the user can use to work with applications that cannot be moved to Linux or a central server.

The virtual machine is a completely functioning Windows machine and can be loaded with the image of a Windows client already in use before the migration to Linux. When using a domain, this means that the virtual machine has to become part of the domain as well. The virtual machine has to be connected to the network via bridging or NAT.

This solution has several disadvantages:

- ▶ A full Windows license is needed for the virtual machine.
- ▶ VMware software needs a license also; this will lead to extra cost.
- ▶ Extra management tasks to keep VMs running on clients.
- ▶ Possible extra requirements on the client resources in terms of memory and disk space.
- ▶ The user might be tempted to work in the virtual machine to avoid using the new Linux client.

Whether this solution is feasible with the extra cost depends on the number of clients involved and the cost of alternative solutions.

4.7.5 Dual boot solution

Using a dual-boot client to work around unmigratable applications is essentially the same as the VMware solution, except that in this case, the user can use either Linux or Windows but not both at the same time.

The disadvantages of this solution are:

- ▶ A full Windows license is needed to boot the client using a Windows operating system.
- ▶ Extra management and support tasks are needed for the Windows part of the dual-boot system.
- ▶ Extra user complexity:
 - Context: It might appear unclear at times whether the client is running Windows or Linux.
 - The user cannot switch quickly and will need an extra partition to work with files on both operating systems.
- ▶ User might be tempted to boot to Windows and keep working using the old client.

Advantages over the VMware solution are that there are no VMware license costs, and no extra memory or CPU power is needed since only one operating system will be running at any one time. In this case extra disk space is needed in the form of at least one partition. For both the VMware and the dual-boot solution, it is best to make a minimal Windows client available. This way, only the application that still needs Windows is run under Windows.

4.7.6 What to do if all else fails

There might be applications left that do not lend themselves to be moved off of a Windows client using any of the methods that we have mentioned. If these applications are to be used after the client migration because of economical, legal, legacy, or other reasons, a solution has to be found.

At present, it is possible to use a Windows emulator (such as Wine) to support running Windows applications on a Linux-based client.

The most economical solution in the case of unmigratable applications might be to consolidate all unmigratable applications to a fixed number of Windows clients. These are then used by the entire user population to access the unmigratable applications. These central clients will most likely be used in a single-user mode with a single user sitting at the keyboard; if multiple users can access the application at once it is better to use a remote solution such as the one described in 4.7.2, “Terminal Server, Citrix Metaframe, or NoMachine NX”.

solutions” on page 100. The centralized clients can be used remotely by tools that enable remote use by a single user. An example of a tool such as this is VNC. A VNC server can remotely connect the display, mouse, and keyboard to the VNC client on the Linux desktop. By using a secure socket connection, it is possible to make sure that the remote connection is secure.

Of course, if the unmigratable application is a heavily used application, all of these methods are unusable and a Linux client migration becomes extremely difficult.

4.8 Deploying the new client

Once a suitable Linux client has been designed, it has to be deployed within the organization. The architecture and methodology for deployment has to be worked out in the planning stages. Otherwise, the pilot project might end up with a couple of very nicely designed Linux clients, but without demonstrating a proven method for efficiently managing deployment of that client image to the rest of the organization. In the case of larger organizations with many hundreds or thousands of clients, this can become a critical consideration.

Several planning issues in deploying the client are:

- ▶ Method: Use complete image with post-deployment script or use the distribution automatic installation method such as Red Hat’s kickstart or Novell’s autoyast.
- ▶ Update frequency: How to best update the deployed clients
- ▶ Personalization of the client: Store user data on the client or in the network?

4.8.1 Deployment method

Different methods exist to deploy the new Linux clients. Before designing and creating the new clients, you must select and test the method.

Planning Tip: Decide on a deployment method before designing and creating the new client. The method of deployment directly determines the extent to which the installation and customization of the new clients can be automated. This is a key total cost consideration in your migration plan.

Two different methods of deployment might be:

- ▶ Using the existing enterprise deployment method or a new tool to deploy based on partition or disk images

- ▶ Using the distribution's automatic installation method extended with custom built packages for the locally customized client

Image-based deployment

Choosing an image-based deployment has an impact on the design of the client. The image can either be deployed using existing enterprise deployment methods or using a tool such as partimag (an open source version of the popular Symantec Ghost tool):

<http://sourceforge.net/projects/partimage/>

When the client is going to be deployed as an image, less care has to be taken when building the client to make sure that every customization is caught in a package (usually an RPM). However, extra care goes into creating an overall personalization script or method to change settings after the image has been deployed to make that client unique and configured for the particular user.

Install-based deployment

Several distributions have a mechanism for automatic, unattended installations. The enterprise distributions each have their own. Red Hat uses kickstart, which essentially uses an answerfile, called the *kickstart* file, to control Red Hat's installer application, anaconda. Novell has autoyast, which also has a configuration file to automate responses to the installation procedure. Both these methods are based on the RPM package format. These mechanisms can be extended with "homegrown" RPMs and have the option of running post-installation scripts after all RPMs have been installed.

Using the install-based deployment method (usually based on RPMs) means that all extra software and configurations have to be included as RPMs as well. This has to be identified in planning before building the Linux client, so that each application that is needed can be built into a supporting RPM package for delivery and installation.

4.8.2 Update deployed clients

Depending on the method chosen to deploy the clients, the method for keeping existing clients up-to-date might change as well.

Image based deployment

When using images the update scenario is twofold. The image has to be updated for fresh installs and a method has to be conceived to update existing clients.

If all software on the client is installed in the form of RPMs, updating existing clients boils down to updating those RPMs that are needed. However, if parts of

the client are not installed from RPMs, updating those parts becomes more challenging.

Install-based deployment

In the case of install-based deployment, the fresh clients are kept up-to-date by installing the latest versions when deploying a new client. Since all software is delivered as an RPM it is fairly easy to update software.

Enterprise distributions have management tools to handle this update process, as discussed in Appendix B, “Using enterprise management tools” on page 255.

4.8.3 Personalization of deployed clients

Once a client is deployed, it needs customization and personalization. The customization part can be handled by post-deployment scripts and can include:

- ▶ Hostname
- ▶ IP address
- ▶ Specific remote filesystem mounts
- ▶ Specific printers
- ▶ Specific applications

The personalization part is usually input with user data of the individual user using that client. This can either be handled the first time the client is booted through a mechanism such as firstboot or through a script or application that the user has to run once logged in.

The firstboot mechanism is an application that is run once (at first boot) and can personalize the desktop client. After running, a parameter is set in a file to prevent firstboot running again. The advantage of this mechanism is that this runs before logging on and means the user does not need an account and the account can be created during the process using data the user provides.

When the firstboot mechanism is not available or the user uses a central login account already available, personalization can be done by executing a script or application that personalizes the desktop. This might require logging off and on again or even a reboot.

Whether a user’s data and personalization information is stored locally or in the network has an impact on the method to choose.

This section so far has described how to deploy Linux to users without incorporating their settings from the old client. Tools exist that help to automate the migration of many common sets of personalization data from Windows to

Linux. One of these tools is described in detail in Appendix C, “Automating desktop migration using Versora Progression Desktop” on page 277.

4.9 Post-migration troubleshooting and technical support

After migrating a group of clients to Linux, the clients will have to be supported—not just in the normal day-to-day operations, but also for problems arising from the migration itself.

Not all of the support is of a technical nature. The usage patterns on the Linux client will be different from what users were used to on their previous clients.

The administration of a Linux client also needs a new methodology for troubleshooting. Finding and solving problems under Linux is very different from Windows. Administrators have to adapt to the new troubleshooting methods for Linux clients. The first step in this process for the Linux client is not a reboot, which is most often the first step on a Windows client.

4.9.1 What to expect

To prevent a lot of post-migration troubleshooting, the users have to be prepared for the new Linux client. The expectations for the Linux client have to be managed very carefully.

Changes for the user in migrating to the Linux client might be:

- ▶ The user no longer has full control over file system contents.
- ▶ Even when the desktop presents a similar look and feel, the process of changing the desktop settings is completely different.
- ▶ Right-click menus are either gone or present different options.
- ▶ The look and feel of favorite desktop applications has changed, or those applications have been entirely replaced with functional equivalents.

Most of these changes can be managed and anticipated by providing proper training and enablement materials. Remember, to minimize the amount of transition support, try to follow the guide for using “Bridging Applications”, as described in 3.4.1, “Bridging applications can separate retraining from migration” on page 58.

4.9.2 How to handle the unexpected

While it is possible to fully prepare the user in advance, some problems will still arise post-migration and during normal operation. Most of these problems are tasks for which support staff and administrators can be prepared and trained.

To tackle the unexpected problems, the support staff has to use a Linux-oriented approach to problem solving. To enable support staff to become used to this different approach, the differences have to be investigated.

In general, Windows operating system methodology for troubleshooting usually starts with:

- ▶ Rebooting the system
- ▶ Checking the event log
- ▶ Checking drivers and installing the latest versions

The Linux operating system has more easily identifiable modules. Also, there are lots of log files. This means that the methodology for troubleshooting starts with:

- ▶ Identify the affected module of the Linux distribution.
- ▶ Check all involved system logs starting with the syslog.
- ▶ Verify that configuration files do not contain errors or faulty settings.
- ▶ Restart services involved.
- ▶ Check errors generated by restart.

When using a Windows client, a reboot will solve more than half of the problems; when using a Linux client, the problem usually remains after a reboot. This means that support staff has to learn how to approach the problem on the new client.

Generally, the best way to prepare support staff for Linux client problems depends on which level of support is given. In the case of helpdesk or level 1 support, the best training is using a Linux client. When the support moves more toward administrative tasks, basic Linux (or even UNIX) training will have the best effect.

4.9.3 When to contact vendor enterprise support

Vendor support is usually contacted when the problem is found to be a bug or when software is not acting as stated in specifications or manuals. If the problem is not related to a vendor-specific part of the Linux operating system or suite of applications, it is also possible to find a solution or a fix in the open source community.

The decision for searching community support has to be made for each issue. When a support contract with an enterprise distribution vendor is in place, it is

best to explore that avenue first, because trying to incorporate a community fix in a vendor distribution might lead to new problems and jeopardize your support agreement with the vendor.



Linux architecture and technical differences

In this chapter, we illustrate differences by presenting a top to bottom view of the desktop Linux software stack, the dynamics of the free software and open source movements, the definition of a “distribution”, the importance of standards, and then sections related to specific technical differences.

- ▶ 5.1, “What is Linux” on page 110
 - Free software and open source, distributions, and standards
- ▶ 5.2, “Technical differences” on page 112
 - Components of the Linux operating system and a low to high-level discussion of the major operating system functional layers

5.1 What is Linux

Linux is an operating system that was initially created by Linus Torvalds. He began his work in 1991 when he made available Release 0.02 of the kernel through a newsgroup on the Internet. Development continued, driven by a loosely coupled team of programmers and release 1.0 of the Linux kernel was made available in 1994.

The kernel's main role is to supply an interface to the hardware and peripherals. It also manages the memory and schedules execution of processes. A lot of additional software is required to provide all of the functions expected from a complete operating system. Fortunately, open source developers, including IBM, have contributed time and effort to put all of the necessary pieces in place.

Today, Linux is a fully functional, flexible, reliable, and free operating system that runs in many different hardware environments and on an extremely diverse set of architectures. Linux is all about the freedom to choose the features and configuration of your operating system.

Linux, both the kernel and associated software, is developed and distributed under the GNU General Public License (GPL) and its source code is freely available to everyone. The GPL means you can share and modify the software without being charged for it and makes sure the software remains free of charge for everyone. For more information about the GPL, use following URL:

<http://www.linux.org/info/gnu.html>

For more information about the Free Software Foundation and the GNU project, refer to:

<http://www.fsf.org>

5.1.1 Distributions

There are hundreds of distributions (commonly called *distros*) of the Linux operating system, available from sources all over the world. A distribution is a organization that packages an installable set of components including the Linux kernel, and makes it available to a broad audience. Most distributions have their own feature set, and some are customized for special hardware configurations or for special purposes. Some well-known distributions are: Red Hat, Novell SUSE, Debian, Gentoo, and Mandriva. Although the operating system from these organizations is freely available, many also sell retail versions that offer greater levels of support should you need it.

What is a Distribution ?

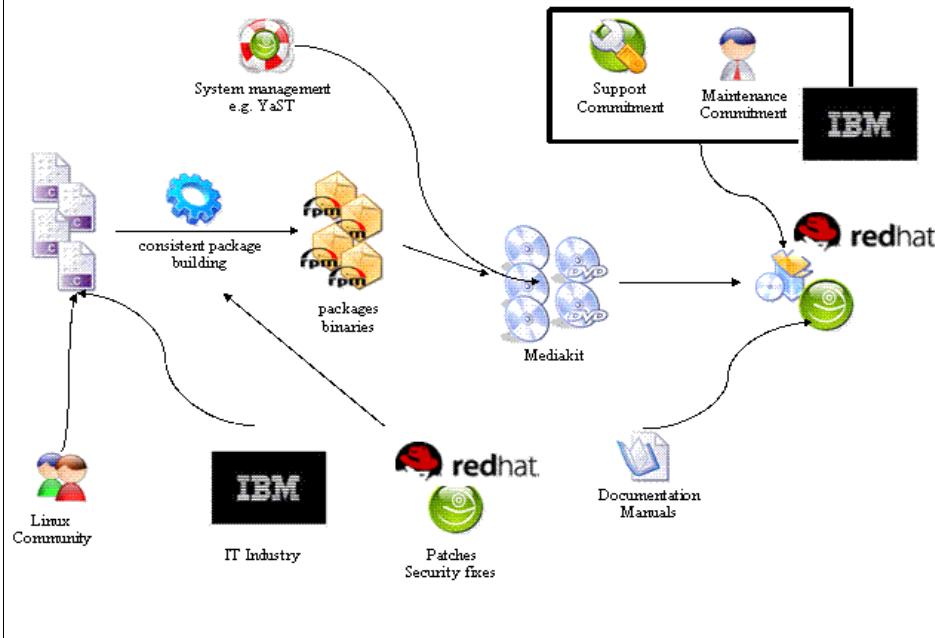


Figure 5-1 Distribution contents

Most of the latest Linux distributions offer the following:

- ▶ Full 32-bit and 64-bit architectures
- ▶ Preemptive multitasking
- ▶ Support for multiple users and groups
- ▶ Full networking capabilities
- ▶ Protected memory
- ▶ Clustering, including load balancing and failover
- ▶ Dynamic reconfiguration
- ▶ SMP support

5.1.2 Standards

Recently, a lot of effort has been put into standardizing Linux and higher-level subsystems. Organizations such as the Free Standards Group, freedesktop.org, and the Open Source Development Labs (OSDL) started working on many different projects in this area. These projects use and extend existing standards such as POSIX, the Single UNIX Specification, XML, DOM, CORBA, and many more. Some example projects are:

- ▶ Filesystem Hierarchy Standard
- ▶ Linux Standard Base at <http://www.linuxbase.org>
- ▶ Internationalization (OpenI18N)
- ▶ Printing (OpenPrinting)
- ▶ Accessibility
- ▶ Clustering (Open Cluster Framework)
- ▶ Data Center Linux
- ▶ Carrier Grade Linux

Note: The LSB 3.1 specification has been approved and published. See 1.7, “Desktop Linux futures” on page 7.

Most of these standardization efforts were focused on low-level functionality and standardization of the file system layout, a new printing architecture, stabilization or creation of APIs for clustering and high availability, and to make sure that programs from independent software vendors (ISVs) are binary compatible between different Linux distributions.

From a standardization point of view, the Linux desktop is currently a fast moving target. A lot of promising development is currently underway that will evolve the desktop Linux landscape even further. For example:

- ▶ New X server extensions and modularization
- ▶ Cairo (vector graphics system)
 - With X, OpenGL, and Postscript or PDF output
- ▶ HAL (hardware abstraction layer)
- ▶ GStreamer (streaming media framework)
- ▶ D-BUS (message bus system)

5.2 Technical differences

This section introduces and discusses many of the technical differences between the architecture of Windows and Linux clients.

Linux has innovations in technology and features that are not available on Windows. Linux is also highly modular and extremely configurable. And certain features have been implemented differently than they are implemented in the Windows operating system.

A Linux operating system consists of the kernel and several crucial components. Some of the terms used in this chapter might be new to someone coming from a

Windows environment. For more information about these terms, refer to Appendix A, “Linux glossary for Windows users” on page 233.

The Linux operating system contains the following components (as shown in Figure 5-2 on page 114):

Linux kernel	Provides a set of services or kernel features to applications allowing them to run. Device drivers and modules can be added to the kernel to expand these services. The kernel also supports concepts such as multitasking and a multi-user environment.
Devices	A device is an entity within the operating system that forms the interface to hardware. The device driver handles translation of operations on the device in the operating system to physical actions in the hardware.
Modules	A module is dynamically loadable or unloadable kernel code that provides a service, such as a device driver. Modules provide the ability for the kernel to interface with hardware, making the hardware functionality available for use by the kernel or user space applications. Once a module is loaded into the kernel, it is considered part of the kernel, but it can be unloaded when not needed anymore.
Application level (user space)	
	Environment where applications that make use of the kernel services run. This environment or user space can be a graphical or a command line-based user interface. An application has access to the kernel services through invocation of system calls.

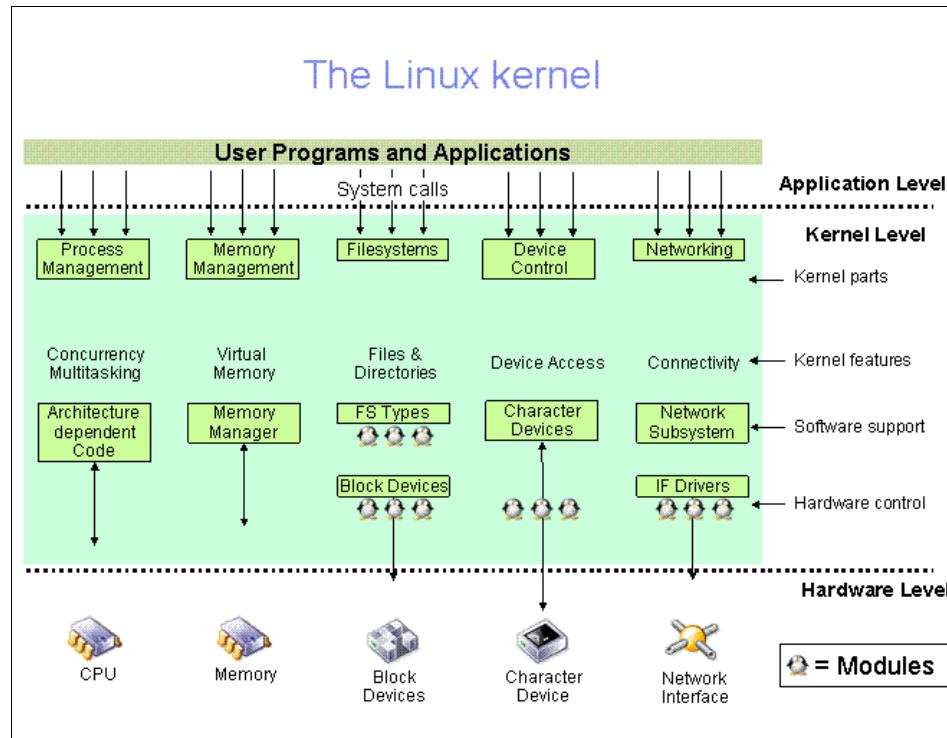


Figure 5-2 Linux Operating System

5.2.1 Kernel booting process

One of the biggest differences when migrating to Linux is its stability. It is conceivably possible to never need a reboot, unless of course, the kernel needs upgrading. However, when it is time to boot, the process is somewhat different than it is in Windows. The initial kernel booting process is also referred to as *kernel bootstrap*.

Kernel bootstrap process

The kernel bootstrap process is:

1. Processor initialization, only low memory addressed, BIOS starts bootloader (GRUB, LILO).
2. Bootloader loads initrd and kernel into memory; bzImage unzips itself to high memory (/boot).
3. Bootloader starts kernel, and bootloader tells kernel that there is an initrd.

4. Kernel unzips initrd and mounts it as root-filesystem.
5. Within initrd, a program called linuxrc is executed. The linuxrc program loads the modules necessary to mount various physical filesystem devices.
6. The real root filesystem is mounted; initrd is remounted to /initrd.
7. The init process starts (pid 1); init is the only process started by the kernel itself, every other process is a child process of init.

In the previous description, the term initrd indicates a file that is decompressed and used as an initial RAM disk (hence, the name). This initial RAM disk is the root filesystem as the kernel starts running. This initial root filesystem needs to contain modules and drivers for all devices needed. So if the actual root filesystem lives on an ext3 filesystem of a SCSI disk, the initial RAM disk needs both ext3 and SCSI kernel-modules.

5.2.2 Communication, files, and services

Communication in Linux is performed using sockets. A *socket* is a pseudo-file that can be read from or written to. This then corresponds to a receive of a send operation on the communication device. A network connection between two Linux systems, for example, is built from two sockets on both ends, each corresponding to an IP address and a port number.

Devices in Linux are also represented by files, usually in the /dev directory. Writing to the file corresponds to writing to the hardware device. Device drivers handle translation between this file-access and the physical action on the hardware.

As mentioned in 5.1.2, “Standards” on page 111, the Filesystem Hierarchy Standard (FHS) governs the Linux filesystem. Filesystems in most Linux distributions are set up along the lines of the Filesystem Hierarchy Standard (Table 5-1 on page 116).

Table 5-1 The Filesystem Hierarchy Standard

Directory	Description	Remarks
/	Root of the filesystem	All other files and filesystems are sub-directories of this.
/bin	Essential command binaries	Commands such as sh, ls.
/boot	Static files of the boot loader	Sometimes must be own filesystem.
/dev	Device files	
/etc	System configuration	Contents differ for each distribution.
/lib	Shared libraries and kernel modules	
/media	Mountpoint for removable media	Used to be part of /mnt.
/mnt	Mountpoint for temporary mounts of filesystems	
/opt	Add-on application software	
/sbin	Essential system binaries	
/srv	Data for services	
/tmp	Temporary files	
/usr	Secondary hierarchy	Contains an entire subtree, governed by FHS.
/var	Variable data	Logging, PIDs, caches, and so on.

A more detailed description of the FHS can be found at:

<http://www.pathname.com/fhs>

A big difference with the Windows OS is that the concept of driveletters does not exist within Linux. This is an important difference when discussing desktops, because users are used to this concept of the driveletter (C: containing the Windows OS and Program Files, D: usually containing data, E: being the CD-ROM, and so forth).

5.2.3 Multi-user

The way in which users are used in Linux is based on the model of UNIX. Since UNIX was developed as a multi-user operating system from the start, the concept of multiple users is very tightly bound to the workings of the operating system. The Windows operating system originally started as a single-user, single-thread operating system. Since then, Windows has come a long way. But because of this difference in history, there are inherent differences between the way multiple users are handled by both operating systems.

There are several different ways to look at how good an operating system handles multiple users:

- ▶ Can multiple users run processes at the same time?
- ▶ Can multiple users be logged on at the same time?
- ▶ Can applications be installed for multiple users?
- ▶ What are the rights of the different users?

Note: Useful's multi-station client computing solution is an excellent example of how to leverage the multi-user flexibility inherent in the Linux OS and the X architecture. See 7.5, "Multi-station client architecture" on page 162 and Appendix D, "Multi-station computing deep dive using Useful Desktop Multiplier" on page 289 for more details.

Can multiple users run processes at the same time

One way to look at multi-user is that more than one user can run processes at the same time. In Linux, usually all system processes run as the root user. Any number of users can have processes running on a Linux system at the same time. The number can be limited by specific tables in the kernel, but in essence the code enables any number.

In Windows, it is also possible to have processes running for multiple users at the same time. This property is the basic building block for the Windows Terminal Server.

Can multiple users be logged on at the same time

In the case of both operating systems, the answer to this question is yes. Multiple users can be logged on at once.

If a login consists of having a console with a graphical display, this is also possible in both operating systems. In Windows, this is done using Terminal Server and a Remote Desktop Protocol (RDP) connection. In Linux, this can be done by allowing the session manager to create screens on remote Xservers.

A multi-station client architecture provides an excellent example of how a Linux-based system can support multiple user logins, and in fact also support multiple, independently connected KVM (keyboard-video-mouse) connections to a single system. See 7.5, “Multi-station client architecture” on page 162 and Appendix D, “Multi-station computing deep dive using Useful Desktop Multiplier” on page 289.

Can applications be installed for multiple users

If different users want to run an application at the same time, it is extremely useful if each user has individual settings for the application.

In Linux (like UNIX), this is usually handled by “dot-files” (filenames starting with a period) that appear in the user’s home-directory, or by environment variables that are set for the specific user.

Originally on Windows, settings were stored in the system registry, which was originally only created for a single user. This situation sometimes still creates problems in the Windows operating system, but most modern applications on the latest incarnations of the Windows OS handle user settings differently. More and more settings are now stored in files in the “Documents and Settings” structure in Windows.

What are the rights of the different users¹

In Linux, the user with all rights (that is, the superuser) is the user “root”. Many lower level system services in Linux run “as root”, meaning that the process owner ID is root and therefore the process itself is running with root privileges, for example, the system initialization process ‘init’ (always process ID 1). The root user cannot be renamed and, therefore, cannot be more than one root user.

In Windows, the Administrator user is the user with all rights. There can be multiple instances of users defined with equivalent Administrator access levels, and the Administrator user can be renamed to something else.

You could say that Linux is a “single-superuser OS” and Windows is a “multi-superuser OS”. And where multi-user is good, it does not seem a good idea to have multiple superusers, with free-to-be-chosen names.

5.2.4 Graphical and text-based environments

Microsoft Windows is inherently a graphical system. Windows will always boot into a graphical mode, even on servers which do not need a graphical environment for daily operation. Most applications developed for Windows are

¹ Note: At the time of this writing, behavior of the security systems in Windows Vista is not known. In the future, Vista might have different ways of managing administrator level access to the system.

built with a GUI first, with command line automation a lower priority. Proponents say that these graphical tools make Windows easier to use, though detractors claim that the lack of command line options makes it hard to automate tasks on Windows.

As we have stated already, Linux is an extremely modular operating system. Even though most distributions of Linux configure a graphical environment by default, it is always possible to boot Linux solely into a command line interface. In fact, this is a common practice for “headless” servers, that is to say servers which have no display attached and will only be accessed remotely. As such, not loading a graphical environment can save on resources such as memory and disk space. Many system level tools for Linux (such as networking configuration or application packaging tools) are built with a command line interface first. Later, graphical tools are built to wrap the command line interface. This results in a multitude of different graphical “front end” applications that implement the same features provided by the console (command line equivalent) applications in a GUI-based equivalent.

The application structure for graphical environments in Linux is also extremely modular. There are several layers of abstraction provided by different applications, with many choices available in each layer. Table 5-2 demonstrates many of the choices that are available at each layer of the user interface.

Table 5-2 Layers on top of the X Window system

Layer	Choices
Display manager	XDM, GDM, KDM, and Entrance
X Window Server	XFree86 or X.org
Window managers	FVWM, IceWM, WindowMaker, Metacity, KWin, and Enlightenment
GUI toolkits	OpenMotif, GTK+, Qt, wxWidgets, FLTK, FOX, Swing, SWT, and EFL
Desktop environments	KDE, GNOME, GNUStep, XFCE, ROX, Looking Glass, and Metisse

This level of modularity (the ability to independently start, stop, and modify the behavior of the X Window system on a Linux-based computer) is what allows for the success of many-to-one client computing solutions such as Useful’s multi-station Desktop Multiplier. See 7.5, “Multi-station client architecture” on page 162 and Appendix D, “Multi-station computing deep dive using Useful Desktop Multiplier” on page 289 for more details.

In the case where you want a graphical environment to start at the end of the boot process, display managers can be set up as an additional service that is started up. The display manager starts an X Window Server and communicates with that server to display itself. The display manager normally also provides a login screen for a desktop and allows users to choose which window manager or desktop environment to use for their session if multiple choices are provided.

The ability to chose one of the multiple display management environment options (that is, GNOME, KDE, and so forth) during the graphical login process significantly demonstrates the modularity and flexibility of Linux relative to Windows.

The X Window system is essentially a client/server application. The labeling of the client and server components is at first non-intuitive. What would normally be seen as the client application (running on the local workstation or desktop) is called the X Server and the server applications (which can run remotely) are called X clients. The X-Windows architecture is shown schematically in Figure 5-3 on page 121. The X Window Server program has control over the display and mouse+keyboard. The X client programs (in this case, Xterm and Xclock) run either on the local machine (machine A) or on a remote machine (machine B) and communicate with the X Server through sockets.

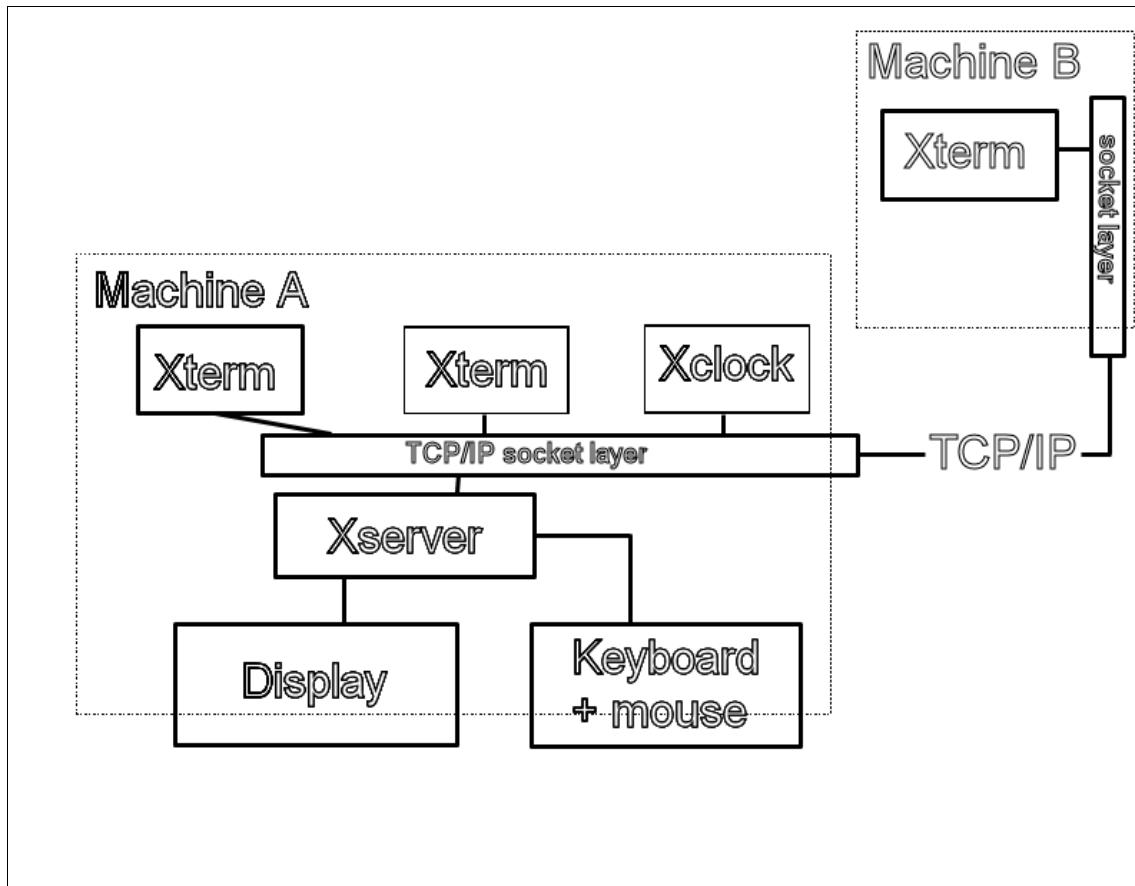


Figure 5-3 Schematic X-windows architecture

The X Window Server handles drawing images on the screen, using the correct video card and monitor drivers. The X server also handles receiving input from devices such as a mouse and keyboard. X Window Servers follow a standard interface, called X11, so that any X server can receive input from an X client. This interface works over network sockets, even on the local machine (it is important to note that local sockets are not limited to network speeds though). This abstraction allows for X forwarding, where an X client on one machine can display on an X server on a different machine, with no extra implementation necessary for the X client.

Window managers are X client programs that control how other X clients are positioned, resized, or moved, see:

<http://xwinman.org>

They can also provide title bars and other decorations to windows, handle window focus, and provide user-specified key and mouse button bindings. Example 5-1 on page 122 shows how to start a nested X server with **Xnest**, which is an X server that is simultaneously an X client. This allows you to test other window managers or desktop environments in a window.

Example 5-1 Nested X server with Xnest

```
Xnest -fp `xset -q | grep fonts` :1 &
xterm -display :1 &
```

To simplify graphical programming, several GUI toolkits have been developed on top of the base X11 libraries. Most GUI toolkits also provide a theming engine, and some base libraries allow for a similar look and feel between applications. There are also many integrated development environments (IDEs) available that further simplify GUI programming:

- ▶ KDevelop and Qt Designer for KDE and Qt
- ▶ Anjuta and Glade for GNOME and GTK+
- ▶ MonoDevelop for Mono
- ▶ Eclipse or NetBeans™ for Java with Swing or SWT

Desktop environments provide a much richer user environment than just simple window managers by providing standard productivity applications such as:

- ▶ E-mail and calendaring
- ▶ Instant messaging
- ▶ Image manipulation
- ▶ Document editing and viewing
- ▶ Scanning
- ▶ CD and DVD burning

These applications are created with a standard GUI toolkit. They have a homogeneous look and feel, theme flexibility, inter-application communication, drag and drop functionality, session management, and virtual desktop management.

Most recent Linux development activity is beginning to standardize around the GNOME and KDE desktop environments (and their respective toolkits of GTK+ and Qt). Some traditional UNIX vendors are also beginning to adopt these environments.

5.2.5 System runlevels

Linux operating systems have the ability to boot into multiple runlevels. A runlevel is a configuration option which defines which processes should be allowed to run, and at what level. This is somewhat analogous with booting Windows into

Safe Mode, though much more granular and configurable. While it is technically possible to boot into eleven different runlevels (the numbers 0 through 9, and also S), only runlevels 1, 2, 3, and 5 are commonly used. (Runlevel 0 is reserved for system shutdown and runlevel 6 is reserved for system reboot. Runlevel S is a special runlevel used by scripts before entering the first runlevel.) By default, the runlevels are laid out as you see in Table 5-3.

Table 5-3 System runlevels and modes

Runlevel	Mode
1	Single user text mode
2	Multiuser text mode without networking
3	Multiuser text mode with networking
5	Multiuser graphical mode with networking

The default runlevel is configured in the file /etc/inittab. With root level access, you can switch between runlevels using the `init` command.

5.2.6 Drives, partitions, and file systems

Unlike Windows, Linux does not treat drives as separate top-level items in the system device hierarchy. Instead, everything is mounted into a file system hierarchy. For instance, the main hard drive is usually mounted as /, while CD drives are mounting as /mnt/cdrom (or /media/cdrom on some distributions). Both methods have advantages and disadvantages. On Windows, it is obvious to tell which files are on which drive. With Linux's format, a faster drive could be used for the system mount point (/), and a larger yet slower drive could be used for home directories (/home), all done without the user noticing the difference.

Similar to mapping a drive to a file share in Windows, Linux can mount network shares (using NFS, SMB, or more) into the current file system, as seen in Example 5-2. It is even possible to mount the users' home directories over the network, in a setup similar to Windows' roaming profiles. When mounting Windows shares (or Samba shares), it is import to note that the SMB protocol does not allow for symbolic links or sockets, and, as such, is not suitable for home directories.

Example 5-2 Mounting a Samba share

```
smbmount //itsont05/data /mnt/itsont05_data
```

There are many different file systems available for Linux, such as ext2, ext3, ReiserFS, XFS, and JFS. Each file system has reasons for and against its use,

though all are mature. Red Hat defaults to using ext3, while Novell Linux Desktop uses ReiserFS. Both ext3 and ReiserFS are journaling file systems. A *journaling file system* keeps an independent log (or journal) of data that is stored. Most Linux file systems also do not need to worry about fragmentation. The file systems work around most of the problems caused by a fragmented hard drive, so that even if a drive was fragmented, it would not cause any significant speed decrease.

Linux also has built-in support for the FAT file system used by DOS and Windows. NTFS, the file system introduced with Windows NT, is currently supported in read-only mode. There is an experimental driver for writing to NTFS, but because this can cause data loss, we do not recommend it for real use.

5.2.7 Virtual memory

Both Windows and Linux have a system for virtual memory. The model behind both is different and can even change from one version to the next.

One difference between Windows and Linux is the way swapping or paging is done. A Linux system will only start paging out memory pages when no physical memory is available for allocation. Paging in occurs when memory pages are needed that are no longer in physical memory. Performance can be adversely affected when there is a lot of paging activity especially when there is a lot of paging in and out. This generally means physical memory is too small, or certain kernel parameters governing virtual memory might be set incorrectly, or possibly, the size of the paging space is not optimal. Windows will sometimes perform paging out of certain memory pages even when physical memory is not exhausted. This is called *preventive paging*. This does speed up freeing up memory pages when the need arises, but it also means that Windows will be paging out more often.

Another difference is that Windows usually uses a swapfile, while Linux uses a swap partition (but can use swapfiles as well). On Windows, the main drive C: usually contains a file named pagefile.sys, and that file is used for virtual memory. The file can grow in size if necessary, however this can lead to the swapfile becoming fragmented, which significantly hampers performance. On Linux, a swap partition is usually created on installation. This partition is set aside for virtual memory use, and is always contiguous. However, because it is a partition, it can never grow larger than the original size to which it was initialized during partitioning, without repartitioning the disk.

Tip: To more efficiently utilize disk space on a dual-boot system, there are ways to make both Windows and Linux use the same partition for managing the virtual memory. More detail about this can be found at:

<http://www.tldp.org/HOWTO/Swap-Space.html>

It is also possible to specify multiple swap partitions or swapfiles on the same Linux machine. This might be necessary if the swap partition created on installation was too small. Unlike Windows, which will automatically allocate more virtual memory, partitioning the drive and allocating it are tasks that a system administrator must perform manually.

Tip: Using separate swap partitions or swap files that are resident on each of the physical hard disks installed in the same system usually has a positive impact on overall system performance.

5.2.8 File links

Unlike Windows, Linux supports shortcuts, called *links*, at the file system level. (Technically, NTFS also supports links, but Microsoft Windows does not use them.) Linux uses two types of links, symbolic links and hard links.

Symbolic links, or symlinks for short, are direct references to another file or directory. A symlink can be created anywhere and can point to a file or directory on any other partition or mount point. Similar to Windows shortcuts, if the original file pointed to is moved or deleted, the symlink will no longer function.

In order to understand hard links, it helps to understand Linux file systems. A file path on Linux is really just a pointer to an object called an *inode*. The inode contains all the information about where a file is stored and how large it is. When creating a hard link, another pointer to a file's inode is created. Only when all of the pointers are deleted is an inode removed. Therefore, if the original file is moved or deleted, the *hard link* still points to the original inode, and thus the file still exists. Hard links, however, cannot point to directories, nor can they point to a file on a separate partition.

5.2.9 Replaceable kernel

Unlike Windows, the Linux kernel can be treated as another modular component of the system. It is possible to have multiple kernels installed on a machine. For instance, you might have the distribution installed kernel by default. However, for instance if that kernel does not support certain new hardware or an optional device, you have the option to download a newer version of the kernel and

compile it to support the additional device. It is then possible to configure the bootloader to display an option to boot either kernel; the default kernel as well as the one which has the new device support functions compiled into it. While this ability is useful for testing out new hardware support or new kernel features, most users do not need to do this. However, it is important to point out that this is another feature of the modular nature of the Linux OS.

Not all distributions give you the ability to choose which kernel to boot out of all the kernels installed. This depends on how the Grand UNified Bootloader (GRUB) is configured. For instance, Novell SuSE enterprise distributions usually replace the kernel in the bootloader automatically with the last installed kernel. For more information about GRUB and how you can manage boot level access to multiple kernels in Linux, see the following sites:

<http://www.gnu.org/software/grub/>
http://en.wikipedia.org/wiki/GNU_GRUB
<http://www.tldp.org/HOWTO/Multiboot-with-GRUB.html>

5.2.10 Device drivers and hardware support

Windows is known for its hardware support and device drivers being available for almost everything. This is based on manufacturers developing drivers for their hardware for Windows first (and sometimes also last). Because not all manufacturers supply Linux drivers, not all hardware can be used under Linux.

However, the open source development framework and the modular structure of the Linux operating system enable everyone to write device drivers. This generally means that as a piece of hardware (that is not supported by the manufacturer of Linux) becomes more popular, an open source device driver appears after some time.

The number of manufacturers now supplying source code for their drivers is increasing. The advantage of providing source code is that the driver can be compiled for any version of the corresponding kernel that is available. The support is usually limited to a number of distributions' kernels for which the manufacturer has tested the driver.

5.2.11 Font support

As we mentioned in 5.2.4, “Graphical and text-based environments” on page 118, the Windows OS has a fully integrated display driver. This includes font support. The Windows OS has many fonts bundled, some of which are copyrighted by Microsoft.

In the case of Linux, the display is driven by the X Window system. Recent incarnations of the X Window system have a separate font server. This font

server can even be used by other systems. Lately, a number of nicely rendered fonts have become available for Linux, either in the public domain or for a small license fee. Novell Linux Desktop and Red Hat Desktop each include fonts from suppliers that ask for a license.

5.2.12 64 bit and multi-core support

Up until recently, 64-bit support was not relevant for desktop operating systems. Lately, the processors used in desktops have not only gone to 64 bit, but also to multiple cores.

Multiple cores in the CPU are handled both by Linux and Windows as multiple CPUs. They are handled in a way similar to how hyperthreading in certain types of CPUs resulted in multiple logical CPUs that are apparent to both Windows and Linux.

Linux has been supporting 64-bit processors for a long time, predominantly because Linux is also available on processors that have been 64 bit for a while (for example, IBM Power architecture). Recently, Windows operating systems have been released in 64-bit versions as well.

The key thing to remember when using a 64-bit version of an OS is that not all 32-bit applications run smoothly, and 32-bit applications cannot always find the correct libraries. Also, drivers cannot be mixed when using 64-bit operating systems. A 32-bit driver in a 64-bit OS can lead to problems.



Part 3

Performing the pilot migration

Part 3 of this book includes:

- ▶ Chapter 6, “Migration best practices” on page 131
- ▶ Chapter 7, “Client deployment models” on page 139
- ▶ Chapter 8, “Client migration scenario” on page 173
- ▶ Chapter 9, “Integration how-tos” on page 195



Migration best practices

In this chapter, we describe several best practice methods you can use in your own Linux client migration projects. The topics we cover include situations which can make a Linux client migration easier and also the use of third-party tools to automate specific migration tasks.

The sections in this chapter are:

- ▶ 6.1, “The transitional desktop” on page 132
- ▶ 6.2, “Choose an installation methodology” on page 132
- ▶ 6.3, “Centralize data locations” on page 133
- ▶ 6.4, “Break down migration into manageable groups” on page 134
- ▶ 6.5, “Minimize impact of down time” on page 135
- ▶ 6.6, “Get user feedback” on page 136
- ▶ 6.7, “Automate the migration” on page 136
- ▶ 6.8, “Use a systems management tool” on page 136
- ▶ 6.9, “Do not migrate until you are ready” on page 137
- ▶ 6.10, “Do not just migrate, upgrade” on page 138

6.1 The transitional desktop

As seen in 3.2.1, “Bridging applications” on page 53, many open source applications run on Windows as well. Migrating to a Windows desktop with a few open source applications is an easier transition than replacing the entire operating system and application stack in one jump. Also, you can gain many of the benefits of using open source tools, such as protection from viruses and spyware, enhanced customizability, and compliance with open standards. Once users are comfortable with the open source application stack, a migration to a Linux desktop running the same applications is far less intimidating.

You might even choose to stage the rollout of open source applications into several steps. For instance, you could start with a migration from Microsoft Internet Explorer to Mozilla Firefox. This involves making sure all intranet and important Internet sites are compatible with Firefox. After this migration, you could switch from Microsoft Outlook or Outlook Express to Mozilla Thunderbird for e-mail. However, because Thunderbird does not include calendaring or Exchange support, you might not be able to implement this change. Finally, you could switch from the Microsoft Office suite to OpenOffice.org, converting any macros or incompatible documents to the new format. Once all users are comfortable with the new applications, switching to a Linux distribution which provides support for Firefox, Thunderbird, and OpenOffice.org is less difficult for users.

6.2 Choose an installation methodology

There are several options available for rolling out Linux desktops. Each option has advantages and disadvantages, and you should choose the methodology that is most appropriate to your organization.

6.2.1 Wipe and Reload

The most obvious installation method, Wipe and Reload involves removing the current Windows installation and installing Linux on the same hardware. This method gets users over to Linux immediately and requires no extra hardware, but can be dangerous if some important data on the Windows installation is not backed up before reloading the machine.

6.2.2 Dual boot

A dual boot installation installs Linux side-by-side with Windows, and allows the user to choose either operating system at boot time. Many distributions can resize Windows partitions during an installation, and they can be set up to read

and write from the main Windows partition. This option allows users to return to Windows if necessary, for example, to use an unmigratable application or to use a more familiar application for a time-sensitive task. It also guarantees that the user's data is preserved and accessible on the Windows partition. However, dual-booting requires maintaining two operating systems with current patches and anti-virus systems. Also, certain users might continue to boot into Windows as their main operating system, without even trying to learn Linux.

6.2.3 Hardware refresh

The migration to Linux desktops could be timed with your organization's next hardware upgrade. When the users get their new hardware, it also comes with a new operating system. As with a dual-boot scenario, the old desktop could be left with the user for a time, to run unmigratable applications or to use a more familiar application for a time-sensitive task. Or, the old desktop could be kept only for backup purposes, in case some data was not migrated to the new desktop. If no hardware refresh is planned for a significant amount of time, this option might not be suitable for your organization.

6.2.4 Hardware round-robin

Similar to the hardware refresh option, in a hardware round-robin, the user is delivered a new computer with the new operating system installed on it. After the user is comfortable with Linux, and all data is definitely recovered from the Windows desktop, then the old desktop can be wiped and used as a new Linux desktop for some other user. This option delivers many of the benefits of a hardware refresh, without requiring purchasing new machines for all users. However, you need as many new machines available as you have users being simultaneously migrated. Also, it works best if your hardware is interchangeable between users. Users receiving noticeably slower machines might not welcome such a change.

6.3 Centralize data locations

The most important personalization on a user's desktop is the user's data. Whether that data is word processing documents, presentation files, or e-mail, users need that data to perform their day-to-day jobs. By centralizing the storage of said data, migrations can be made easier and more reliable.

6.3.1 Central file server

There are many advantages to using a central file server. Easier and more reliable backup procedures, more cost-effective use of storage space, easy

collaboration between users, and data migration are unnecessary. Connecting desktops to a file server can be as simple as mapping a drive to a Windows share (perhaps using a Linux server running Samba) and requiring all users to keep their data in that directory. Laptops are more difficult, because users want their documents available when they are away from the office. Using a synchronizing system such as Novell's iFolder can solve the problem. If a user's files are all stored in the network server, you only need to connect the new Linux desktop to the same server, and all of the user's documents are available on the new machine.

If for some reason files cannot be stored in the network, then it helps to implement a policy which requires files to be in a specific directory structure on each machine. For instance, users could be told to keep all of their documents in the My Documents directory. Then, during a migration copying the My Documents directory to the new Linux desktop ensures that all the necessary documents are on the new machine.

6.3.2 Central mail server

Just like storing files in the network, using a central e-mail server provides many of the same benefits. There are numerous open source mail servers which provide IMAP access. However, these servers only centralize e-mail and not the storage of contacts, calendaring, and tasks. To centralize these features, a groupware server is required. There are still many choices from open source servers such as Open-Xchange, OpenGroupware, and the upcoming Hula project or from commercial servers such as Novell Groupwise, Scalix, or even Microsoft Exchange. Not all groupware servers support all e-mail clients, so more investigation into the available options is required. Once the user stores all e-mail, contacts, and calendar information about the groupware server, the migration of this data only requires setting up the account information about the new Linux desktop.

6.4 Break down migration into manageable groups

Replacing the operating system and applications on every computer in an organization is a daunting task. Not only is it usually not possible to replace every machine in one step, it is certainly unwise. Before a migration, you should break the task of migration into smaller, more manageable tasks. It makes sense to start with the easiest groups to migrate, then continue on down the line. For instance, you might want to move your call center (which only requires basic e-mail and Web access) first, after that, the sales team (which also requires an office suite), and then marketing (which requires graphics tools). You probably want to break each of those groups into smaller groups, such as by project or by

location. It is important to ensure that each of the groups that have been migrated can still interoperate with those that have yet to be migrated. If two groups, which need to work with each other, cannot work together during a migration, then there is unnecessary animosity directed toward the Linux desktop, and the migration project. Instead, either combine the two groups into one, or find a solution to their interoperability needs. This might require the use of a migration to a transitional desktop setup, which was described in 6.1, “The transitional desktop” on page 132. A sample breakdown is shown in Figure 6-1.

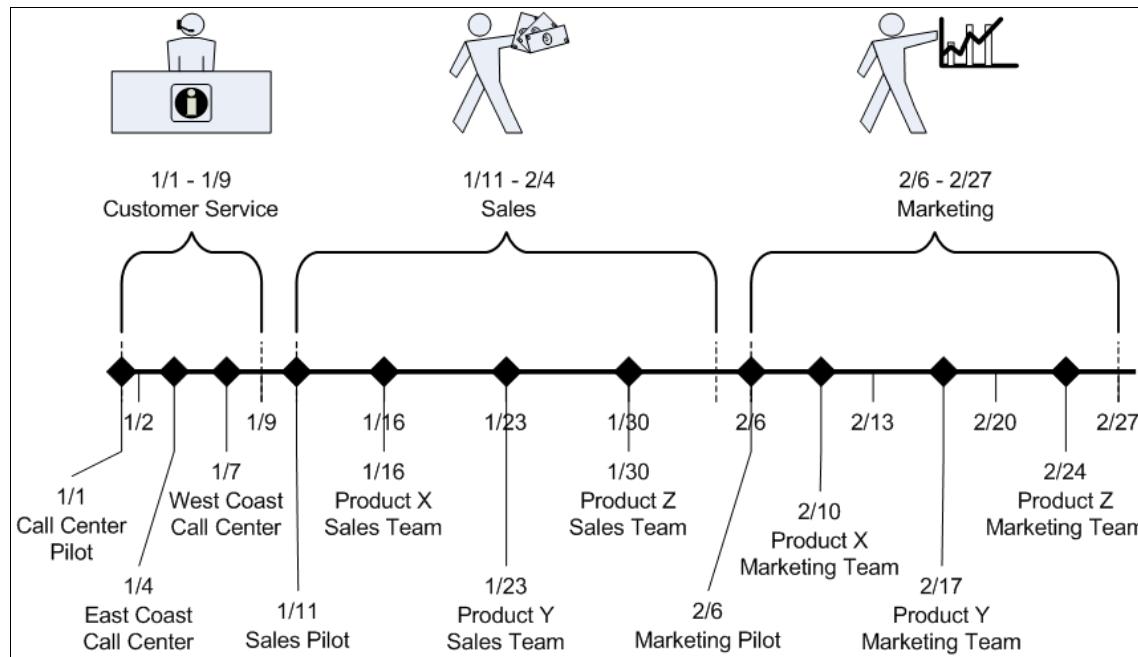


Figure 6-1 A sample migration time line

6.5 Minimize impact of down time

One of the most expensive results of a migration is down time. Any time that users are not able to perform their job duties is lost productivity that costs the company money. While there is never a perfect time for a disruptive migration, minimizing the amount and impact of down time is key. One option is to do migrations during the evening or weekend off-hours. This might require paying overtime or other compensation to the IT staff, but it can severely minimize the outages. Depending on the size and skill set of the group migrating, the migration could also be performed during an all-day training session. This way, the users could return from the training session to start using their new Linux desktops.

6.6 Get user feedback

Before migrating a group of users, select a representative minority to give feedback on the proposed Linux desktop solution. Ensure that the users will be able to perform all the necessary tasks for their jobs under the new operating system. Try to see if the users can perform these tasks without help. Listen to the problems that the users face and try to solve the underlying problem. For instance, users might complain that the new desktop cannot perform certain tasks, when, in reality, they just do not know how to perform the task or solve the problem. Use what you learn from the users' problems to either fix the desktop solution or to tailor the training classes to focus on problem areas.

After migrating each group, learn from any mistakes that happened, and plan better for next group. You might decide to offer more in-depth training or to change your installation methodology.

6.7 Automate the migration

Users have an easier time adjusting to the new operating system if their data and personalized settings have been applied to the new machine. Custom scripts can be written to copy data from the users' My Documents and Desktop directories and to set up some of the settings on the new desktop. However, there are hundreds to thousands of settings that users can have customized on their desktop. There are commercial tools, which can automate migration of system settings such as the user's wallpaper or screen saver settings, and application settings, such as e-mail messages and accounts, Internet bookmarks and home page, custom dictionaries in word processor programs, or instant messaging accounts. These tools also usually handle the migration of data, either from specified directories or the ability to search by file type over the entire hard drive. More information about a tool like this, Progression Desktop from Versora, is available in Appendix C, "Automating desktop migration using Versora Progression Desktop" on page 277.

6.8 Use a systems management tool

Systems management tools, such as IBM Tivoli Provisioning Manager¹, Red Hat Network², and Novell ZENWorks Linux Management³ provide features such as automated installation of software, or even full operating system image deployment through provisioning modules. Many systems management tools

¹ <http://www-306.ibm.com/software/tivoli/products/prov-mgr/>

² <http://www.redhat.com/rhn/>

³ <http://www.novell.com/products/zenworks/linuxmanagement/>

can also be used to schedule the execution of custom jobs, such as running an automated migration tool on multiple machines at once (an example of this is described in Section 6.7, “Automate the migration” on page 136). Some studies have shown that by leveraging a combination of an enterprise systems management tool and an automated migration tool, one technician can complete migration of one hundred or more desktops in a single day, as opposed to much fewer if the process was managed manually.

Important: For medium to large enterprises, you can expect that the business case for a migration cannot be entirely justified by a potential just for cost savings in operating system and application software licensing fees. In fact, in a client migration scenario, you should expect that the most important cost consideration that will come into play in medium to large enterprises will be the overall cost of *managing* the client computing infrastructure on an ongoing basis.

As the size of the supported client base scales up, these costs (IT staffing, support and help desk, system upgrade and replacement, repair and protection from viruses and other attacks, and so forth) greatly overshadow the costs of recurring operating system and application software licensing.

This is why we strongly recommend that the approach for building a cost justification model for a client migration to Linux should focus on the potential for introducing innovative ways to achieve savings in how enterprise client platforms are managed. The integrated management tooling suites provided by strategic Linux IBM Business Partners, such as Novell and their ZENWorks Suite and Red Hat and their Red Hat Network, provide important value added features that any enterprise must consider implementing as part of a Linux client migration plan.

Not only do systems management tools help in the installation and migration of Linux desktops, but they can also help to maintain these desktops throughout their lifecycles. More about systems management tools can be found in Appendix B, “Using enterprise management tools” on page 255.

6.9 Do not migrate until you are ready

Replacing an operating system is a difficult task. It should not be done lightly, or without sufficient planning. Even then, Linux might not be the perfect solution for some users (such as those with many unmigratable applications, or laptop users whose power management, sound, and wireless drivers are not supported under Linux). It might be necessary to have a mixed network of Linux, Windows with only commercial software, and Windows transitional desktops. While this setup is

more complicated than a Linux-only network, it is often still advantageous to migrate some users.

More migrations fail from under-planning than over-planning. While not every contingency can be planned for, the majority of possible problems can be found before the migration of the first user. Sufficient planning requires thoroughly investigating all distributions and software choices, involving the users with the proposed solution, and performing a pilot migration with the systems management and automated migration tools you will use. After that, any and all problems encountered need to be fixed and then run through the testing process again. Only after a successful pilot migration should the migration of the rest of the organization begin. However, before migrating some of the groups, you might need to plan and retest for any additional needs they have.

6.10 Do not just migrate, upgrade

A migration is not just a task; it is an opportunity to switch to a better environment. For instance, a systems management tool could be introduced to ease the migration, but also to manage machines once they have been migrated. Depending on the role of the user, it could ease future application rollout and technical support to switch to a thin client or rich client environment. Perhaps some of the main IT problems have resulted from users having too much control over their desktops; implementing the KDE Kiosk feature on those desktops could keep the users from performing any potentially destructive tasks by limiting them to usage of only approved interfaces. More information about additional desktop configurations is available in Chapter 7, “Client deployment models” on page 139.



Client deployment models

In this chapter, we concentrate on several client deployment models that can make desktop Linux clients easier to manage.

The sections in this chapter are:

- ▶ 7.1, “Restricting the desktop” on page 140
 - Methods for locking down the KDE desktop using the KDE Kiosk framework, the GNOME desktop with gconftool-2, and other tools are described.
- ▶ 7.2, “Remoting tools” on page 154
 - Several Linux-based solutions, which involve separating application logic from the display of that application, are covered.
- ▶ 7.3, “Rich client” on page 156
 - Rich client solutions are introduced and several implementations are discussed.
- ▶ 7.4, “Stateless client” on page 160
 - The goals behind stateless clients are introduced and some solutions are discussed.
- ▶ 7.5, “Multi-station client architecture” on page 162
 - An in-depth discussion about Linux-based multi-station computing architectures.

7.1 Restricting the desktop

An out-of-the-box Linux installation gives users a lot of power to change their settings according to personal preferences. In an enterprise environment, this is not necessarily preferable. There are several ways to lock down a Linux desktop, depending on the applications that need to be locked down. We describe in 4.3.4, “User lockdown” on page 83 how to keep users from using certain hardware or software. In this section, we describe methods for locking down KDE desktops using the Kiosk framework. We also look at how to set default and mandatory settings for GNOME desktops using gconftool-2 and several graphical tools.

7.1.1 KDE Kiosk framework

In this section, we describe how to lock down your KDE desktop with the Kiosk framework that was introduced in KDE 3. Kiosk allows you to disable certain KDE features to create a more controlled environment. It is built on top of KDE’s configuration framework and adds a simple API that applications can query to get authorization for certain operations. We demonstrate both editing the configuration files directly and using the easy-to-use Kiosk Admin Tool¹ administration GUI tool.

For more information about how KDE stores its settings and how to edit them, see “Desktop personalization: KDE Desktop” on page 315.

Profiles

The KDE Kiosk framework should be used in addition to standard Linux security measures; the files that disable certain features should only be writable by the Kiosk administrator, which can be the root user or somebody else designated specifically for that task. The Kiosk framework uses the standard UNIX user and group semantics and adds a profile layer to it. Profiles are configuration sets that specify what can be done when working with KDE and are associated to single users or groups of users.

You have to tell KDE in the global configuration file /etc/kderc (or sometimes in /usr/share/config/kdeglobals) which profiles are available and where to find the mapping file. In Example 7-1 on page 141, we showcase the configuration with two user profiles, which are internationalized (English and German [de]) and owned by the root user (ProfileInstallUser=root). The other three profiles look identical.

¹ <http://extragear.kde.org/apps/kiosktoll/>

Example 7-1 Profiles and the mapping file are specified in /etc/kderc

```
[Directories]
kioskAdmin=root:
profileDirsPrefix=/etc/kde-profile/
userProfileMapFile=/etc/kde-user-profile

[Directories-default]
ProfileDescription=Default profile
ProfileDescription[de]=Standard Profil
ProfileInstallUser=root
prefixes=/etc/kde-profile/default/

[Directories-ITSO]
ProfileDescription=ITSO Test Profile
ProfileDescription[de]=ITSO Standard Profil
ProfileInstallUser=root
prefixes=/etc/kde-profile/ITSO/
```

For our IBM Technical Support Organization (ITSO) migration, we chose five profiles specified by their unique names default, Redbook, ITSO, Designer, and Administrator, and corresponding configuration directories /etc/kde-profile/[Profile Name]. In Table 7-1, you can see the meaning of the five profiles that we are using in our test scenario.

Table 7-1 Profiles and their meanings

Profile	Role
default	No specific role, default
Redbook	Redbook writers
ITSO	ITSO staff
Designer	ITSO staff with designer tasks
Administrator	Has complete access to the machine

In the next step, we have to assign profiles to our users and groups. You can use the graphical KDE Kiosk Admin Tool for that task (see Figure 7-1 on page 142) or just edit the configuration file manually (see Example 7-2 on page 142).

When a user has more than one profile associated with that user, then the settings of the profiles listed first have higher priority in conflicting situations. The same is true for groups in the Groups section.

Example 7-2 Mapping profiles to users and groups in /etc/kde-user-profile

```
[General]
groups=itso,redbook,users
```

```
[Groups]
itso=ITSO
redbook=Redbook
users=default
```

```
[Users]
anette=Designer
root=Administrator
```

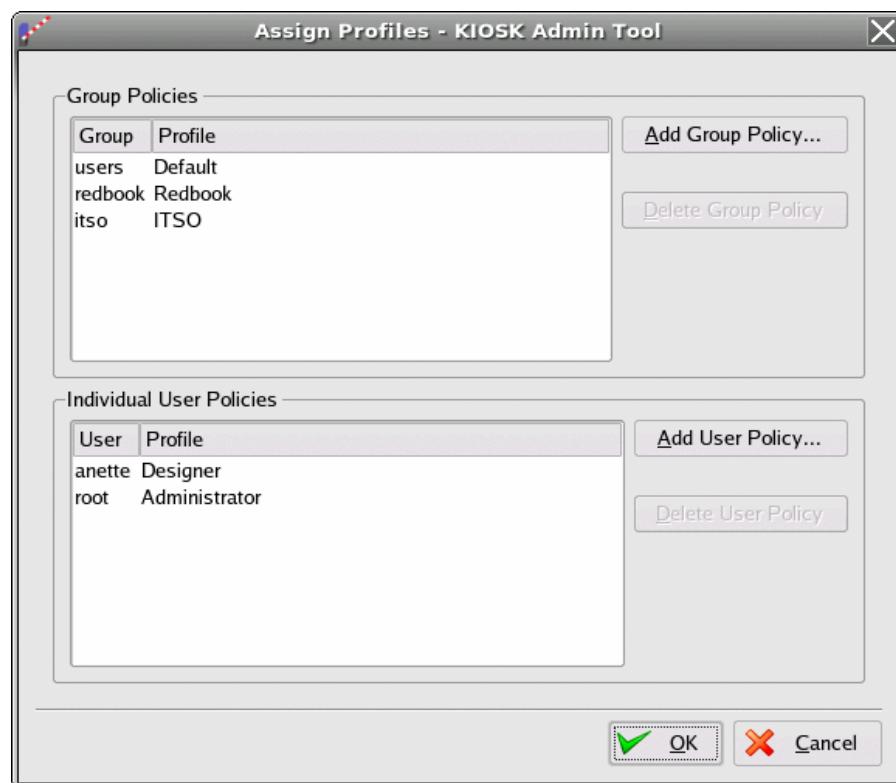


Figure 7-1 Configuring users and groups with the Kiosk Admin Tool

Another interesting case happens when a user not listed in the Users section is part of different UNIX groups that are mapped to different profiles in the Groups section. In that case, the profiles listed earlier will have higher priority. Last but not least, if a user is listed in the Users section, only the profiles in this entry are

taken into account (that is, the UNIX groups the user belongs to and which are mapped to profiles play no role in that case).

Although all these rules are quite straightforward, a simple or even linear profile hierarchy is sufficient in most cases. For example, you can design the default user profile first, add some features for a group needing more privileges, and so on. If there are groups not fitting in that hierarchy (in our case, this could be the designer group), build their profile separately and either merge it with a profile in the first hierarchy or use it on its own.

A nice feature of the Kiosk Admin Tool is its remote configuration ability. In Figure 7-2, you see how to map a local directory to a remote one (you can even change the base URL path). Files are transferred, for example, by the SSH protocol and the usual fish:// semantics or any other protocol that is supported by the KDE networking framework.

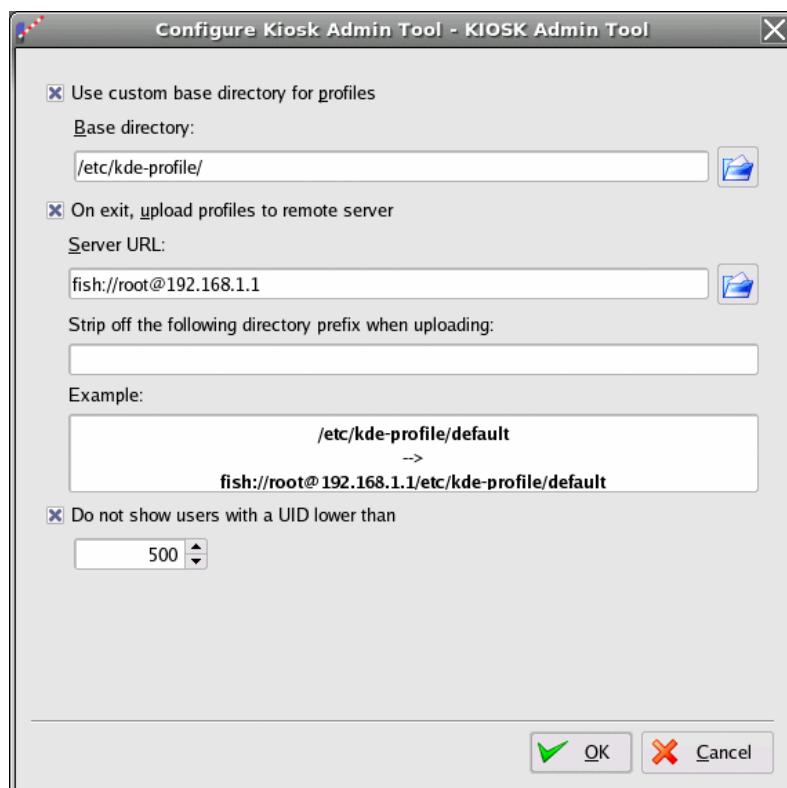


Figure 7-2 Remote configuration with the Kiosk Admin Tool

Kiosk lockdown options

Let us take a look at some of the available general options now. For the general configuration (Figure 7-3):

- ▶ Disable window manager context menu (Alt+F3).
- ▶ Disable bookmarks.
- ▶ Disable all tasks and applications that require root access.
- ▶ Disable access to a command shell.
- ▶ Disable logout option.
- ▶ Disable lock screen option.
- ▶ Disable Run Command option (Alt+F2).
- ▶ Disable toolbar moving.
- ▶ Disable execution of arbitrary .desktop files.
- ▶ Disable starting of a second X session.
- ▶ Disable input line history.
- ▶ Disable “Edit file type” in properties dialog.

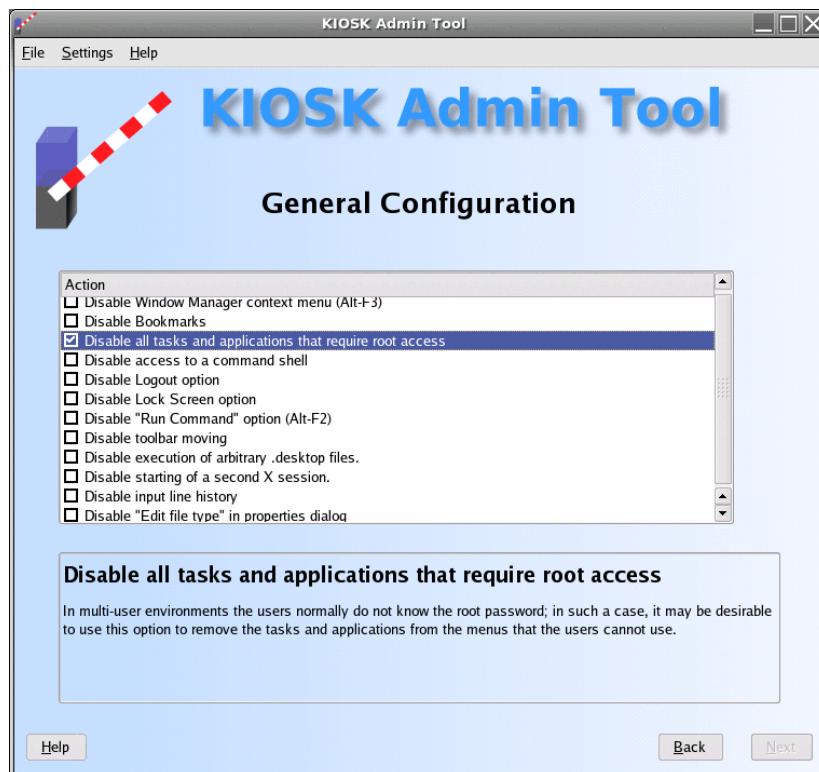


Figure 7-3 General configuration with the Kiosk Admin Tool

The following sections can be configured live, that is, you can add applets to a second Kicker panel, change the background image or color, manipulate menu

entries, and add icons on a second desktop and see the results immediately. An example is in Figure 7-4. Configurable options are:

- ▶ Desktop icon configuration
 - Lock down desktop settings.
 - Disable context menus.
 - Lock down all desktop icons.
 - Lock down system-wide desktop icons.
- ▶ Desktop background configuration
 - Lock down desktop background settings.
- ▶ Screen saver configuration
 - Lock down screen saver settings.
 - Disable OpenGL-based screen savers.
 - Discrete screen savers only.
- ▶ KDE menu configuration
 - Disable all tasks and applications that require root access.
 - Disable menu editing.



Figure 7-4 Theming configuration with the Kiosk Admin Tool

- ▶ Theming configuration
 - Lock down style settings.
 - Lock down color settings.

- Lock down font settings.
- Lock down window decoration settings.
- ▶ Panel configuration
 - Lock down panel.
 - Disable context menus.
 - Disable menu editing.
- ▶ Network proxy configuration
 - Lock down proxy settings.

The disabling of actions in the next two sections is an art itself since there are so many of them. We look at that in more detail in “Action restrictions” on page 148.

- ▶ Konqueror configuration
 - Disable properties in context menu.
 - Disable Open With action.
 - Disable Open in New Tab action.
 - Disable file browsing outside home directory.
- ▶ Menu actions (Figure 7-5 on page 147)
 - Disable **File** → **New**.
 - Disable **File** → **Open**.
 - Disable **File** → **Open Recent**.
 - Disable **Help** → **About <Application>**.
 - Disable **Help** → **About KDE**.



Figure 7-5 Menu actions configuration with the Kiosk Admin Tool

Immutable configuration file entries

As we have seen, it is quite easy to generate lockdown profiles with the graphical Kiosk Admin Tool, but what does this program really do behind the scenes? Starting with KDE 3, configuration entries can be marked immutable, and once such a value has been read, its value cannot be changed again through KConfig or user entries in \$KDEHOME (normally \$HOME/.kde).

Entries can be marked immutable on an entry, group, or file basis by adding [\$.i] at the right places, as shown in Example 7-3 on page 148. If KDE does not have write access to the user's configuration files, they will automatically be considered immutable and the user will be warned about that fact. If you do not like this behavior, add warn_unwritable_config=false to the KDE Action Restrictions section in /etc/kdrc (or kdrglobals on the global, profile, or user level) to disable this warning for all applications. Non-writable user configuration files are not a foolproof lock down mechanism because users can potentially rename these files and add new ones according to their tastes. Consider the file system mechanisms an add-on to the much more sophisticated KDE Kiosk framework.

Example 7-3 Immutable entries are marked with [\$i]

```
[ScreenSaver]
Enabled[$i]=true

[Desktop0][$i]
Wallpaper=/usr/share/backgrounds/images/default.png
WallpaperMode=Scaled

[$i]
[Applet_1]
ConfigFile=kminipagerappletrc
DesktopFile=minipagerapplet.desktop
FreeSpace=0
WidthForHeightHint=92
```

Action restrictions

Using the Kiosk Admin Tool, we have already encountered action restrictions that are configured on a profile level in the `kdeglobals` file in the KDE Action Restrictions section. For our ITSO profile, look at Example 7-4 to see what kind of entries have been generated.

Example 7-4 Action restrictions in /etc/kde-profile/itsoshare/config/kdeglobals

```
[KDE Action Restrictions][$i]
action/kdesktop_rmb=false
action/kicker_rmb=false
action/menedit=false
editable_desktop_icons=false
editable_system_desktop_icons=false
movable_toolbars=false
run_command=false
user/root=false
```

There are many actions available and a lot more will be added in the future by the applications using the KDE framework. You have actions that refer on a global level to menu and toolbar entries (for example, `action/file_new`) and general actions for:

- ▶ Printing (`print/options`)
- ▶ Screensavers (`opengl_screensavers`)
- ▶ Desktop (`logout`, `lock_screen`, `movable_toolbars`, and `start_new_session`)

There are also actions related to standard KDE programs such as:

- ▶ Konqueror (`action/openintab`) and KDesktop
- ▶ KWin (`action/kwin_rmb`)
- ▶ Kicker (`action/kicker_rmb`)

- ▶ Konsole (action/show_menuBar)

There are also a lot of additional application actions that can be explored as usual with the **dcop** command on the command line or with the KDE program kdcop in an easy-to-use graphical way. Enter, for example, one of the following commands after starting kmail to see what actions kmail offers:

```
% dcop kmail qt objects | grep KActionCollection | cut -d '/' -f 3  
% dcop kmail kmail-mainwindow#1 actions
```

There are also actions that refer to applications that need to be run as a different user. They are prefixed with user/ followed by the username (for example, user/root=false to disable all application entries that require root access). If you are interested in even more subtleties, check the Kiosk readme:

<http://webcvs.kde.org/cgi-bin/cvsweb.cgi/kdelibs/kdecore/README.kiosk>

URL restrictions

It is possible to restrict URL-related actions with the Kiosk framework based on the action, the URL in question, or in some cases the referring URL. As you can see in Example 7-5, the general syntax is quite long, so we will explain some simple cases.

Example 7-5 URL restriction: General syntax

```
[KDE URL Restrictions]  
rule_count=<N>  
rule_1=<act>,<ref_proto>,<ref_host>,<ref_path>,<proto>,<host>,<path>,<enabled>  
...  
rule_N=<act>,<ref_proto>,<ref_host>,<ref_path>,<proto>,<host>,<path>,<enabled>
```

In the first part of Example 7-6, you can see how to disable browsing with KDE's file dialogs outside the \$HOME directory by using the list action. The first rule disables browsing any directories on the local file system, while the second enables browsing in the \$HOME directory, which is exactly what we want. You can also see that KDE expands the environment variable \$HOME. Usually, you have to put [\$e] after the entry name, or even [\$ei] to prevent the application replacing the entry with the actual environment value's value after saving, but this is not necessary everywhere. The second part of Example 7-6 shows how to allow the user to open files in the \$HOME and \$TMP directories but nowhere else on the file system (opening files from the Internet is still possible though).

Example 7-6 URL restriction examples

```
[KDE URL Restrictions] [$i]  
rule_count=2  
rule_1=list,,,file,,,false  
rule_2=list,,,file,$HOME,true
```

```
[KDE URL Restrictions][$i]
rule_count=3
rule_1=open,,,file,,,false
rule_2=open,,,file,$HOME,true
rule_3=open,,,file,$TMP,true
```

You can also use shell commands in KDE configuration files, as in Example 7-7, but do not overuse this feature.

Example 7-7 Using shell commands in KDE configuration files

```
Host[$e]=$(hostname)
...
[Icons]
Theme[$e]=$(source /usr/share/config/kdeglobals.defaults && echo $Theme)
```

Resource restrictions

The last lockdown capability that we discuss is the KDE resource restrictions, which make it impossible for users to override file lookup outside \$KDEHOME/share (where most KDE applications put their resources such as HTML documentation or sound files) with resources lying in \$KDEHOME/share. In the first part of Example 7-8, we do this for all resources, while in the second part we use a more granular approach by just restricting sound, localization, and wallpaper files.

Example 7-8 Resource restriction covering all resources

```
[KDE Resource Restrictions][$i]
all=false

[KDE Resource Restrictions][$i]
sound=false
locale=false
wallpaper=false
```

Some of the entries we can use are:

- ▶ data (share/data)
- ▶ html (share/doc/HTML)
- ▶ icon (share/icon)
- ▶ config (share/config)
- ▶ pixmap (share/pixmaps)
- ▶ apps (share/applnk)
- ▶ xdgdata-apps (share/applications)
- ▶ sound (share/sound)

- ▶ locale (share/locale)
- ▶ services (share/services)
- ▶ servicetypes (share/servicetypes)
- ▶ mime (share/mimeLnk)
- ▶ wallpaper (share/wallpaper)
- ▶ templates (share/templates)
- ▶ exe (share/bin)
- ▶ lib (share/lib)
- ▶ all (share/)
- ▶ data_<application name> (share/apps/<application_name>)

This is somewhat confusing since not all directory names are mapped to the same names in the configuration file. Example 7-9 shows how to use the Control Module and Custom Restriction sections in the kdeglobals file.

Example 7-9 More KDE restriction options

```
[KDE Control Module Restrictions]
kde-background.desktop=false
kde-colors.desktop=false
kde-fonts.desktop=false
kde-kwindecoration.desktop=false
kde-proxy.desktop=false
kde-style.desktop=false

[KDE Custom Restrictions]
restrict_file_browsing=true
```

Kiosk limitations

Disabling the run command and access to a command shell might be preferable for a public terminal, but it is quite easy to open a shell with Mozilla, which does not use the Kiosk framework. When you disable the logout option, you also have to disable shutting down the X server by pressing Ctrl+Alt+Backspace, or the whole machine with Ctrl+Alt+Delete (these settings can be configured in /etc/inittab), otherwise, the restrictions could be easily worked around. There are a lot of side effects to consider in order to make the whole setup bullet-proof, because many Linux applications and subsystems do not use the KDE Kiosk framework.

7.1.2 GNOME lockdown options

The GNOME desktop uses a database, which is represented by Extensible Markup Language (XML) files to build and customize all of its settings. This database is built from three sources: the settings, default, and mandatory sources. Changes made by a user typically go into the settings source. The

default source contains default options that are used for each setting that has not been configured by a user. Finally, the mandatory source overrides users' personal preferences and disallows changes to the associated setting. The settings source is usually stored in \$HOME/.gconf. The default source is found in /etc/gconf/gconf.xml.defaults on Red Hat-based systems and /etc/opt/gnome/gconf/gconf.xml.defaults on SUSE-based systems. And, the mandatory source is /etc/gconf/gconf.xml.mandatory on Red Hat-based systems and /etc/opt/gnome/gconf/gconf.xml.mandatory on SUSE-based systems.

For more information about how GNOME stores its settings and how to edit them, see "Desktop personalization: GNOME Desktop" on page 317.

gconftool-2

The GNOME desktop ships with a tool called gconftool-2. You can use it from the command line to edit the GConf database, and you can edit any of the configuration sources. Because the default and mandatory sources apply to all users, they typically can only be edited by the root user. Also, these changes directly edit the GConf database sources, and, thus, they do not affect any GConf session that is currently running.

If you want to remove the keyboard shortcut for bringing up the Run Application dialog (which is normally Alt-F2), you run the following command:

```
gconftool-2 --direct --config-source  
xml:readwrite:/etc/gconf/gconf.xml.mandatory --type bool --set  
/apps/panel/global/run_key false
```

If you want to set the default number of workspaces for new users, but you still allow them to change this setting later, you run the following command:

```
gconftool-2 --direct --config-source  
xml:readwrite:/etc/gconf/gconf.xml.defaults --type int --set  
/apps/metacity/general/num_workspaces 2
```

Graphical tools

GUI-based tools for locking down the GNOME desktop also exist. Similar to the use of gconftool-2, gconf-editor can be used to directly edit the default and mandatory sources. To do so, launch gconf-editor as root, and then choose **File → New Defaults Window** or **File → New Mandatory Window**. Just as with the command-line tools, changes to the default and mandatory sources do not take effect for any GConf sessions that are already running.

Pessulus

Another available tool is Pessulus. Pessulus is a graphical tool, which configures the lockdown settings built-in to GNOME, which otherwise can only be configured via GConf. Pessulus is included in the GNOME admin suite starting

with version 2.14 of GNOME. These settings include limiting which panel applets can be added, and whether desktop icons can be edited. More information about Pessulus is available at:

<http://www.gnome.org/~vuntz/pessulus/>
<http://live.gnome.org/Pessulus>

Sabayon

Sabayon is a graphical systems administration tool that can be used to manage GNOME desktop configuration settings. Sabayon brings a new approach to policy editing. Instead of providing a pre-built GUI to edit each of the settings, Sabayon launches a nested X session that runs an entire GNOME desktop environment (more information about X sessions can be found in 5.2.4, “Graphical and text-based environments” on page 118). Any settings changed in the nested environment are recorded, and at the end of the session, those settings can be applied to either the default or mandatory GConf stores. Sabayon also detects which local files have been edited, which is necessary for applications that do not use GConf, such as Mozilla Firefox or OpenOffice.org. Sabayon shows you a list of settings that have been changed, which you can then save as default or mandatory settings.

More information about Sabayon can be found at:

<http://www.gnome.org/projects/sabayon>

As an example, graphic designers need a profile which contains a shortcut to The GIMP on the panel and has your company’s Web site as the home page. To create this type of profile through Sabayon, simply add a new profile, then click **Edit**. In the nested session that appears, add a shortcut to The GIMP on the panel, then launch Firefox and set the home page. As seen in Figure 7-6 on page 154, Sabayon then shows the two changes that have occurred, which you can save as default or mandatory settings.

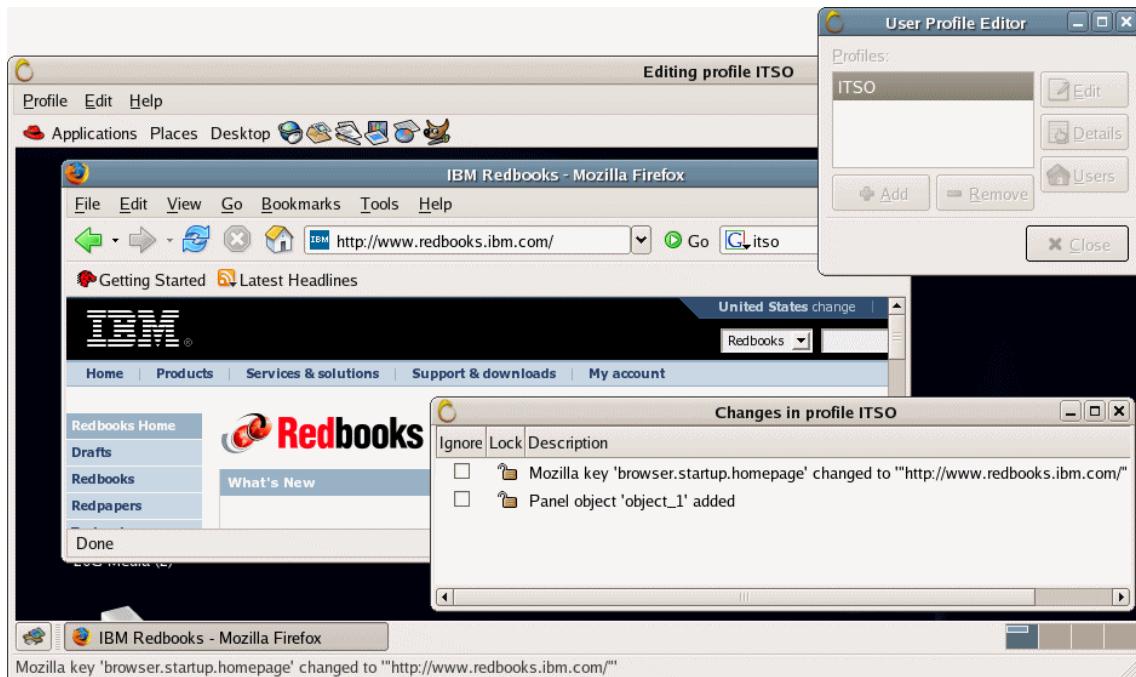


Figure 7-6 Sabayon editing a profile

7.2 Remoting tools

Due to the modular nature of Linux, there are numerous options when implementing a thin or slim client desktop solution. The options include basic remote access, where one workstation remotely displays the desktop of another workstation, network booting diskless hardware, remote execution of a single application with a local display, or multiple users connecting to a single workstation.

7.2.1 Remote access

Basic remote access is where the entire application logic resides on a server and the client only runs presentation logic. Implementations include Virtual Network Computing (VNC), NoMachine NX, or even rdesktop, a Linux client for Microsoft Remote Display Protocol (RDP) servers. However, because this kind of remoting usually requires a significant amount of network resources, remoting of the entire operating system is usually not used as a user's standard environment. Instead, they are used for situations such as remote diagnosis and help desk scenarios, or accessing a desktop workstation while out of the office.

7.2.2 Thin client

Thin client solutions involve many clients (either standard workstations or diskless terminals) booting over the network off of a central server. Because applications are all run on the server, upgrades and patches only need to be applied to one location. Also, most user data is stored on the server, so users can use any thin client connected to the server and receive the same experience. Depending on the solution in use, thin clients can require a substantial amount of network bandwidth between the clients and the server. Modern thin client solutions include the Linux Terminal Server Project at:

<http://www.ltsp.org>

And various thin client solutions from Neoware:

<http://www.neoware.com>

7.2.3 Application forwarding

The X client/server environment (as described in 5.2.4, “Graphical and text-based environments” on page 118) allows a more granular approach to remoting. Because X clients already send their displays through networking sockets to connect to a local X server, forwarding the display to a remote X server is trivial and requires no changes to the client. There are numerous scenarios where application forwarding could reduce technical support issues, such as configuring a desktop to run e-mail and Web browsing applications locally, but to connect to a network server to run an in-house application which receives frequent updates. X forwarding can be tunneled over a Secure Shell (SSH) connection to encrypt the session, allowing for secure application forwarding over the Internet or an unsecure portion on your intranet. It is also possible to use NoMachine NX or FreeNX, which compresses the session significantly, thus making applications far more responsive over slower connections, and usable even over dial-up connections.

Similar application remoting can be done with Windows applications running through products such as Sun Secure Global Desktop Software (formerly from Tarantella) or Microsoft Terminal Server. These services can be useful for unmigratable applications; the unmigratable application can run on the server, while all other applications run locally.

7.2.4 Multi-station computing

Thin client solutions are often limited in performance and their deployment requires extensive infrastructure. Performance and response time are highly sensitive to network traffic between the client and the server, potentially frustrating some users. Multi-station computing based on standard desktop PC

hardware overcomes many of these challenges. Due to widespread availability of multi-headed video cards and USB input devices, many personal computers can be configured for two local users simply with the addition of a new set of user input and output devices such as keyboards, mice, and monitors. With the addition of one or more video cards, several users can be simultaneously supported without taxing a desktop computer's multi-gigahertz CPU or gigabytes of memory and hard disk storage.

Deploying multi-station systems instead of an individual computer for each user can reduce infrastructure build-out costs of both initial hardware and software purchases, as well as through future savings in electricity and data network infrastructure. In addition, the total cost of ownership of each workstation is dramatically reduced during the life of the system due to reduced hardware and software support and maintenance. With multi-station systems, IT departments can refresh software images on fewer computers while also reducing software subscription and support costs. And at the end of the computer life cycle, disposal costs are reduced.

Consider deploying multi-station client architectures whenever user workstations require clustering in close proximity, and their primary application usage patterns do not involve CPU or memory intensive requirements. We discuss multi-station computing strategies in more detail later in this chapter (7.5, “Multi-station client architecture” on page 162).

7.3 Rich client

Rich client technology is based on a client/server relationship that attempts to bring all the advantages of a browser-based thin client solution while inheriting none of the disadvantages compared to a local application-based thick client solution. In a rich client environment, applications are hosted on a central server, which clients download on demand and run completely on the local machine. As such, application upgrades or patches only need to be applied to the central server, and then are propagated to the client on the next run, greatly reducing the costs of rolling out an upgrade or application patch. And, because the application logic runs on the client, rich client solutions take full advantage of the power available on a standard desktop. By running the application locally, a full-featured application environment can be used, unlike browser-based solutions which severely limit the environment. Finally, rich clients can cache the latest version of an application, resulting in minimal network traffic during online use, and allowing for offline use that a thin client or browser-based solution cannot provide. The sweet spot for rich-clients is illustrated in the upper right quadrant of Figure 7-7 on page 157.

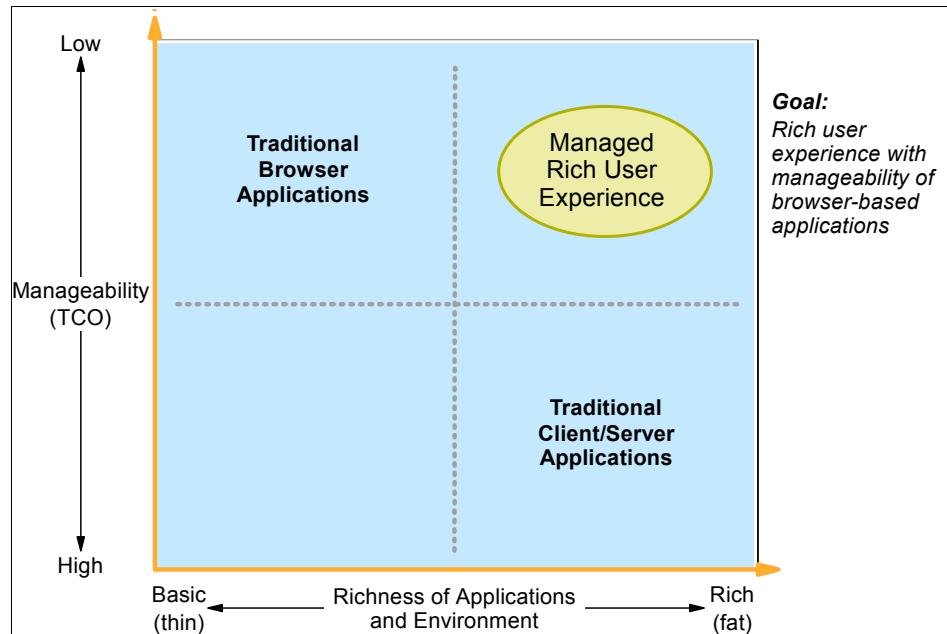


Figure 7-7 Manageability of rich clients: The magic quadrant

7.3.1 Eclipse and the Eclipse Rich Client Platform

Eclipse is an open source IDE and integration platform. To develop an understanding why the Eclipse platform was created, you should consider what is currently happening at the intersection of two different IT industry trends:

- ▶ The explosion in Internet use. This led to a need for Web application services, specific file formats, enabling protocols, markup languages, media types, and the many different types of client applications and developer tools needed to tie it all together in a seamless experience.
- ▶ The open source movement. The exploding popularity and increasing credibility of open source software development methods naturally encourage development of a supporting set of open standards.

What follows then is the need to create a free, open, standards-based, and extensible application development framework. Meeting the need for a standards-based open and extensible application development platform has from the beginning been one of the primary goals of the Eclipse project. And with the release of Eclipse 3.0, the Eclipse platform became more than just a Java-based IDE and tool integration platform. It also became a very capable host for client applications. Thus, the Eclipse Rich Client Platform was born.

*“While the Eclipse platform is designed to serve as an open tools platform, it is architected so that its components could be used to build just about any client application. The minimal set of plug-ins needed to build a rich client application is collectively known as the **Rich Client Platform**. ”²*

7.3.2 IBM Workplace Client Technology

The new IBM Workplace Client Technology™ provides a flexible and extensible framework for applications. It is built on a standards-based platform and utilizes the Eclipse technology as a base component. Among its approaches to help enable low total cost of ownership (TCO), a key element of IBM Workplace Client Technology is its ability to provide a no-touch deployment model coupled with policy-based management. Additionally, as an enterprise solution, IBM Workplace Client Technology provides built-in security features and a managed, synchronized relational data store that supports both connected and disconnected operations. As one example, this secure data store can provide document management capabilities, and when combined with plug-in document editor support (another capability of this technology), it gives enterprises a new level of document management combined with low TCO and ubiquity.

IBM Workplace Client Technology moves beyond the browser, enabling not only browser capabilities, but also the ability to securely and incrementally download, update, and intelligently cache the next generation of “rich” and hybrid client applications. These applications run locally on the user’s machine using an encrypted, synchronized content store with security features and offline capabilities. Such applications harness the full power of the user’s machine to deliver the state of the art in capability and performance while continuing to be centrally deployed, upgraded, and administered at the server, side by side with the browser-based applications.

IBM Workplace Client Technology delivers this next generation of rich applications with the low total cost of ownership comparable to the TCO of traditional browser-based applications.

For more information about IBM Workplace Client Technology, see the IBM Redpaper *IBM Workplace Client Technology (Rich Client Edition) Technical Overview*:

<http://www.redbooks.ibm.com/abstracts/redp3884.html?Open>

7.3.3 IBM Workplace Managed Client

IBM Workplace Managed Client (WMC) is a specific implementation of IBM Workplace Client Technology. It is a server-managed application that hosts key

² From the [eclipse.org Rich Client Platform home page](http://eclipse.org/RichClientPlatform), found at: <http://eclipse.org/rcp/>

collaboration applications and productivity tools and can easily be extended using the Eclipse-based application development framework. It offers offline support for e-mail, calendaring, scheduling, and document management.

The main characteristics of the IBM Workplace Managed Client compared to other client technologies can be summarized as:

- ▶ A centrally managed, policy-based client provisioning system that ensures that initial installations as well as maintenance updates can be applied on the server side, and that users can experience those new updates client-side and fully dynamically
- ▶ A synchronizing secure data store that is locally offline accessible and centrally manageable (for backups and so forth)
- ▶ A componentized architecture that allows for gradual extension and that can be distributed on demand
- ▶ An application development platform that users and third-party vendors can use to extend the framework to their needs

Applications that come with WMC include:

- ▶ A messaging framework for sending and receiving e-mails and an instant messaging product for real-time chat sessions
- ▶ An embedded Web browser to access additional portlets on the server side and to directly browse the Web
- ▶ A document library that is synchronized to the server and can be shared with other team members

IBM Workplace Managed Client includes an office productivity editor for creating, editing, and sharing a variety of document types including: word processing, spreadsheet, presentations, and project management. The IBM productivity tools are compatible with Open Office applications and support the open standard OASIS open document format (ODF). The IBM Workplace Managed Client productivity tools can also be used to create, edit, and save documents in a variety of other formats - including Microsoft® Office application formats. By supporting open standards document formats, IBM Workplace Managed Client ensures cross-platform and cross-application document exchange without being locked into the proprietary vendor formats.

OpenDocument: The OpenDocument format (ODF), short for the OASIS Open Document Format for Office Applications, is an open document file format for saving and exchanging editable office documents such as text documents (including memos, reports, and books), spreadsheets, charts, and presentations. This standard was developed by the OASIS industry consortium, based upon the XML-based file format originally created by OpenOffice.org.

OpenDocument is the only standard for editable office documents that has been vetted by an independently recognized standards body, has been implemented by multiple vendors, and can be implemented by any supplier (including closed source software vendors as well as developers using an OpenSource license).

IBM Workplace Managed Client also offers a new collaboration tool called *Activity Explorer*. Activity Explorer lets teams of users manage projects through an ad hoc workflow which groups together information objects that are related to an ongoing project and are shared among team members. The information objects can be documents, files, and notes. The ad hoc workflow is presented in graphical, hierachal format and makes it easier to refer to past work, visualize project status, and discover opportunities for further progress.

For Lotus Notes® users, there is a Notes plug-in available (the Linux version is currently in a BETA stage) that provides access to all Lotus Notes databases inside the WMC framework.

For more information about the IBM Workplace Managed Client, refer to the Redbook *IBM Workplace Managed Client 2.6 on Linux*, SG24-7208-00, at:

<http://www.redbooks.ibm.com/abstracts/sg247208.html>

7.4 Stateless client

A *stateless client* is a desktop installation that stores no important state information on the client. This means that not only is every user's personalization data stored on a central server, but operating system and application installation and configuration are performed through a central server. Stateless clients can either boot the entire operating system from the network or can cache an installation locally and synchronize updates at a regular interval. A stateless installation includes benefits of a thin-client solution in that user data is backed up centrally and available to any client automatically, and that application patches and upgrades only need to be performed on one central server. However, unlike thin-client solutions, stateless clients run all applications locally,

thus, utilizing the full power behind most desktop workstations, and require no specialized network infrastructure. Some stateless Linux installations do not appear to be any different than a standard Linux installation. However, if a client machine were to fail, it could be completely rebuilt nearly instantaneously, with no data lost. The benefits of stateless clients are clear, however as of the time of writing this IBM Redbook, stateless Linux required running in-development software or a custom implementation.

“Stateless Linux ... where the root partition is mounted read-only and synched transparently with a central source, giving the central-state advantages of thin client with the low hardware and network infrastructure costs of a cheap-Intel-box thick client.”

— Seth Nickell, from:

<http://www.gnome.org/~seth/blog/sabayon>

7.4.1 Red Hat’s Stateless Linux project

Currently in development, Red Hat’s Stateless Linux project aims to include stateless Linux with Red Hat Enterprise Linux 5. To deploy Stateless Linux, first a prototype server needs to be created. This server stores the snapshots, or client installations, that are to be distributed to each stateless client. A snapshot is a complete installation of an operating system in a subdirectory on the server. This environment is entered via the **chroot** tool (For more about **chroot**, see the Linux man page), and then managed via standard management tools, such as **yum**.

Stateless Linux supports booting operating system snapshots over the network, from a read-only partition on the local hard drive, or off a CD or DVD-ROM. Updates to a snapshot take effect on the next boot for network boot machines, while local hard drive-based snapshots periodically poll the server for updates. Naturally, CD and DVD-based snapshots have to be burned again for each snapshot release.

A description of the goals and theories behind Stateless Linux is available at:

<http://people.redhat.com/~hp/stateless/StatelessLinux.pdf>

More information about Stateless Linux can be found at:

Stateless Linux tutorial:

<http://fedora.redhat.com/docs/stateless>

Stateless Linux project:

<http://fedora.redhat.com/projects/stateless/>

7.4.2 Custom implementation of stateless client

A customized stateless Linux installation method has been implemented at a large insurance company in Germany, using a highly specialized initrd adaptation. They use about 4,500 laptop computers and another 2,000 stationary clients. Those clients are booted over the network (with PXELinux³) and configured while booting according to:

- ▶ What hardware is detected (using Kudzu⁴ for auto detection)
- ▶ What applications need to be run (for example, some applications require a screen resolution of 1024x768, even if the hardware could do better)
- ▶ How they are connected to the network (Olympic Token ring, ISDN, GPRS Option Card, or Bluetooth Mobile phone)
- ▶ System default login rights: What user and group is logging on (which results in access to different applications)

All configuration data (such as /etc or configuration files in the users' home directories) is assembled during the boot process and written to memory, such that even with access to a hard drive from one such system, it is impossible to tell which network this machine was connected to or who has been working with it. Each user needs to have an up-to-date system at any time. As such, stationary users have access to all applications via NFS. Mobile users (who have a pre-installed Linux system on their Thinkpads) get their applications rsynced during the boot process, though certain parts of the system are rsynced only when the network connectivity is fast enough (for example, a X11 server update does not run over a dial-up connection). After booting, users have thirty seconds to log into the system with a Omnikey USB smart card reader and password; otherwise, the system is rebooted (and in certain cases, even repartitioned and destroyed). This configuration allows for secure data backup, easy installation and upgrade rollout, and a fast, reliable desktop environment for users.

7.5 Multi-station client architecture

In this section, we present an in-depth discussion about how Linux-based multi-station computing architectures that are being pioneered in the marketplace by Userful⁵ Corporation provide an innovative and potentially extremely cost-effective strategy for deployment and management of client workstations.

³ <http://syslinux.zytor.com/pxe.php>

⁴ <http://rhlinux.redhat.com/kudzu>

⁵ <http://www.userful.com>; <http://userful.eu>

The ability to support multi-client architectures aptly demonstrates the flexibility of the Linux client software stack, and specifically, how the flexibility and modularity of the X windows system allows for design of innovative many-to-one client computing architectures (refer to 5.2.4, “Graphical and text-based environments” on page 118).

This section introduces the multi-station computing strategy and discusses deployment considerations in detail. See Appendix D, “Multi-station computing deep dive using Useful Desktop Multiplier” on page 289 for more technical background, case studies, and implementation details.

7.5.1 Multi-station computing and Useful Desktop Multiplier

Multi-station computing software enables a single computer system to be shared by two or more connected users, each user having an independent workstation consisting of a keyboard, monitor, and mouse. For instance, multi-station Linux “desktop virtualization” software such as Useful’s Desktop Multiplier can enable a single IBM IntelliStation® system to provide many users with fully independent desktop workstation services, simply by plugging in extra video cards, monitors, USB keyboards, mice, and other peripheral devices directly to the single shared system.

Providing multiple workstations with a single high-end desktop system (rather than using multiple individual PC systems) can yield significant savings in hardware and software acquisition costs, and it also multiplies those savings for ongoing energy and system administration overhead. Powerful computer systems such as high-end IBM IntelliStation workstations can provide fully independent desktop workstations to many local users, all directly connected to the shared system using regular VGA and USB cabling. Even more users can be connected to the single shared system by using PCI bus expansion units and distance extension technologies as needed.

Multi-station desktop computer systems are ideal for environments where groups of workstations are clustered together in close proximity. Example scenarios include classrooms, point of service systems, training labs, kiosks, call centers, Internet cafes, voting and registration systems, and back-office environments where PCs are used as low-cost console terminals for servers and networking equipment.

Using a powerful and reliable IBM IntelliStation system to serve independent desktops to many directly connected users offers an efficient alternative to deploying multiple desktop computers and can deliver significant advantages over traditional methods of providing desktop systems. Figure 7-8 on page 164

illustrates this concept by showing the combination of one IBM IntelliStation A Pro workstation and two dual-head graphics cards providing a platform that supports four separate client workstations.



Figure 7-8 *IntelliStation A Pro workstation supporting four workstations using two dual-head video cards*

7.5.2 What is multi-station computing

The concept of a single computer providing independent desktop environments to multiple users is not new. Many solutions enable desktops to be remotely served to one or more client computers over a network using a single powerful computer such as a mainframe or terminal server.

Traditional terminal server-based solutions require a PC or proprietary thin-client hardware station to be deployed at each user's physical desktop. However, traditional networked thin client solutions are often limited in display and computing performance, and their deployment typically requires additional network bandwidth capability along with a powerful server system and often uses as much power infrastructure as regular desktop PCs.

Multi-station computing, in contrast, is based on using standard desktop PC hardware to serve desktop UIs directly to each user's monitor, and is designed to overcome many of the challenges and limitations of networked-based thin client solutions. For new users, there are potential cost savings and management

advantages in deploying additional screens and keyboards into existing multi-user platforms, instead of complete stand-alone computer systems.

A basic multi-station configuration

In a multi-station configuration, multiple client workstations simultaneously share the resources of a single computer. Each user's desktop environment and application processes are supported independently. See Figure 7-9.

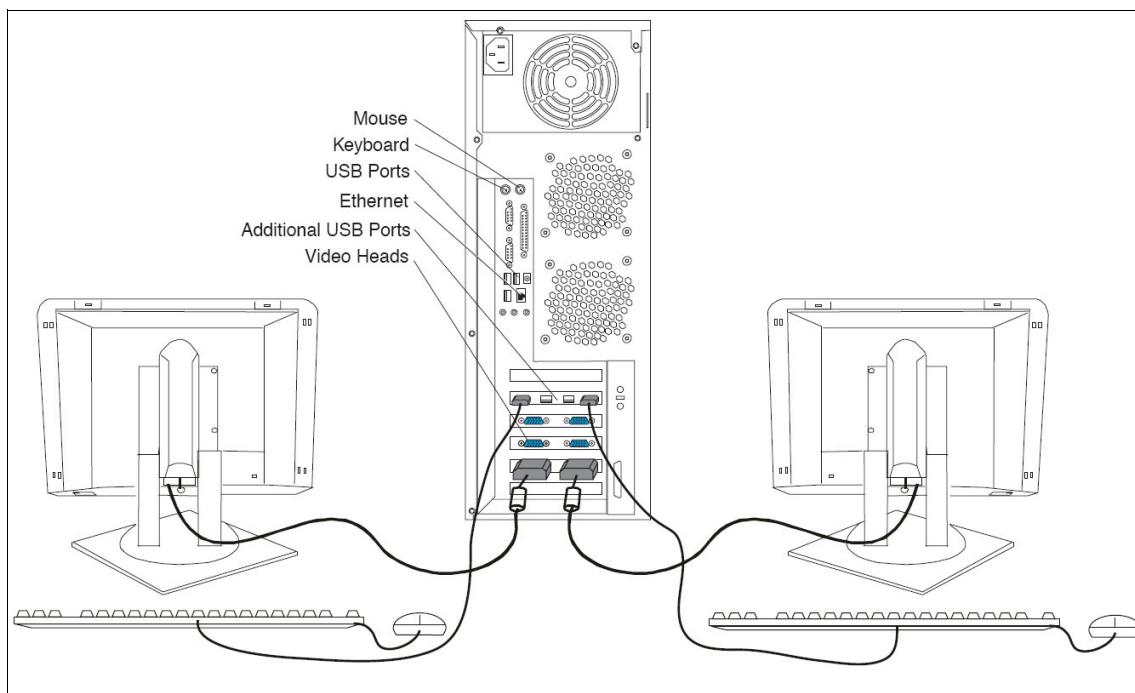


Figure 7-9 IntelliStation A Pro showing VGA and USB connections supporting two client workstations

Each station has an independent set of input and output devices, consisting of a monitor, mouse, and keyboard, which are configured to work together as a set. Figure 7-9 illustrates connections to support two client workstations. Each user is provided with an individual own private desktop environment from which the user can access fixed and removable input, output, and storage devices. These devices are directly connected to the workstation through the use of extension cables, USB hubs, and other devices as necessary to provide stations at almost any distance from the shared workstation. You can configure each user's desktop to consist of a single desktop on a single display, or more advanced setups such as multiple virtualized desktops on a single display, or even multiple desktops on multiple displays.

7.5.3 Approaches to desktop consolidation and deployment

Desktop virtualization is one of several approaches to desktop consolidation and deployment. Multi-station platforms use virtualization software to manage each user's desktop similar to the way a terminal server or virtual machine operates, however multi-station systems also provide each user with a direct connection to their virtual desktop, eliminating the client computer altogether and consolidating many computer systems into a single shared workstation. In this section, we compare and contrast several popular desktop deployment scenarios with a multi-station computing approach.

Comparison with thin client systems

Thin client technology relies on small footprint client workstations that facilitate user interaction between a server and a remote user via a standard data network. PCs contain both the hardware and software to function independently of any other device, while thin clients require connections to servers that provide a set of remotely hosted terminal services: login, session management, hosted applications, and so forth.

Multi-station computing solutions share the advantages of thin client systems such as centralized management and minimal hardware at the physical desktop. However, multi-station client architectures differ with thin client architectures in some key aspects: they can be much less dependent on network connection availability, or network bandwidth limitations, and do not require one-to-one deployment of complete client systems for each user.

Tip: As soon as you move above hardware level considerations, multi-station and thin client architectures can become complimentary instead of competing design strategies. This is illustrated in the following section.

Comparison with remote desktop systems (RDP, VNC, and XDMCP)

The Remote Desktop Protocol (RDP), X Display Manager Control Protocol (XDMCP), and Virtual Network Computing (VNC) protocol are industry-standard methods of connecting to desktops on remote computer systems through a local area network or the Internet.

A user of a PC or thin client system runs an RDP, VNC, or XDMCP client application locally, which captures user input from the local keyboard and mouse, sending it to the remote server through a network connection. All of the processing for applications displayed on the remote desktop is performed on the server, which means that the server hardware and configuration sets the hard limit on the number of simultaneous users who can connect to the host server. In

addition to this physical limit, some remote desktop-capable operating systems (including Windows) might require a license for each connecting client, which imposes a software limit to the number of concurrent users on that server. You should consult with your terminal services vendor as to whether client access licensing savings can be realized through a multi-station approach. See Figure 7-10.



Figure 7-10 On a multi-station system, each client can run a different remote desktop, terminal, or virtual machine

In a multi-station configuration, a single computer serves multiple users, each of whom could operate one or more remote desktops through local RDP, VNC, or XDMCP client applications. This is illustrated in Figure 7-10 on page 167. Using remote desktop software, each user on a multi-station system can connect to almost any remote Windows, *NIX, or Apple desktop hosted on the remote

system. All of the processing for applications displayed on the remote desktop is done on the host system, with minimal load placed on the multi-station client system. Multi-station systems can support multiple thin client and terminal emulation sessions running concurrently with graphical desktop environments such as Windows Terminal Services using Remote Desktop Protocol (RDP), Virtual Network Computing (VNC), XDMCP, or Citrix ICA. Multi-station configurations also support a variety of ANSI text environments, such as IBM 3270, 5250; VT52, VT100, and many other terminal applications, running in parallel with each other, and with local applications.

Remote desktop clients and servers are available for most popular operating systems, including Windows, Linux, and Mac OS X.

7.5.4 Where and when to deploy multi-station systems

There are many factors that you should consider when planning a multi-station deployment. This section discusses deployment considerations pertaining to multi-station Linux desktops and reviews several options for minimizing disruptions and maximizing the value derived from leveraging a shared computing platform. See also Chapter 6, “Migration best practices” on page 131.

New deployments

Always consider deploying multi-station systems whenever two or more users are in close proximity and their primary computer usage patterns do not involve frequent CPU or memory intensive application demands. New multi-station deployments make the most sense as a simple management and cost reduction enhancement when a desktop Linux migration or deployment is already being considered. Multi-station desktops based on Linux pose few, if any, additional challenges over those of single-user desktop Linux deployments. As with single-station Linux workstation deployments, a multi-station Linux deployment benefits from an existing technical infrastructure and expertise capable of supporting Linux workstations and users.

A few key factors should be noted when considering multi-station deployments:

- ▶ IT planners considering multi-station deployments into open plan offices should ensure that walls and furnishings permit running USB and VGA extension cables from monitors and peripherals in user work areas to the multi-station computers. In the typical deployment scenario, a cluster of four to six users share one multi-station computer located at a centrally-accessible cubicle. This configuration minimizes the use of extension cables, and all cabling can be run along or within partitions without crossing floor traffic.
- ▶ Although more costly than extension cables, add-on solutions that carry keyboard, video, and mouse (KVM) over standard category 5 network cable

can be used to deploy client workstations hundreds of feet away from the shared multi-station computer. KVM over Cat5 and other distance solutions such as fiber-optic cabling make it easy to deploy workstations in environments where tight user clusters might not be appropriate or even possible. As with USB and VGA extension cables, ensure that Cat5 and fiber-optic cables used in this manner are well protected from the environment.

- ▶ With the rapid and continuous decline in computer memory prices and the common availability of 1 GB memory modules, planners should consider deploying workstations with as much memory as possible. Typical PC motherboards support up to 4 GB of memory per processor, offering enough capacity for each user to have a 512 MB share. Providing additional memory during the initial deployment ensures that multi-station systems have adequate memory to support more users, and more intensive usage patterns.

Some of these considerations are illustrated in the case studies included in Appendix D, “Multi-station computing deep dive using Useful Desktop Multiplier” on page 289.

Multi-station desktops as a replacement for existing PCs

Deploying multi-station Linux desktops as replacements for existing workstations could be a cost-effective way to maximize the use of current infrastructure. Replacing several older computers with a more powerful multi-station system reduces network complexity and power demands. This makes it possible to add more workstations without substantial additions to the existing technology infrastructure. And as with conventional desktop replacement programs, in almost all cases, existing workstation monitors and input devices can be reused on multiple generations of new multi-station systems.

Provisioned multi-station desktop implementations

In addition to general purpose provisioned office desktops (as described above), multi-station Linux has many applications in fixed function, transactional, provisioned desktop implementations.

The majority of enterprise desktop computers are deployed and managed by IT departments to offer specific business functionality to specific groups of users. The tasks to be performed at these provisioned desktops are known prior to deployment, and the computers themselves are typically imaged and set up with precisely the applications needed for the work that the users need to perform. Individual users generally cannot change settings or install software on these provisioned computers. Linux and open source platforms are ideal for delivering stable and reliable provisioned computer systems that leverage the advantage of a multi-station deployment.

Point of service and call center applications

Multi-station platforms can provide cost-effective access for point of service applications such as service desks, ticket vending, and registration systems. Computer terminals at point of service and point of sale locations are often located in close proximity such that multi-station systems can provide cost-effective access to back-end systems. Since applications running on these systems are typically based on terminal emulation, distributed Java applications, or Web-based technologies, multi-station solutions can easily provide an effective alternative to deploying dedicated PCs or thin client terminals. A multi-station approach provides significant cost savings where the cost of on-site service is high by reducing the number and complexity of computer workstations in the field.

Back office and data centers (terminals)

Multi-station systems combined with terminal emulation software could enable IT departments to more efficiently deploy green-screen terminal replacements that connect securely with terminal-based applications such as online transaction processing (OLTP) systems and console monitors. Each multi-station system could provide up to ten independent user stations, with each running one or more terminal services applications connected to the back-end system using common protocols such as TN3270 and TN5250. Using a multi-station Linux system to provide console terminals eliminates the need to acquire and maintain proprietary terminals and potentially uses less rack space than terminal emulation solutions based on single-station PCs.

Public computer access and education environments

Applications running in public access or educational environments are typically not computationally intensive, users are often clustered in close proximity, and their needs are typically well met with desktop Linux. IT expertise and the resources needed to maintain computer hardware is often scarce in these environments, hence the reduced maintenance of the multi-station approach combined with Linux's enhanced security and client lock down capabilities could provide many significant advantages.

7.5.5 Advantages of deploying multi-station Linux systems

Multi-station systems can potentially provide all the advantages of both thin client and rich client environments, effectively integrating the performance benefits of rich client computing with the cost and management benefits of centralized server-based computing environments.

Performance

Multi-station Linux multiplies the overall return on IT investment, especially when deploying high performance and high quality IBM IntelliStation hardware, because all of the users on the shared system benefit from the increased performance and reliability gains.

Compared to multi-station systems, thin client devices could be more susceptible to negative performance impacts because they depend almost entirely on the server system and underlying network infrastructure to provide applications, storage, and CPU processing power.

In contrast, each station of a multi-station deployment could be directly connected to a single system, thus sharing in the full capabilities of a powerful workstation-class system that might not be affected by variability in network and remote server performance. In a properly designed environment, multi-station systems could provide substantially higher performance for users compared to using dedicated thin-client or low-performance PC systems, while reducing the overall network traffic loading as compared to client/server systems.

Infrastructure and system management

Deploying large quantities of computers anywhere is typically an extremely complex, time-consuming, and expensive process. The total maintenance and support costs for the average corporate PC system can be significant.

Multi-station Linux technology provides an alternative strategy that could reduce redundant hardware, software, energy and system management costs.

Userful Desktop Multiplier, like other approaches to desktop virtualization and desktop consolidation such as remote desktops, can substantially reduce the number of software and support subscriptions required. In addition, because the number of deployed systems is reduced versus using single-station systems, remote management, software updates, and support costs are also reduced.

Security

Since each user on a multi-station client is connected directly to a common computer system, there is no need for client hardware or software at the user's physical desktop other than a regular keyboard, mouse, and monitor. This added security feature can help to facilitate compliance with increasingly stringent internal standards or regulatory access control requirements.

Reducing the number of deployed system disks enables IT staff to more rapidly respond to emerging software security threats that require software updates. Using centralized multi-station deployments reduces the number of systems that need to be both physically and logically secured, and allows the consolidated server systems to exist in centralized and distributed locations as needed.

Cost: Economy of scale

For most organizations, the up-front costs of deploying desktop computer systems typically represents a small fraction of the total lifecycle cost of maintaining those systems. The remaining costs consist mostly of IT staffing costs, hardware and software maintenance, and lost productivity due to interruptions for software updates, troubleshooting, viruses, and various technical support issues.

The multi-station client computing strategy introduces the potential for significant cost savings in total PC lifecycle management. This is due to all of the inherent economy of scale benefits that have been discussed so far.



Client migration scenario

In this chapter, we demonstrate an example client migration. We describe the client as it is used before the migration. The important applications on the client will be identified. We describe a migration plan for this client based on the information in Chapter 3, “Organizational and human factors planning” on page 49, and Chapter 4, “Technical planning” on page 61. The result of the actual migration is shown in the last part of this chapter.

The sections in this chapter are:

- ▶ 8.1, “Example client migration” on page 174
Details on the pre-migration client environment are discussed.
- ▶ 8.2, “Migration plan details” on page 175
Migration plan details and strategy are presented as a result of assessing the existing client environment.
- ▶ 8.3, “Performing the migration” on page 178
The actual migration sequence of steps, with some key configuration details included.

8.1 Example client migration

We perform a migration of a single type of client to show how the methods described in the planning part of this book can be put into practice. The example client migration is a simple one. However, the infrastructure integration is almost completely as described in 4.2, “Integrating with existing network services” on page 69.

8.1.1 Assess the client usage pattern

The client we use for the example migration is an ITSO desktop used to write this book. The migration client has the following properties:

- ▶ It is a registered workstation in an NT4 domain.
- ▶ Microsoft Internet Explorer is used to access intranet and Internet sites.
- ▶ Adobe FrameMaker is the primary content authoring application.
- ▶ Jasc Paint Shop Pro is used to create and work on screen captures.
- ▶ Network printers are used for all printing jobs.
- ▶ Microsoft Outlook connects to Microsoft Exchange for e-mail.

The client is running Windows 2000 with Service Pack 4 installed. We plan to migrate to Linux using the Red Hat Desktop distribution.

8.1.2 Identify the most important applications and infrastructure integration points

Since the user role for this workstation is primarily for writing books, Adobe FrameMaker is the most important application, and the most important infrastructure components are printing and access to network file shares.

The first thing to note about the Adobe FrameMaker application is that there is no appropriate Linux alternative. There is no Linux native version, and moving to another application on Linux is not acceptable from a business point of view. In other words, FrameMaker is a unmigratable application that has to be handled in the manner described in 4.7.2, “Terminal Server, Citrix Metaframe, or NoMachine NX solutions” on page 100.

Printing using the network printers and accessing shares in the NT4 domain means that the methods from 4.2, “Integrating with existing network services” on page 69, have to be followed.

8.2 Migration plan details

In this section, we want to discuss the migration steps, which possible problems we could come across, and how we plan to establish the functional continuity.

After assessing a current ITSO desktop and its environment, as we have done in the section before, we have to build a plan that includes the necessary work to make the client functional after removing the Windows operating system.

8.2.1 Client approach

The ITSO resident workstation would fit into either the Basic Office or Advanced Office workstation logical segment, as defined in 3.1.1, “Functional segmentation - Fixed function to general office” on page 50. And since the users use these workstations primarily for authoring technical papers using FrameMaker, as well as doing research using a Web browser, they map most closely to the Fat client type, as discussed in 4.4.2, “Logical segmentation - Thin, slim, or fat” on page 85.

8.2.2 Graphical environment

Another key decision that has to be made is what type of graphical windowing environment to use on the Linux client. Alternatives are discussed in 4.3.2, “Linux desktop environments” on page 76.

In this special case, we are not going to discuss advantages and disadvantages of one or the other environment; both of them would provide enough functionality.

But as GNOME is the default session manager for Red Hat distributions, we decided to stay with this choice for our sample scenario. The version that is included in Red Hat Desktop 4 at the time of this writing was GNOME 2.8.

8.2.3 Hardware

Another topic to be considered before migrating is the hardware that is in use.

In this case, no peripherals are connected to the client, and all other resources (such as printers, for example) are accessed through the network.

The migration PC platform for this example is listed in Table 8-1 on page 176.

Table 8-1 Hardware and descriptions

Hardware	Description
PC model	IBM Netvista Type 6759
Processor	Intel Pentium® III 866 MHz
Chipset	i810 integrated
Graphics card	i810
Sound card	i810
Optical drive	CD-ROM 24x IDE
Floppy	3.5 inch floppy drive
Harddisk	IDE Harddisk 40 GB

Because the graphics and sound controller are integrated with the chipset, we are expecting some additional work to identify and install appropriate Linux drivers for these devices. A check on the Intel Web site provided us with drivers for these two components. All other parts are known to be supported in Linux, because they are standard components.

8.2.4 Application continuity

In Table 8-2 on page 177, we show our application mapping to support this sample migration scenario.

Table 8-2 Application mapping to support sample migration scenario

Application on Windows	Application on Linux
Microsoft Internet Explorer	Mozilla Firefox
Microsoft Outlook	Novell Evolution with Novell Connector for Exchange
Windows Explorer	Nautilus
WinZip	FileRoller
ICQ	Gaim
Microsoft Word	OpenOffice.org Writer
Microsoft Excel	OpenOffice.org Calc
Microsoft Powerpoint	OpenOffice.org Impress
Adobe Reader	Acrobat Reader for Linux
Adobe FrameMaker	Adobe FrameMaker (via Windows Terminal Services)
Jasc Paint Shop Pro	The GIMP

8.2.5 Windows networking

Windows networks can become quite complex, especially when using roaming profiles, Active Directory structures, or when combining several trusted domains. In our environment, none of these features are in use. From a network services integration point of view, this makes the switch to Linux relatively easy.

Our migration clients will need to mount existing Windows file server shares and be able to use existing printers also shared by the Windows domain servers. Therefore, we need the domain name and a valid user for authentication to the existing ITSO domain.

The distinct names in this domain are:

- ▶ Domain name = ITSOAUSNT
- ▶ Primary domain controller = ITSONT00
- ▶ Backup domain controllers = ITSONT01, ITSONT02, and ITSONT03
- ▶ Usernames for residents = ausresxx

We will use methods described in detail in 4.2, “Integrating with existing network services” on page 69, and in Chapter 9, “Integration how-tos” on page 195, to complete network services integration of the Linux client.

8.3 Performing the migration

In this section, we discuss performing the migration.

8.3.1 Basic installation tasks

The first thing to do is complete a base installation from the distribution CDs. In the case of Red Hat Desktop, the CD set contains four CDs. Special packages for the desktop are available from the Red Hat Network. These contain third-party applications such as Acrobat Reader or Real Player and a set of high-quality truetype fonts from AGFA.

During installation, we accepted the default selections for application packages as defined by the installation program. On our target desktop hardware, the Red Hat Anaconda installation program completed the base installation without any problems. Also, the Red Hat hardware probing library called kudzu worked as expected. All hardware components were detected correctly. The correct modules for the integrated video and audio chipset were loaded successfully. The network card driver was loaded, and the network was activated successfully at the end of the installation. The system successfully received a DHCP lease from the DHCP server in the ITSO network. Name resolution worked as well, without any extra configuration steps.

Setting up remote administrative access using VNC

Once the installation completed, we chose to immediately modify our client setup to provide remote terminal access to it from other workstations in the network using VNC. This way we could complete any additional administrative tasks to complete the install remotely.

To begin this process, we first checked that the SSH daemon was running and that every connection through VNC gets a complete desktop. One way to enable this for the root user is to modify the file \$HOME/.vnc/xstartup, as shown in Example 8-1.

Example 8-1 Enabling VNC connections

```
#!/bin/sh

# Uncomment the following two lines for normal desktop:
unset SESSION_MANAGER
exec /etc/X11/xinit/xinitrc
.....
```

Also, we had to enable Xdmcp in /etc/X11/gdm/gdm.conf and set up a session in /etc/sysconfig/vncservers.

After we start by executing the command `service vncserver start`, we can connect to any VNC client and work on the complete graphical interface.

Important: Allowing complete access through VNC can be dangerous, because it enables connecting to the local X server from any place. If it is needed, editing `/etc/hosts.allow` or `/etc/hosts.deny` can allow you to limit who can access the workstation.

Preparing to integrate with Windows network services

The next step in our migration is to connect the client to the existing networking services. In our existing environment, we integrate with Windows NT and Windows 2000 servers, so we follow the instructions in Chapter 9, “Integration how-tos” on page 195.

Our first goal is the ability to use the services of the existing Windows domain, primarily user authentication. In order to be able to log on to the Linux client using Windows usernames and passwords, it is necessary to set up winbind.

We edited `/etc/samba/smb.conf` for a proper setup in our network environment.

Example 8-2 Changes to smb.conf for Linux client domain authentication

```
[global]
    workgroup = ITSOAUSNT
    security = domain
    password server = ITSONT00,ITSONT01,ITSONT02,ITSONT03
    ...
    ...
    winbind separator =
    idmap uid = 10000-20000
    idmap gid = 10000-20000
    winbind enum users = yes
    winbind enum groups = yes
    template homedir = /home/%D+%U
    template shell = /bin/bash
```

8.3.2 Integrating existing network services

After the winbind daemon has started, the command `wbinfo -u` should deliver the current user list of the Windows domain. But before this works, it is necessary to join the domain with the Linux client. As long as there is no machine account in the domain for the client, it is not possible to fetch the list of users using winbind. Therefore, we have to join the domain with the following command:

```
net join -a ITSOAUSNT -U administrator
```

Important: Do not forget to join the domain with the command `net join -a <domainname>`. This creates a valid machine account on the domain controller and gives the Linux client a valid SID for authentication.

Because we are now able to read the user database on Windows, only two more steps are needed to log in on the Linux client with an existing account managed by the existing Windows domain.

Next, `nsswitch.conf` is changed according to 9.2, “Using winbind to make domain users known locally” on page 199, which delivers the mapping of Windows users and Groups to the Linux uids and gids.

Example 8-3 Changes to /etc/nsswitch.conf

```
passwd: files winbind
group: files winbind
```

By performing the command `getent passwd` for testing purposes, all users should be listed with their correct uids and gids.

Now it is time to enable the login process to use the mapped users. This is done by using two different PAM modules, which is described in detail in 9.4.1, “How to authenticate users using winbind and PAM” on page 208.

Because we use Red Hat Desktop for our pilot migration, all information in “Winbind and PAM on Red Hat Desktop” on page 208 applies.

At first, we set up the `pam_winbind` and the `pam_smb_auth` module, which allow system authentication with a user of the domain ITSOAUSNT. In order to enable this module, we edited the `/etc/pam.d/system_auth` file, as in Example 8-4.

Example 8-4 Edit the /etc/pam.d/system_auth file

```
 #%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth      required      /lib/security/$ISA/pam_env.so
auth      sufficient   /lib/security/$ISA/pam_unix.so likeauth nullok
auth      sufficient   /lib/security/$ISA/pam_winbind.so use_first_pass
auth      sufficient   /lib/security/$ISA/pam_smb_auth.so use_first_pass nolocal
auth      required      /lib/security/$ISA/pam_deny.so

account  required      /lib/security/$ISA/pam_unix.so
account  sufficient   /lib/security/$ISA/pam_winbind.so
.....
```

It is now possible to authenticate both with a local or a domain user account. A first test on the console shows that it is working, but at this point, logging in on the GNOME welcome window generates an error.

The reason for this problem is that at the time of the first login, there is no home directory for the associated user ID. Thus, GNOME had no file system path in which to create the personalization and settings files during first login. To solve this problem, we used another pam module, pam_mkhomedir. By using this module, a home directory is created at logon time, if it does not exist yet.

It is necessary to add the following line in the /etc/pam.d/system_auth file.

Example 8-5 Addition of pam_mkhomedir entry in /etc/pam.d/system-auth

```
%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth      required      /lib/security/$ISA/pam_env.so
auth      sufficient   /lib/security/$ISA/pam_unix.so likeauth nullok
auth      sufficient   /lib/security/$ISA/pam_winbind.so use_first_pass
auth      sufficient   /lib/security/$ISA/pam_smb_auth.so use_first_pass nolocal
auth      required      /lib/security/$ISA/pam_deny.so

account  required      /lib/security/$ISA/pam_unix.so
account  sufficient   /lib/security/$ISA/pam_winbind.so
.....
session  optional     /lib/security/$ISA/pam_mkhomedir skel=/etc/skel umask=0022
```

Now, we can enter the username and password on the GNOME welcome window, a home directory is created, and the complete GNOME desktop is available.

We log on using the ID:

ITSOAUSNT+AUSRES06

The login username has the format:

<domainname:separator:username>

So, the GNOME desktop shows an icon with ITSOAUSNT+AUSRES06's home. Unfortunately, it is not possible to separate the domain and username on a standard GNOME installation; the winbind separator will always be visible.

Mounting Windows file shares

After enabling domain user logon, it should also be possible to use the file shares on the Windows servers. Our example client needs to map three separate file shares using the same credentials. We have to store the user credentials in a file

in order to avoid having to interactively use the `mount` command on each login. The `smbmount` command allows the use of a credentials file. For security reasons, we placed the credentials file in root's home directory and set the file permission such that only root has access to the file. The file `/root/.credentials` looks like Example 8-6.

Example 8-6 Credentials file for mounting shares

```
username = ausres06
password = password
domain = itsoausnt
```

The most logical place for such a mount execution is in `/etc/fstab`, where it is possible to define the file system to `smbfs`, and the file shares would be mounted at the same time as the local file systems.

Example 8-7 Mounting file shares in /etc/fstab

/dev/hd1	/	reiserfs	acl,user_xattr	1 1
/dev/hda5	/scr	auto	noauto,user	0 0
/dev/hda2	swap	swap	pri=42	0 0
.....				
/dev/fd0	/media/floppy	subfs	fs=floppyfs,procuid,nodev,nosuid,sync	0 0
//itsont05/data	/shares/data_g	smbfs	credentials=/root/.credentials	0 0

This works fine unless the Windows file share uses nonstandard characters. In contrast to UNIX-like operating systems, Windows allows you to use space characters and non-ASCII characters (such as German umlauts) in the names of file shares. You might run into problems in the case where the Windows file shares contain special characters. For example, the `/etc/fstab` file uses the space character as a delimiter, so if a share contains a space, it takes the following string as a mount point. So, while mounting the “project data” file share, the mount command complains about several errors in the `fstab` file, as it tries to use `data` as a mount point.

Tip: When mounting Windows file shares in a Linux file system, it is very reasonable to check the names of the shares for special characters. Renaming them to be UNIX-compliant can save a lot of time and annoyance.

Thus, we had to use another way of mounting the file share, and we decided to extend the `/etc/rc.local` file with the corresponding `smbmount` commands.

Modifications to `rc.local` are shown in Example 8-8 on page 183.

Example 8-8 Extended rc.local file

```
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local

#mounting windows file shares
smbmount //itsont05/data /shares/data_g -o credentials=/root/.credentials,
gid=ITSOAUSNT+Domain\ Users,dmask=0775
smbmount //itsont02/"project data" /shares/projectdata_e -o credentials=/root/.credentials,
gid=ITSOAUSNT+Domain\ Users,dmask=0775
```

If using file shares with special characters here, it is necessary to put them in quotation marks, as shown in the last line of Example 8-8 above. The gid and dmask options of the **smbmount** command are used to ensure that the permissions are set in the right way for the Linux directories in which the shares are mounted. If they are left out, users would not have write permissions because of the default umask in the root directory. The permissions that are set on the Windows file server are inspected only after a user has full access to the mount point. It is evident that the **smbmount** does not override permissions on the server side.

Important: Be careful with permissions on the directories that you use as mount points; make sure that they fit to the permissions given on the Windows server. Using a gid in the command prevents file access from users who are not members of the group.

Printers

Printer services are provided via the smb protocol as well. We decided to use CUPS to integrate our network printers on the Linux clients.

Tip: If drivers for your printer are not included in the standard installation of CUPS, it might be possible to find the right ppd (Postscript Printer Description) driver file for your printer on the Internet. The correct ppd driver file for your printer should be added to the directory /usr/share/cups/model. Then after restarting CUPS, the new driver is available when adding a printer. Methods for finding a ppd driver file are:

- ▶ Search your printer manufacturer's Web site for Linux support or ppd driver file downloads.
- ▶ Purchase printing software supporting Linux:
<http://www.easysw.com/printpro>
- ▶ Try a Windows PPD file provided by Adobe:
<http://www.adobe.com/products/printerdrivers/winppd.html>
- ▶ See the CUPS home page for more links:
<http://www.cups.org>

By following the steps in “Create a printer using CUPS Web interface” on page 221, it was extremely easy to add the printers with the right credentials.

8.3.3 Application configuration and installation

As pointed out earlier in this chapter, the main application on the resident PC is Adobe FrameMaker. Interestingly enough, Adobe offered a Linux version some time ago, but as of Version 6.0 this product was cancelled. Because the client needs FrameMaker Version 7.0 with some special plug-ins, it is not possible to install this application on Linux. Thus, we have to find a way to handle this “unmigratable” application (see 4.7, “Unmigratable applications” on page 99, for a detailed discussion about strategies).

The solution we chose to support FrameMaker users on the migrated Linux client was to use a remote terminal services method using Windows Terminal Server (WTS). WTS makes it possible for a Linux client to connect using Remote Display Protocol (RDP) to an application session running on the WTS host. Other terminal services solutions for this type of application hosting requirement exist, but for this small scenario, we decided using WTS is satisfactory. Our next step was to install the application on the Terminal Server.

Because it is not guaranteed that every application can run on a Terminal Server, we ran several tests by starting FrameMaker several times in different sessions. We learned that our version of FrameMaker was not optimized for this scenario, since each FrameMaker application instance running on the WTS host consumed an equal amount of system memory.

Important: If an application is not optimized for terminal servers, it takes the same amount of memory for each instance. The server needs to have enough main memory to handle the expected Terminal Server session load.

The connection to the Terminal Server is made by tsclient, a GTK+ GUI for the rdesktop application. Using a launcher file on the desktop, it is possible to use a configuration file while executing tsclient. Our command syntax for starting the tsclient session is as follows:

```
tsclient -x $HOME/framemaker.tc
```

The file framemaker.tc contains all of the information for the connection, such as the IP address of the host and the username to connect to.

An example window capture of FrameMaker running inside a Terminal Server client on Linux is provided in Figure 8-1 on page 190.

Image manipulation

On the Windows-based client, we used Jasc Paint Shop Pro for manipulating images and taking screen captures. Because Paint Shop Pro is not available for Linux, we decided to use The GIMP as a functionally equivalent Linux application instead. This open source application is very mature and provides a good functional equivalent to Paint Shop Pro for the specific uses that the Windows client in this scenario needs. Further information can be found at:

<http://www.gimp.org>

The GIMP is included as part of the standard installation of Red Hat Desktop, so no extra configuration was necessary. A screen capture showing The GIMP running on the migrated desktop is shown on Figure 8-2 on page 191.

Browser

For our migration scenario, we decided to use the Mozilla Firefox Browser. Firefox is the browser-only project from the Mozilla Foundation. Because we do not need mail capabilities (these are provided by Novell Evolution), using the stand-alone browser offering from Mozilla seemed reasonable.

We recommend that you check important Web sites or especially Web clients for compatibility with the browser before the migration. Some Web sites use ActiveX® controls or special JavaScript™ functions, which are only supported by Microsoft Internet Explorer. Because this is not the case for our migration client, it is easy to make the switch and use Mozilla Firefox instead of Internet Explorer.

E-mail client

To make our client migration scenario more realistic, we added the requirement that existing clients use Microsoft Outlook, which connects to a Microsoft Exchange 2000 server. Our migration task includes migrating from the Windows-based Outlook client to a Linux-based messaging client that provides similar functionality and that integrates with the existing Exchange 2000 messaging system.

Novell Evolution with Novell Connector for Exchange

Our migration solution consists of using the Novell Evolution e-mail client in conjunction with the Novell Connector for Microsoft Exchange Server. The Evolution e-mail client has an architecture that supports building special connectors for different groupware applications. A connector for Microsoft Exchange 2000 and 2003 was developed and was released under the terms of the GPL in May 2004. A key factor in enabling the development of the Exchange connectors was Microsoft's inclusion of a Web interface to Exchange called Outlook Web Access (OWA). OWA uses WebDAV for communication to the Exchange Server message store. So it became possible to develop Exchange connectors by implementing a WebDAV interface, instead of having to reverse engineer the proprietary MAPI protocol of Exchange.

Restriction: Only the Exchange versions 2000 and 2003 support WebDAV connections. As of the writing of this book, a connector supporting native Linux client access to Exchange Server 5.5 was still not available.

We installed the required Novell Connector for Microsoft Exchange Server RPMs that were delivered with Red Hat Desktop and found it to be quite easy to set up a connection to the Exchange Server. There were some requirements to get this working, mainly enabling Outlook Web Access on the server.

Important: Before using the Exchange connector, you need to check prerequisites. It is necessary to start the Exchange virtual server in the HTTP section of the Exchange system manager. You also need to enable HTTP protocol for the users. This is done in the user settings through the Active Directory Users console.

More information about the Novell Connector for Microsoft Exchange Server can be found on the Novell Evolution page at:

<http://www.novell.com/products/evolution>

Settings migration

In order to keep the user productive, we want to maintain their settings and data. Users are supposed to keep their documents on a network share, so we do not need to search the entire hard drive for documents. In case users keep some notes or important scratch files on their local machine, we still move data from the My Documents and Desktop directories. E-mail messages are stored on the Exchange Server, so we need to only move the account information. User defined bookmarks and the Internet home page should also be migrated to make the migration easier.

Manual migration

First, we look at performing this migration manually. The Windows version of Firefox can import customized data such as bookmarks and the home page from Microsoft Internet Explorer, which the Linux version cannot. Thus, on the Windows machine, we install Firefox and launch it to start Firefox's import feature. Next, we enable Windows sharing if it is not already enabled and share out the My Documents and Desktop directories. We also share out Firefox's settings directory, which is a subdirectory in the user's Application Data directory, in the Mozilla folder. The full path to this directory on our machine was:

```
C:\Documents and Settings\AUSRES06\Application Data  
 \Mozilla\Firefox\Profiles\xyjib4ms.default
```

As the last step on the Windows machine, we launch Microsoft Outlook and ensure that all e-mail is stored on the server. If there are any local folders, we create a new folder on the server and copy the mail onto the server.

On the Linux client, we now use Nautilus to connect to the Windows shares. We copy the contents of the My Documents directory into a directory named Documents in the user's home directory. We then copy all of the data files (but not the shortcuts) from the old Desktop share onto the user's desktop. Next, we launch Firefox to initialize a new profile. Then, we find the new profile directory, which is in the user's home directory, in the .mozilla folder. The full path on our machine was:

```
/home/ITSOAUSNT+AUSRES06/.mozilla/firefox/xdch2a44.default
```

Now we copy all the files from the old profile share on the Windows machine into this folder, overwriting all files that are currently there. Finally, we launch Evolution and add a new account to the Exchange Server.

Automated migration

Because there are several authors using a similar environment, we do not want to migrate each desktop manually. Instead, we use Progression Desktop to automate the migration. More information about Progression Desktop is found in Appendix C, "Automating desktop migration using Versora Progression Desktop"

on page 277. Because we have more than ten workstations, we decide to use the command line interface with a template to make the migration step easier. We also write a simple script to make the migration two simple steps.

First, we create a template, which contains all of the data we want to migrate and save that on our network share. Then for the Windows store operation, we use the following batch file to install the .NET framework and Progression Desktop and then store the settings into a Platform Neutral Package (PNP) onto our network share.

Example 8-9 Batch file to install and run Progression Desktop on Windows

```
cmd /c \\itsont05\data\ProgressionDesktop\Windows\dotnetfx.exe /q:a /c:"install /q /1"
cmd /c msieexec /quiet /i \\itsont05\data\ProgressionDesktop\Windows\zf7fg42j9.msi
"c:\Program Files\Versora\Progression Desktop\ProgressionDesktopCore.exe" --store
--file=\\itsont05\data%\UserDomain%+%\UserName%.pnp
--template=\\itsont05\data\ITSOtemplate.xml
```

Next, we create a bash script for the Linux apply operation, which temporarily installs Mono on the desktop (because Red Hat Desktop does not ship with Mono), and we run Progression Desktop to apply the user's settings from the network share. Finally, the bash script cleans up the machine by removing Mono.

Example 8-10 Bash script to install and run Progression Desktop on Linux

```
#!/bin/bash
/shares/data_g/monoinstaller/mono-1.1.8.2_1-installer.bin \
--mode unattended --prefix /var/tmp/mono --AddMonoToPath 0
/var/tmp/mono/bin/mono /shares/data_g/ProgressionDesktop/Linux/ProgressionDesktopCore.exe \
--apply --file=/shares/data_g/$USER.pnp --template=/shares/data_g/ITSOtemplate.xml
rm -rf /var/tmp/mono
rm -f ~/Desktop/Mono.desktop
```

To migrate any machine from Windows to Linux, all we need to do is run the first batch file on the Windows machine, and, once it finishes, run the bash script on the Linux destination.

Customization

There is an extremely high degree of flexibility in modifying the look and feel of desktop Linux. For our client migration scenario, we chose to modify just a few settings as a way of demonstrating some possibilities.

The first thing we did was to adjust the default background color and insert the ITSO Redbooks image on the background. You can do this easily by using the background settings application for the GNOME Desktop. Another possibility,

especially to set this desktop background image as a mandatory value, is the use of the gconftool-2 with a command syntax as in Example 8-11.

Example 8-11 Using gconftool-2 to replace the desktop background image for all users

```
# gconftool-2 --direct --config-source \
  xml:readwrite:/etc/gconf/gconf.xml.mandatory --type string --set \
  /desktop/gnome/background/picture_filename itso-logo.png
```

Further information about how to change values in the GNOME configuration database can be found in “Desktop personalization: GNOME Desktop” on page 317. Specific details about using gconftool-2 to make mandatory changes can be found in 7.1.2, “GNOME lockdown options” on page 151.

You can create another nice effect by inserting another logo for the menu button. By default, Red Hat has replaced the GNOME logo with its own custom image. We replaced this by editing the following file:

```
/usr/share/pixmaps/redhat-main-menu.png
```

While the name of this file is set in GConf, we decided to take the easy way out and just rename our ITSO logo file to redhat-main-menu.png. Of course, it has to be 48x48 pixels large, and for a better look, the logo had to be made transparent in The GIMP.

The final customization was the creation of links to the mapped Windows file shares. For convenience, we renamed them to the equivalent drive letters.

The output of these customizations can be viewed in Figure 8-5 on page 194.

8.3.4 Screen captures: Client migrated to Linux

On the following pages, we provide screen captures of the most important applications now running on the Linux client:

- ▶ Adobe FrameMaker on Windows Terminal Server - Figure 8-1 on page 190
- ▶ The GIMP on Linux Client - Figure 8-2 on page 191
- ▶ Novell Evolution on Linux Client connected to Microsoft Exchange 2000 server - Figure 8-3 on page 192
- ▶ Mozilla Firefox on Linux client - Figure 8-4 on page 193
- ▶ Showing the new desktop, customized for ITSO - Figure 8-5 on page 194

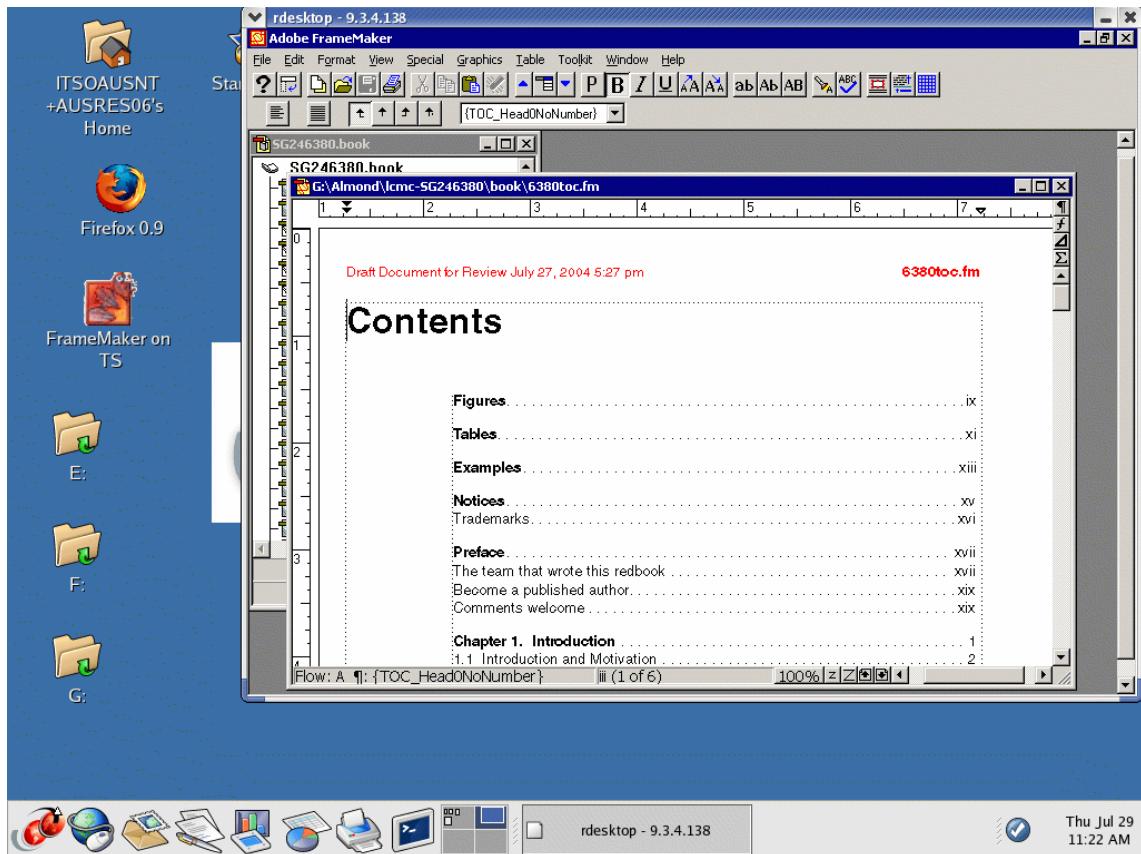


Figure 8-1 FrameMaker on Windows Terminal Server session using rdesktop¹

¹ Adobe product screen shot reprinted with permission from Adobe Systems Incorporated.

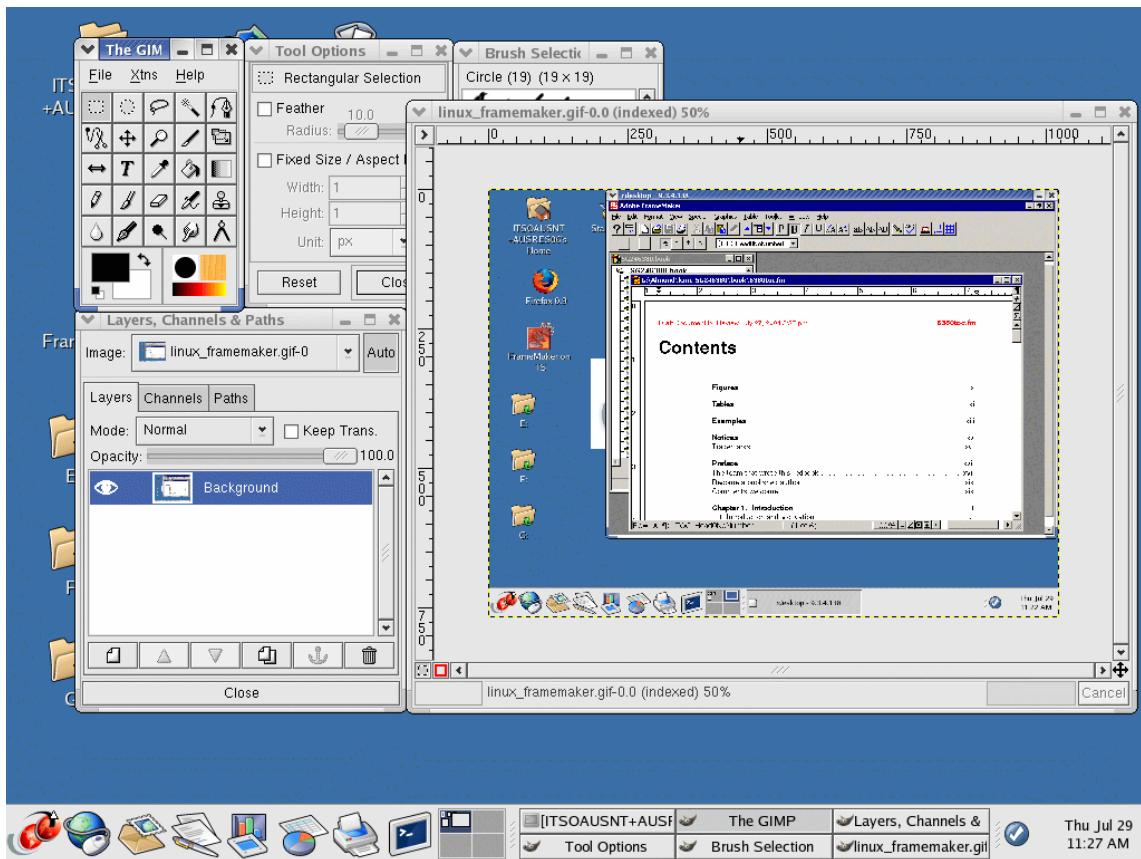


Figure 8-2 Running The GIMP and taking a screen capture of FrameMaker

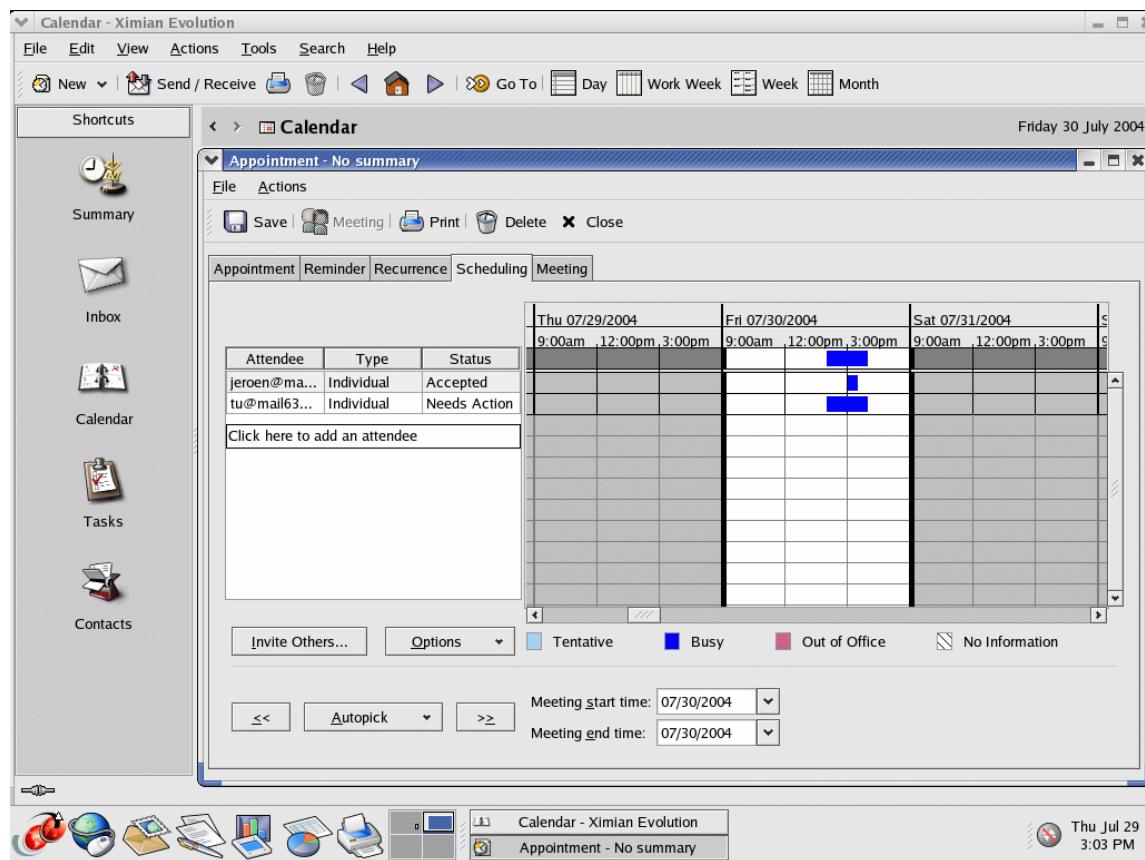


Figure 8-3 Novell Evolution connected to Microsoft Exchange 2000



Figure 8-4 Mozilla Firefox on the Linux client



Figure 8-5 New Linux desktop showing organizational customizations for ITSO



Integration how-tos

This chapter demonstrates additional methods for integrating Linux clients into an existing Windows domain (NT4 or Active Directory). Many integration issues are covered, including mounting home directories from SMB shares at logon.

There are two ways to authenticate users to a Windows domain:

- ▶ Using winbind provided by Samba
- ▶ Using LDAP to connect directly to Active Directory (this needs changes to the Active Directory schema)

The sections in this chapter are:

- ▶ 9.1, “How to join a Windows domain” on page 196
- ▶ 9.2, “Using winbind to make domain users known locally” on page 199
- ▶ 9.3, “How to use LDAP to connect to Active Directory” on page 204
- ▶ 9.4, “Pluggable Authentication Modules and the domain” on page 207
- ▶ 9.5, “How to mount a share on the Linux client” on page 214
- ▶ 9.6, “Automatically mounting home directories at logon” on page 216
- ▶ 9.7, “How to use network printers in the domain” on page 219

All examples and how-tos in this section assume at least Samba Version 3.0. Some of the functionality is either not present or not mature in earlier versions.

9.1 How to join a Windows domain

Most how-tos discuss in detail how to add Linux *servers* to a domain. This how-to describes adding a Linux *client* to a domain. Adding a client to an NT4 domain is different from adding a client to an Active Directory domain. We discuss both in detail.

In most examples in this section, we use a domain AD6380 with Primary Domain Controller SMB3LAB26 and Backup Domain Controller SMB3LAB27. In some examples (using Windows 2003 Active Directory), we use the domain AD6380 with domain controller W2K3AD.

9.1.1 Joining an NT4 domain

We use Samba to connect to the domain. The minimum smb.conf looks like Example 9-1.

Example 9-1 smb.conf for joining NT4 domain

```
[global]
    workgroup = AD6380
    security = domain
    password server = SMB3LAB26,SMB3LAB27
```

Replace the example domain and password servers with your own domain name and the correct names or addresses for the primary (and backup) domain controllers.

You can then join the domain using:

```
net join -S SMB3LAB26 -U administrator
```

Replace SMB3LAB26 with the name (or IP address) of your own primary domain controller and use any domain account that has the right to add machines to the domain. The command prompts you for the password of the domain account “administrator”.

More details about joining an NT4 domain can be found in the current Samba-3 release version of the Samba-HOWTO-Collection. The collection can be found at the following location:

<http://samba.org/samba/docs/>

9.1.2 Joining an Active Directory domain

In this case, we need both Samba and Kerberos to connect. We need Kerberos to authenticate against a Windows 200x KDC.

In the example, we use domain AD6380.LOCAL with AD server SMB3LAB26.

The minimum smb.conf contains the lines given in Example 9-2.

Example 9-2 smb.conf for joining Active Directory domain

```
[global]
    realm = AD6380.LOCAL
    workgroup = AD6380
    security = ads
    password server = SMB3LAB26
```

For the realm, take care to use the correct case, since Kerberos is case sensitive. Use the fully-qualified name (that is, AD6380.LOCAL) for the realm and the short name for the workgroup; otherwise, warnings might appear.

The minimum krb5.conf looks like Example 9-3.

Example 9-3 krb5.conf for joining Windows 200x Kerberos realm

```
[libdefaults]
    default_realm = AD6380.LOCAL

[realms]
    AD6380.LOCAL = {
        kdc = SMB3LAB26:88
        admin_server = SMB3LAB26
    }

[domain_realm]
    .kerberos.server = AD6380.LOCAL
```

Make sure the name of the Kerberos server is in the DNS in such a way that a reverse lookup on the IP address returns the NetBIOS name of the KDC or the NetBIOS name followed by the realm. It should not return the host name with a domain attached. The easiest way to ensure this is by putting it in the /etc/hosts entry.

Since Kerberos tickets are heavily time dependent, it is important to make sure that the AD server and clients have the same time. Because Windows clients get their time from the domain controller, the Linux client can use Samba tools to get the time from the server as well. You do this using the `net time set` command. This fetches the time from the AD server and sets the local clock.

Important: Make sure to use the right version of Kerberos (either MIT or Heimdal). For AD on Windows Server® 2003, Heimdal should be at least at Version 0.6 and MIT at least at Version 1.3.1. Also check to see if the kinit and klist commands actually come from the correct implementation and not from Java. The Java kinit is known not to work.

When using Novell Linux Desktop, the heimdal and heimdal-tools packages might not be installed by default. These are needed to connect to AD on Windows 2003 Server.

Important: Make sure clients and the Active Directory (or Kerberos) server have the same time within a defined allowed skew.

Microsoft Windows servers use Simple Network Time Protocol (SNTP), which is a simplified version of the (UNIX/Linux) Network Time Protocol (NTP). The packets and formats are identical and thus can be used together. This means that the Linux clients can be synchronized with the AD servers using ntp.

You can test the Kerberos configuration by doing a **kinit USERNAME@REALM** to make sure the password is accepted by the Windows 200x KDC.

Important: Only newly created accounts in ADS or accounts that have had their passwords changed once since migration work. If an account stems from before the migration (or installation in the case of Administrator), the **kinit** command returns a message about a wrong encryption. Changing the password of the account resolves this problem.

To actually join the AD domain, you execute the following:

```
net ads join -U administrator
```

This prompts for the administrator password, for example, joining client machine client1 to the domain AD6380 using administrative user idsadmin (see Example 9-4).

Example 9-4 Example of joining client1 to domain AD6380

```
[root@client1 root]# net ads join -U idsadmin
idsadmin password:*****
Using short domain name -- AD6380
Joined 'CLIENT1' to realm 'AD6380.LOCAL'
```

Joining a particular organizational unit can be done by first getting the correct credentials and then joining the unit. For example, if you want to join the domain (that is, create a computer account) in a container called Clients under the organizational directory Computers/ITSO, you execute:

```
kinit Administrator@AD6380.LOCAL  
net ads join "Computers\ITSO\Clients"
```

Important: Since Windows 2003 uses SMB signing, you might have to put the following line in the smb.conf file when trying to join a Windows 2003 ADS.

```
client use spnego = yes
```

To check a join to an AD domain (at all times), use the command:

```
net ads testjoin
```

More details about joining an Active Directory domain can be found in the *Samba HOWTO collection section 6.4*:

<http://samba.org/samba/docs/Samba-HOWTO-Collection.pdf>

9.2 Using winbind to make domain users known locally

After joining a domain as described in 9.1, “How to join a Windows domain” on page 196, it is necessary to add all domain accounts to the Linux client if the domain accounts are going to log on to the client. In smaller domains, this is not a problem, but it is generally not good administrative practice. This means that adding an account to the domain means adding an account to all clients. That negates the advantage of using a domain.

In this section, we describe how to use winbind to avoid creating domain accounts locally. The winbind daemon takes care of translating domain accounts to uids and gids to the client operating system.

9.2.1 Common implementation of winbind

The winbind daemon reads its settings from the smb.conf file. You must add the lines in Example 9-5 to the smb.conf to enable winbind to function.

Example 9-5 Lines added to smb.conf for winbind

```
[global]  
winbind separator = +  
idmap uid = 10000-20000  
idmap gid = 10000-20000  
winbind enum users = yes
```

```
winbind enum groups = yes
template homedir = /home/%D%U
template shell = /bin/bash
```

The separator means that accounts are written as “AD6380+Administrator” for the Administrator account in the domain AD6380. The other entries give the uid and gid range to use for domain accounts and what to use as shell and home directory in case of actual logon. We have found that a plus sign (+) as a winbind separator works most successfully within the Linux environment.

Alternatively, it is possible to not use a separator and domainname at all. This way the default domain is used. To enable this, add the following line to the smb.conf file:

```
winbind use default domain = yes
```

and probably change the template homedir to:

```
template homedir = /home/%U
```

All domain users show up without the domainname and separator. This might cause problems if local users and domain users have the same name.

Tip: Use the default domain option when in doubt. The separator and domainname can lead to problems in applications. Using pam_mount for the default domain avoids a lot of problems.

The idmap uid and gid entries tell winbind which range it can use to generate uids and gids for domain accounts. Choose a range that does not overlap the local users.

Important: If you have an extremely large domain, you might not be able to incorporate all users in your idmap uid range. This is only problematic if you use winbind on a domain client, which all users use through the network.

After these changes, the winbind daemon has to be started. Usually, this is done using the command:

```
/etc/init.d/winbind start
```

Make sure the winbind daemon starts on system boot. On most distributions (certainly on Red Hat Enterprise Linux and Novell Linux Desktop), this is done using the following command:

```
chkconfig winbind on
```

The winbind functionality can be tested by using the `wbinfo` command. Executing with the `-u` option shows all domain accounts translated and the `-g` option shows all domain groups.

Example 9-6 Example output of wbinfo command

```
[root@client1 root]# wbinfo -u
AD6380+Administrator
AD6380+Guest
AD6380+TsInternetUser
AD6380+idsadmin
AD6380+krbtgt
AD6380+HOST/client1
AD6380+SMB3LAB26$
```

Important: The service called name service caching daemon (nscd) interferes with proper functioning of winbind. *Never* run nscd on systems where winbindd runs.

With these settings, the winbind daemon is able to translate domain accounts to local users. We tell the system to actually use winbind by putting a reference in the Name Service Switch (NSS) configuration.

The configuration file for NSS is `/etc/nsswitch.conf`. This file contains entries such as:

```
passwd: files example
```

This means that when a request for a password comes in, first a routine in `/lib/libnss_files.so` is called to resolve it and then a routine in `/lib/libnss_example.so`. Once the request is resolved, the result is returned to the application. To add winbind to the resolve path, we change the following lines in `/etc/nsswitch.conf` in the following manner:

```
passwd: files winbind
group: files winbind
```

To test this, execute the command:

```
getent passwd
```

This returns not only the entries in the file `/etc/passwd`, but also the domain accounts that were translated by winbind.

Example 9-7 Partial example output of getent passwd using winbind in NSS

```
[root@client1 root]# getent passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
```

```
.....  
AD6380+Administrator:x:10000:10000:Administrator:/home/AD6380+Administrator:/bin/  
n/bash  
AD6380+Guest:x:10001:10000:Guest:/home/AD6380+Guest:/bin/bash  
AD6380+TsInternetUser:x:10002:10000:TsInternetUser:/home/AD6380+TsInternetUser:  
/bin/bash  
AD6380+idsadmin:x:10003:10000:idsadmin:/home/AD6380+idsadmin:/bin/bash  
AD6380+krbtgt:x:10004:10000:krbtgt:/home/AD6380+krbtgt:/bin/bash  
AD6380+HOST/client1:x:10005:10001:client1:/home/AD6380+HOST/client1:/bin/bash  
AD6380+SMB3LAB26$:x:10006:10002:SMB3LAB26:/home/AD6380+SMB3LAB26_:/bin/bash
```

Because of the way that winbind works (namely, handing out the first uid in the range to the first domain user it has to authenticate and then storing the mapping), the mapping between domain user and uid is not the same on all clients. This leads to problems if the clients are using a shared file system based on NFS. The mechanisms discussed in 9.2.2, “Alternate implementations of winbind” on page 202 avoid this problem.

Important: When winbind fails because the Kerberos connection is lost because of time skew, the daemon has to be restarted after fixing the time skew.

The winbind daemon translates users and generates home directories and shells. These home directories are not created by winbind. We show you how to do this in 9.4.3, “PAM and home directories” on page 212.

Using the winbind daemon creates the connection to the domain for translating users. To enable domain users to log on to the Linux client, we need a change in the PAM configuration, as shown in the section 9.4.1, “How to authenticate users using winbind and PAM” on page 208.

9.2.2 Alternate implementations of winbind

The primary difference in the alternative implementations is the way that uid and gid are generated by winbind. In the previous section, winbind chooses uid and gid for each user from the range provided. This generally happens in the order used, which means it is not the same on all clients in a domain.

This section describes two ways of handling this:

- ▶ idmap_rid backend to generate from Active Directory information
- ▶ idmap_ad backend to obtain uid and gid from an AD with an extended schema (for example, extended with Services for UNIX)

These idmap back-ends are quite new and, thus, still a bit experimental. But both remove one of the disadvantages of winbind as a solution in enterprise environments.

idmap_rid backend

This back-end calculates a uid and gid for domain users based on the domain SID. To enable this option, include the lines given in Example 9-8 in the smb.conf file.

Example 9-8 Example of entries in smb.conf to use the idmap_rid backend

```
idmap uid = 10000-20000
idmap gid = 10000-20000
idmap backend = idmap_rid:AD6380=15000-20000
allow trusted domains = no
```

Use these settings and set the default domain to no to clearly distinguish the domain accounts. Part of the output of **getent passwd** looks like Example 9-9.

Example 9-9 Partial output of getent passwd when using idmap_rid

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
...
AD6380+administrator:x:15500:15513:Administrator:/home/administrator:/bin/bash
AD6380+guest:x:15501:15514:Guest:/home/guest:/bin/bash
AD6380+w2k3ad$::x:16003:15516:W2K3AD:/home/w2k3ad_:/bin/bash
AD6380+krbtgt:x:15502:15513:krbtgt:/home/krbtgt:/bin/bash
AD6380+testuser:x:16108:16121:Test User:/home/testuser:/bin/bash
AD6380+rhdt4$::x:16112:15515:rhdt4:/home/rhdt4_:/bin/bash
AD6380+rhdt4a$::x:16113:15515:rhdt4a:/home/rhdt4a_:/bin/bash
AD6380+ldapbind:x:16117:15513:ldapbind:/home/ldapbind:/bin/bash
AD6380+nld9$::x:16120:15515:nld9:/home/nld9_:/bin/bash
```

It is clear from the output in Example 9-9 that this is no longer a simple algorithm for picking the uid and gid numbers one after the other in the allowed range. The major advantage of this is that the mapping is the same on all clients within the AD6380 domain.

idmap_ad backend

In the next section about LDAP, the concept of the Microsoft Services for UNIX (SFU) is explained. Using the SFU, the schema for the Active Directory is extended with uid and gid information. Using the idmap_ad backend, it is possible to get at this information and use that with winbind.

This way the mapping on all clients is consistent, because the mapping is actually done in the Active Directory.

The idmap_ad backend is enabled using the following line in the smb.conf file:

```
idmap backend = idmap_ad
```

9.3 How to use LDAP to connect to Active Directory

Active Directory is an implementation of the Lightweight Directory Access Protocol (LDAP). Thus, we can use an LDAP client to connect to an AD to gather information about users. However, since the AD is not designed to store UNIX/Linux users, some of the necessary objects are missing in the default schema. One of the first steps is to extend the schema with the needed object-classes.

A simple way to extend the schema is to install the “Services for UNIX” (SFU) from Microsoft. These include the object-classes needed to store the information needed.

Install SFU using the default settings, as follows:

1. Extract files to a directory that is not C:\SFU, for example, C:\Temp\SFU.
2. Run setup.exe.
3. Accept Standard Installation.
4. On the Security Settings window, leave both options blank.
5. Choose **Local Username Mapping Server** and **Network Information Services**.
6. Choose the domain name.
7. Reboot the AD server.

This installation has extended the schema of AD to include fields such as uid and gid.

The next step is to create a user in the Active Directory to be used by LDAP clients to bind to the directory. Choose a highly secure password and set it to not expire.

Given the schema extensions, it is now possible to provide groups with a gid field and users with uid, shell, and a home directory, as shown in Figure 9-1 on page 205.

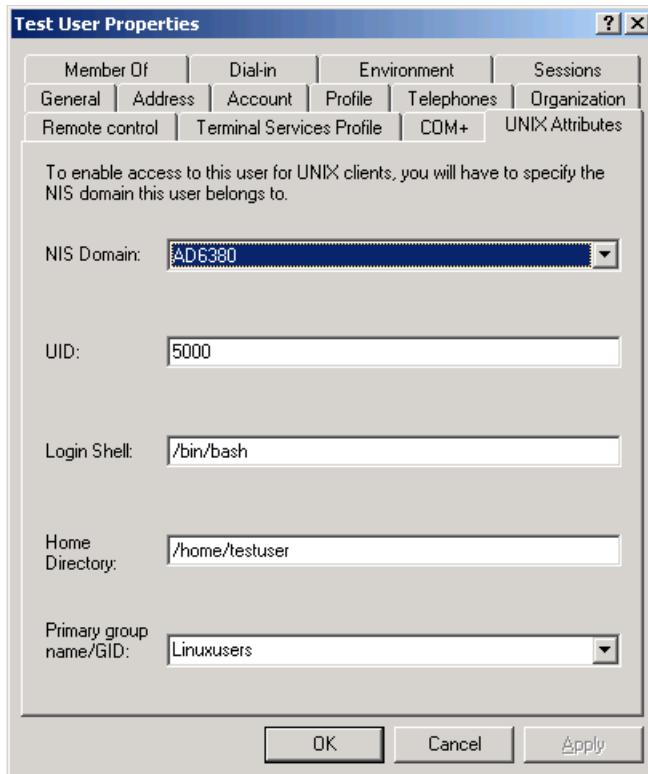


Figure 9-1 Active Directory User properties window with UNIX Attributes

In this example, we have created a user called testuser with UID 5000, a Login Shell, and a Home Directory. We have also created a group or Primary group name/GID of Linuxusers with gid 5000.

More information about Services for UNIX and integrating Active Directory as the directory for a Linux or UNIX client can be found in the document *Solution Guide for Windows Security and Directory Services for UNIX* available from Microsoft.

To enable the system to use LDAP to connect to the Active Directory, you must configure the LDAP configuration file, /etc/ldap.conf. This configuration file contains the following information:

- ▶ Host of the directory, that is, the IP address of the Active Directory server (if more than one, separated by spaces)
- ▶ The base node in the directory to start the search
- ▶ The bind user to use to connect to AD

- ▶ The password for the bind user (in plain text, so you secure the configuration file)
- ▶ Scope to search
- ▶ SSL enablement
- ▶ A number of entries to map domain schema objects to objects needed by the NSS

An example configuration file is shown in Example 9-10.

Example 9-10 Partial /etc/ldap.conf file

```
host 192.168.100.110
base cn=Users,dc=AD6380,dc=LOCAL
binddn cn=dapbind,cn=Users, dc=AD6380,dc=LOCAL
bindpw Welcome2006
scope sub
ssl no
nss_base_passwd cn=Users,dc=AD6380,dc=LOCAL?sub
nss_base_shadow cn=Users,dc=AD6380,dc=LOCAL?sub
nss_base_group cn=Users,dc=AD6380,dc=LOCAL?sub
nss_map_objectclass posixAccount user
nss_map_objectclass shadowAccount user
nss_map_attribute uid sAMAccountName
nss_map_attribute uidNumber msSFU30UidNumber
nss_map_attribute gidNumber msSFU30GidNumber
nss_map_attribute loginShell msSFU30LoginShell
nss_map_attribute gecos name
nss_map_attribute userPassword msSFU30Password
nss_map_attribute homeDirectory msSFU30HomeDirectory
nss_map_objectclass posixGroup Group
nss_map_attribute uniqueMember msSFU30PosixMember
nss_map_attribute cn cn
```

The following entries are changed in the NSS configuration to enable the system to use the domain users through LDAP:

```
passwd: files ldap
group: files ldap
```

To test this, use the command:

```
getent passwd
```

This should now also show the testuser entry in the passwd output. This is shown in Example 9-11 on page 207.

Example 9-11 Partial output of getent passwd using ldap in NSS

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
...
testuser:ABCD!efgh12345$67890:5000:5000:Test User:/home/testuser:/bin/bash
```

Important: Notice how the users show up without a domain name and without a separator character. This also means that you should take care to prevent overlap, that is, the same user in both local files and in the domain.

So, instead of generating the uid and gid entries on each client, this method takes those entries from Active Directory. However, this means that these entries have to be put there in the first place.

The differences, advantages, and disadvantages of the two methods, winbind and LDAP, were already mentioned in 4.2, “Integrating with existing network services” on page 69.

Part of this section is based on the information available at:

<http://enterprise.linux.com/enterprise/04/12/09/2318244.shtml?tid=102&tid=101&tid=100>

9.4 Pluggable Authentication Modules and the domain

Once the Linux client is part of a domain (as described in 9.1, “How to join a Windows domain” on page 196) and knows about the domain accounts using either winbind (as described in 9.2, “Using winbind to make domain users known locally” on page 199) or LDAP (as described in 9.3, “How to use LDAP to connect to Active Directory” on page 204), we can configure the system to allow domain accounts to log on.

Pluggable Authentication Modules (PAM) is a system for abstracting authentication and authorization technologies. Using PAM modules, it is possible to change the way applications authenticate or authorize accounts without having to recompile the application.

In most distributions, the configuration of PAM is governed by files in /etc/pam.d/. Usually there is one file per application. So, changing authentication modes for an application is as simple as adding a corresponding module to the config file.

9.4.1 How to authenticate users using winbind and PAM

Since the PAM implementation of Linux distribution differs, we look at our test distributions in detail.

Winbind and PAM on Red Hat Desktop

Red Hat has implemented the pam_stack module. This means that configuration files in /etc/pam.d/ can use settings from other files, essentially stacking settings. For this purpose, Red Hat has implemented most of the settings in the /etc/pam.d/system-auth configuration file. This means that we only have to add winbind PAM modules to this file, and then, all applications using PAM are winbind aware.

Because the file /etc/pam.d/system-auth is generated by **authconfig**, care has to be taken in running this command after making changes manually.

Example 9-12 A sample system-auth file contains the lines in Example 9-12. Example of part of /etc/pam.d/system-auth file for winbind authentication

```
%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth      required      /lib/security/$ISA/pam_env.so
auth      sufficient   /lib/security/$ISA/pam_unix.so likeauth nullok
auth      sufficient   /lib/security/$ISA/pam_winbind.so use_first_pass
auth      sufficient   /lib/security/$ISA/pam_krb5.so use_first_pass
auth      sufficient   /lib/security/$ISA/pam_smb_auth.so use_first_pass nolocal
auth      required      /lib/security/$ISA/pam_deny.so

account  required      /lib/security/$ISA/pam_unix.so
account  sufficient   /lib/security/$ISA/pam_winbind.so
.....
```

Once pam_winbind.so has been incorporated into the system-auth file, all applications using the file through pam_stack.so are now winbind aware. This means that we can log on to the Linux client using a domain account, remembering to use:

```
<domainname><winbind-separator><accountname>
```

So an example in our test domain is AD6380+Administrator.

If it is problematic that winbind-enabled users are available for all applications using the pam_stack module, then pam_winbind.so calls could be placed only in the configuration files for those applications that need it.

Winbind and PAM on Novell Linux Desktop

The Novell Linux Desktop (NLD) does not use a pam_stack module to get settings from a central file. This means that for each application that needs to use winbind authentication, you must add the module to the configuration file in /etc/pam.d.

We take the configuration file for sshd as an example. After incorporating the pam_winbind.so module, the file looks like Example 9-13.

Example 9-13 Configuration file /etc/pam.d/sshd after incorporating winbind

```
#%PAM-1.0
auth    sufficient  pam_winbind.so
auth    required    pam_unix2.so use_first_pass  # set_secrpc
auth    required    pam_nologin.so
auth    required    pam_env.so
account required   pam_unix2.so
account sufficient pam_winbind.so
account required   pam_nologin.so
password required  pam_pwcheck.so
password required  pam_unix2.souse_first_pass use_authtok
session required   pam_unix2.sonone # trace or debug
session required   pam_limits.so
```

For every PAM-enabled application that you want to enable for domain users, you must add both the auth and account line for pam_winbind.so (before the nologin lines). Also, to prevent a double password prompt, you should add the parameter use_first_pass to any pam module needing a password in the configuration file apart from the pam_winbind.so module.

9.4.2 How to authenticate users using LDAP and PAM

To be able to use PAM together with LDAP to authenticate users, we need to do extra configuration for LDAP.

Because the objects in the SFU-extended Active Directory schema do not have the same name as the objects needed by PAM, we need some mapping entries. An example is shown in Example 9-14.

Example 9-14 Mapping entries for /etc/ldap.conf file

```
pam_login_attribute sAMAccountName
pam_filter objectclass=user
pam_member_attribute msSFU30PosixMember
pam_groupdn cn=unixusergroup,dc=AD6380,dc=LOCAL
pam_password ad
```

The highlighted line in Example 9-14 on page 209 enables us to narrow the users from AD that can actually connect to the Linux client. PAM only accepts the user if the user is a member of the group unixusergroup in AD.

Since the PAM implementation of Linux distribution differs, we look at our test distributions in detail.

LDAP and PAM on Red Hat Desktop

Red Hat has implemented the pam_stack module. This means that configuration files in /etc/pam.d/ can use settings from other files, essentially stacking settings. For this purpose, Red Hat has implemented most of the settings in the /etc/pam.d/system-auth configuration file. This means that we only have to add LDAP PAM modules to this file, and then, all applications using PAM are LDAP aware. The PAM module for LDAP is called pam_ldap.

Since the file /etc/pam.d/system-auth is generated by **authconfig**, you must take care in running this command after making changes manually.

Example 9-15 A sample system-auth file contains the lines in Example 9-15.Example of part of /etc/pam.d/system-auth file for LDAP authentication

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth      required      /lib/security/$ISA/pam_env.so
auth      sufficient   /lib/security/$ISA/pam_unix.so likeauth nullok
auth      sufficient   /lib/security/$ISA/pam_ldap.so use_first_pass
auth      sufficient   /lib/security/$ISA/pam_krb5.so use_first_pass
auth      required      /lib/security/$ISA/pam_deny.so

account   required      /lib/security/$ISA/pam_unix.so
account   required      /lib/security/$ISA/pam_ldap.so
.....
```

Once pam_ldap.so has been incorporated into the system-auth file, all applications using the file through pam_stack.so are now LDAP aware. This means that we can log on to the Linux client using a domain account.

If it is problematic that domain users are available for all applications using the pam_stack module, then you could place pam_ldap.so calls only in the configuration files for those applications that need it.

Important: Take care to put pam_ldap as required in the account section; otherwise, a user can get through that does not belong to the group that was entered in /etc/ldap.conf as required.

Example 9-16 shows the output of login with a domain user using SSH where the domain user is not part of the unixusergroup group (as defined in Example 9-14 on page 209), and the account field for pam_ldap is sufficient.

Example 9-16 Example of ssh succeeding even with incorrect group membership

```
[root@RHDT4a ~]# ssh testuser@localhost
testuser@localhost's password:
You must be a msSFU30PosixMember of cn=unixusergroup,dc=AD6380,dc=LOCAL to
login.
Last login: Fri Jan 13 10:09:34 2006 from rhdt4a
-bash-3.00$
```

Example 9-17 shows the output when the pam_ldap account field is required. The login fails but no helpful message is shown.

Example 9-17 Example of ssh failing because of incorrect group membership

```
[root@RHDT4a ~]# ssh testuser@localhost
testuser@localhost's password:
Permission denied, please try again.
```

LDAP and PAM on Novell Linux Desktop

The Novell Linux Desktop (NLD) does not use a pam_stack module to get settings from a central file. This means that for each application that needs to use LDAP authentication, the module has to be added to the configuration file in /etc/pam.d.

We take the configuration file for sshd as an example. After incorporating the pam_ldap.so module, the file looks like Example 9-18.

Example 9-18 Configuration file /etc/pam.d/sshd after incorporating LDAP

```
 #%PAM-1.0
auth    sufficient   pam_ldap.so
auth    required     pam_unix2.so use_first_pass  # set_secrpc
auth    required     pam_nologin.so
auth    required     pam_env.so
account required    pam_unix2.so
account required    pam_ldap.so
account required    pam_nologin.so
password required   pam_pwcheck.so
password required   pam_unix2.souse_first_pass use_authtok
session required    pam_unix2.sonone # trace or debug
session required    pam_limits.so
```

For every PAM-enabled application that you want to enable for domain users, you must add both the auth and account line for pam_ldap.so (before the nologin lines). Also to prevent a double password prompt, add the parameter use_first_pass to any pam module needing a password in the configuration file apart from the pam_ldap.so module.

Important: Take care to put pam_ldap as required in the account section; otherwise, a user can get through who does not belong to the group entered in /etc/ldap.conf as required.

9.4.3 PAM and home directories

Domain-enabled users do not exist on the client locally. Either the winbind configuration in /etc/samba/smb.conf or the entries in the SFU extension of the AD schema tell the system which shell to use and where the home directory of the user is located (this function is performed by the /etc/passwd file for local users). However, the home directory is not created by either winbind or LDAP.

This problem can be solved in a number of ways:

- ▶ Create all possible home directories (empty) on all clients.
- ▶ Create all home directories on a server file system that is mounted on all clients (either through SMB or NFS).
- ▶ Use the pam_mkhomedir.so module to create a home directory at first logon.

The first two options seem straightforward to implement but lead to management issues every time a user is added to the domain. The last option is seen as a best practice and consists of adding a line to the PAM configuration files that creates the home directory if it does not exist. This change of the PAM configuration file has to be done once when the Linux client “master” is created.

Red Hat Desktop

On Red Hat systems, the pam_mkhomedir module is added to /etc/pam.d/system-auth. This way the home directory is created when a user logs in through a login (for example, a terminal prompt), a secure shell session, or a graphical logon.

The line that you must add to the file is shown in Example 9-19.

Example 9-19 Example of part of the /etc/pam.d/system-auth file, including pam_mkhomedir

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth      required      /lib/security/$ISA/pam_env.so
```

```

auth      sufficient   /lib/security/$ISA/pam_unix.so likeauth nullok
auth      sufficient   /lib/security/$ISA/pam_winbind.so use_first_pass
auth      sufficient   /lib/security/$ISA/pam_krb5.so use_first_pass
auth      sufficient   /lib/security/$ISA/pam_smb_auth.so use_first_pass nolocal
auth      required     /lib/security/$ISA/pam_deny.so

account   required     /lib/security/$ISA/pam_unix.so
account   sufficient   /lib/security/$ISA/pam_winbind.so
.....
session  optional    /lib/security/$ISA/pam_mkhomedir skel=/etc/skel umask=0022

```

The skel option tells the module where to get the skeleton files to copy to the newly created home directory. The umask governs the creation and subsequent permission settings on the directory.

Novell Linux Desktop

Since NLD does not use system-auth and pam_stack.so, a pam_mkhomedir.so has to be added to every configuration file for applications that let users log on. Examples of these applications are SSH, telnet, gdm, xdm, and login. The first login of every domain user through any of these applications creates the home directory.

The line that is added to the file is of the form:

```
session optional pam_mkhomedir skel=/etc/skel umask=0022
```

An example for the ssh application is given in Example 9-20.

Example 9-20 Example of /etc/pam.d/sshd file including pam_mkhomedir.so

```

%%PAM-1.0
auth      sufficient   pam_ldap.so
auth      required     pam_unix2.so use_first_pass
auth      required     pam_nologin.so
auth      required     pam_env.so
account   required     pam_unix2.so
account   required     pam_ldap.so
account   required     pam_nologin.so
password  required     pam_pwcheck.so
password  required     pam_unix2.so use_first_pass use_authok
session  optional    pam_mkhomedir.so skel=/etc/skel umask=0022
session   required     pam_unix2.so none      # trace or debug
session   required     pam_limits.so

```

The skel option tells the module where to get the skeleton files to copy to the newly created home directory. The umask governs the creation and subsequent permission settings on the directory.

Example 9-21 show how a home directory is created when the domain user testuser logs on for the first time.

Example 9-21 Output of logging in with domain user for first time

```
n1d9:~ # ssh testuser@localhost
Password:
Creating directory '/home/testuser'.
Creating directory '/home/testuser/bin'.
Creating directory '/home/testuser/Documents'.
Creating directory '/home/testuser/public_html'.
Creating directory '/home/testuser/.xemacs'.
Creating directory '/home/testuser/.fonts'.

Last login: Fri Jan 13 16:09:47 2006 from localhost
testuser@n1d9:~> pwd
/home/testuser
```

9.5 How to mount a share on the Linux client

This section describes in general some of the ways to approach data on Windows shares.

One way is to permanently connect the share by mounting it somewhere in the filesystem. The shares can either be mounted using an smb mount or a CIFS mount.

If mounting is not an option, use the command smbclient. The smbclient works similarly to an ftp-client. Using the smbclient, it is possible to get and put files from and to a Windows share.

9.5.1 Mounting a share using smbfs

A share is mounted using the mount command in the following way:

```
mount -t smbfs //servername/sharename /mountpoint -o user=domainuser
```

The mount command calls smbmount to perform the mount of the share with the name sharename on server servername on the mountpoint specified. To connect to the server, the user domainuser is used and the command prompts for the password to use.

This is the most commonly used way to mount Windows share on non-Windows operating systems using Samba.

However, if the share resides on a server that performs SMB signing (most recent Windows servers do), this fails with an error message. This is because the Samba client does not handle SMB signing and, thus, cannot mount the share.

9.5.2 Mounting a share using CIFS

Common Internet File System (CIFS) is a platform-independent file sharing system built around the principles in the SMB-based filesystems of Microsoft. Recent 2.6 Linux kernels have CIFS built-in. Mounting a share as a CIFS filesystem enables the Linux client to go around the SMB signing problem mentioned before.

Also, a CIFS mount can be used to mount Microsoft DFS™ filesystems, which is something that `smbmount` also cannot handle.

The syntax for a mount using CIFS is highly similar to a mount using `smbfs`:

```
mount -t cifs //servername/sharename /mountpoint -o user=domainuser
```

Tip: Because of future compatibility, use CIFS where possible.

9.5.3 Use of `smbclient`

The `smbclient` command essentially turns a Windows share into an `ftp-server`. The command is invoked as follows:

```
smbclient //servername/sharename -U domainuser -W domainname
```

The command prompts for the password and connects to the share. Some of the commands that are available in the `smbclient` environment are:

- ▶ `get` - retrieves a file from the share to the local filesystem
- ▶ `put` - copies a file from the local filesystem to the share
- ▶ `cd` - changes working directory in the share
- ▶ `lcd` - changes working directory in the local filesystem
- ▶ `help` - shows all of the available commands
- ▶ `quit` - exits the `smbclient` environment

Example 9-22 shows how to connect to a share and how to get help on the commands in the `smbclient` environment.

Example 9-22 Example use of `smbclient` to connect to a Windows share

```
n1d9:~ # smbclient //w2k3ad.ad6380.local/share1 -U administrator
Password:
Domain=[AD6380] OS=[Windows Server 2003 3790] Server=[Windows Server 2003 5.2]
smb: \> help get
HELP get:
```

```
<remote name> [local name] get a file  
smb: \>
```

9.6 Automatically mounting home directories at logon

One of the great functionalities of Windows is the Single Sign On (SSO) function. Once you log on to a Windows operating system, it takes your password to try to mount shares that you have installed as remountable at next logon.

A similar functionality can be created on a Linux client using the pam_mount module. This PAM module is not completely mature yet. It is not included in all enterprise distributions. But with a little extra work, it can be made to function.

Important: Since September 2005, pam_mount seems to be getting more attention. New versions become available more rapidly. Although, these do not work on all distributions.

The SMB and CIFS filesystems do not allow the creation of symbolic links or sockets, because the underlying Windows share does not allow these constructs. This means that any application that needs to create a link or a socket cannot run from such a mounted filesystem.

Restriction: Take care when mounting SMB shares as users' home directories when using graphical logon. Some graphical desktop environments do not work in a SMB-mounted file system, most importantly, those that depend on symbolic links or sockets, because SMB file systems do not work with symbolic links and sockets.

Tip: Mount the user's domain share in a subdirectory of the home directory, thus, avoiding all issues with desktop environments.

The pam_mount module not only mounts SMB and CIFS file systems, but also NCP, loop-mounted encrypted file systems, and basically any file system handled by the `mount` command.

9.6.1 pam_mount on Red Hat Desktop

The module is added to the `/etc/pam.d/system-auth` file to enable automatic mounting for all login modes. The module consists of both an "auth" part, which

acquires the password through PAM, and a “session” part, which does the actual mounting. Because this is a session module, this enables an unmount of the file systems when the session closes.

Tip: Check if you need automatic mounting for all modes to log in. For example, a **su** from root to a user does not work because a password is not provided.

The lines to add look like this:

```
auth      required      pam_mount.so
```

And:

```
session  optional      pam_mount.so
```

The auth line should go before the pam_unix and pam_winbind lines. The session line should go in the session section. Remember to add a **use_first_auth** argument for the line not having one before adding pam_mount; otherwise, a second password prompt shows for that entry.

The pam_mount module has its own configuration file /etc/security/pam_mount.conf. This file contains settings such as where the **mount** and **umount** commands are found, debug settings, whether mount points should be created, and which file systems should be mounted for which users.

A minimum pam_mount.conf file for just mounting SMB shares looks like Example 9-23.

Example 9-23 Minimum pam_mount.conf

```
debug 1
mkmountpoint 1
options_require nosuid,nodev
lsof /usr/sbin/lsof
fsck /sbin/fsck
losetup /sbin/losetup
unlosetup /sbin/losetup -d
smbmount /usr/bin/smbmount
umount /usr/bin/smbumount
volume * smb smb3lab26 & /home/&/domainshare uid=&,dmask=0750 - -
```

This enables debugging on pam_mount. Mount points are created when they are non-existent and the share to be mounted is indicated in the last line.

The asterisk near the beginning of the last line in the example above indicates that this volume is mounted for every user. The ampersand character (&) in the

definition is expanded to the username of the user logging in. So, the volume line in Example 9-23 on page 217 tells pam_mount to mount a share named after the username from server smb3lab26 onto mount point /home/<username>/domainshare. The options indicate that the mount should be owned by the indicated uid and have permissions as indicated by dmask.

Important: In some cases, the pam_mount module fails to setuid root, which means that normal **mount** commands fail with an “only root can do that” message. This is, however, becoming much less frequent.

A possible problem when mounting shares from a domain is that the ampersand character (&) in the pam_mount.conf file expands to the entire username, including the domain name and the winbind separator. Thus, it is best when using pam_mount to go with the option winbind use default domain = yes in the smb.conf file to get domain usernames without the domain and winbind separator. Otherwise, extra scripting is needed to perform removal of these parts of the username before performing a mount.

Tip: When using pam_mount, use simple usernames for domain users. This means setting the option use default domain as described in 9.2.1, “Common implementation of winbind” on page 199.

9.6.2 pam_mount on Novell Linux Desktop

The situation using Novell Linux Desktop and pam_mount is slightly different from using Red Hat Desktop. The newest version of pam_mount does compile on Novell Linux Desktop. However, using pam_mount with sshd seems extremely difficult.

Pam_mount is added to all files in /etc/pam.d for all services that need automatic mounting. The module consists of both an “auth” part, which acquires the password through PAM; and a “session” part, which does the actual mounting. Because this is a session module, this enables an unmount of the file systems when the session closes.

Important: Since some versions of OpenSSH split the authentication off into a separate process, the password cannot be passed from the auth part of pam_mount to the session part of pam_mount. This means that pam_mount does not work in those cases.

The lines to add look like this:

```
auth      required      pam_mount.so
```

And:

```
session optional pam_mount.so
```

The auth line should go before the pam_unix and pam_winbind lines. The session line should go in the session section. Remember to append a **use_first_auth** argument to each auth line preceding the pam_mount line; otherwise, a second password prompt will be triggered by the pam_mount entry.

The pam_mount module has its own configuration file /etc/security/pam_mount.conf. This file contains settings such as where the **mount** and **umount** commands are found, debug settings, whether mount points should be created, and which file systems should be mounted for which users.

A minimum pam_mount.conf file for just mounting SMB shares looks like Example 9-23 on page 217.

9.7 How to use network printers in the domain

Being able to use existing printers is one of the most important preconditions for a new type of client in any environment. We do not want to keep separate printing environments for every type of client.

This means that the Linux client has to use the (network) printers available in the domain. We can achieve this in two ways:

- ▶ Print directly to the printer using its network interface.
- ▶ Print to the printer using the Windows print server and its SMB interface.

Both options are available using CUPS from the Linux client. We suggest using the second option, however, since all printing in the domain (and user's environment) is equal. This means there are just prints and not Windows prints and Linux prints.

You need to be running CUPS on the Linux client. Check this by:

```
/etc/init.d/cups status
```

This should return that CUPS is running and give its PID. If not, start it using:

```
/etc/init.d/cups start
```

Make sure that CUPS is started at the next reboot using:

```
chkconfig cups on
```

You also need to make sure that there is a link to smbspool in the CUPS back-end directory:

```
# ls -al /usr/lib/cups/backend/smb  
1rwxrwxrwx 1 root root 21 Mar 1 12:34 /usr/lib/cups/backend/smb ->  
../../..../bin/smbspool
```

You can verify smb support in CUPS using this command:

```
# lpinfo -v | grep smb  
network smb
```

You can create a new printer in two ways:

- ▶ Using the **lpadmin** command
- ▶ Using the CUPS Web interface

We describe both methods in detail.

Create a printer using **lpadmin** command

You can create a printer from the command line. This enables scripted creation of many printers. This means that creation of domain printers can be incorporated into a login profile, such that users do not have to create printers manually on their client.

To create the printer on the command line, we use the following command:

```
lpadmin -p printer1 -E -v smb://Administrator:*****@SMB3LAB26/printer1 -D  
"printer1 in AD6380 example domain"
```

You need to change the fields to the names in your own domain. The first printer1 is the local printer name and the second printer1 is the name of the printer share. The server SMB3LAB26 is the print server. If the printer share is available for everyone, you can leave out the username and password in the smb: URL. Since using a user-protected printer share exposes passwords in scripts and login profiles, the best practice is to make printer shares available for everyone.

You can also add the -m option to set the model. Or use the -l option to set a location.

Create a printer using CUPS Web interface

You could accomplish the same thing as above by using the CUPS Web interface. We do this following these steps:

1. In a browser, load the CUPS administrative interface (Figure 9-2) using this URL: <http://localhost:631/>.

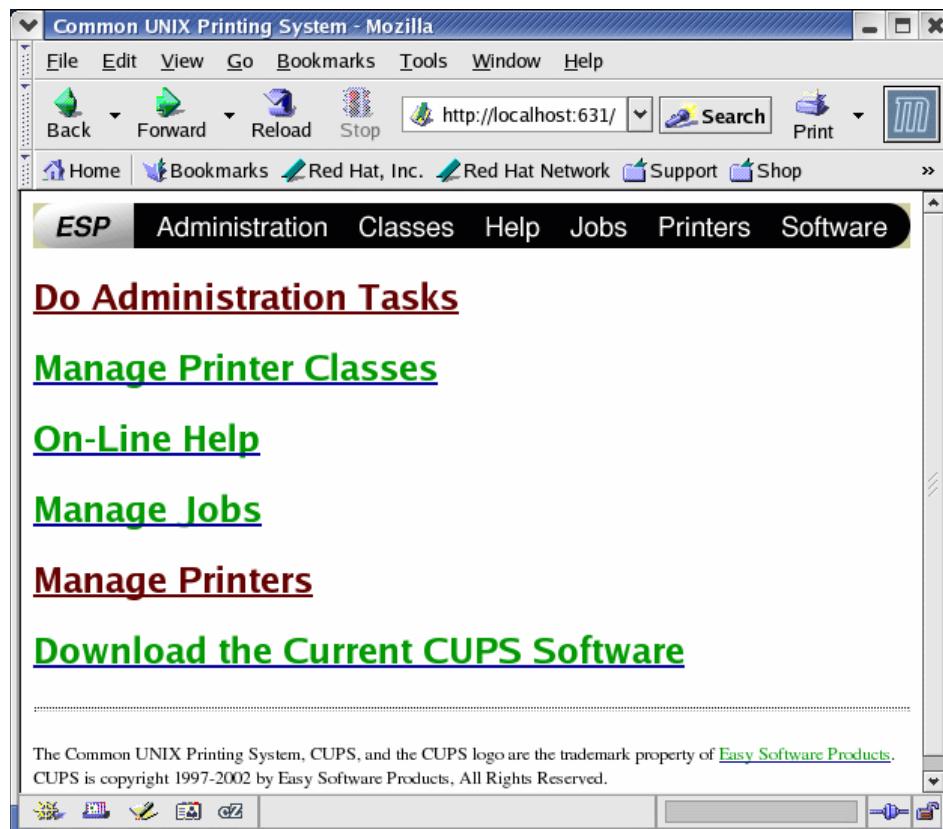


Figure 9-2 CUPS Web interface initial page

2. Then click **Manage Printers**. See Figure 9-3 on page 222.

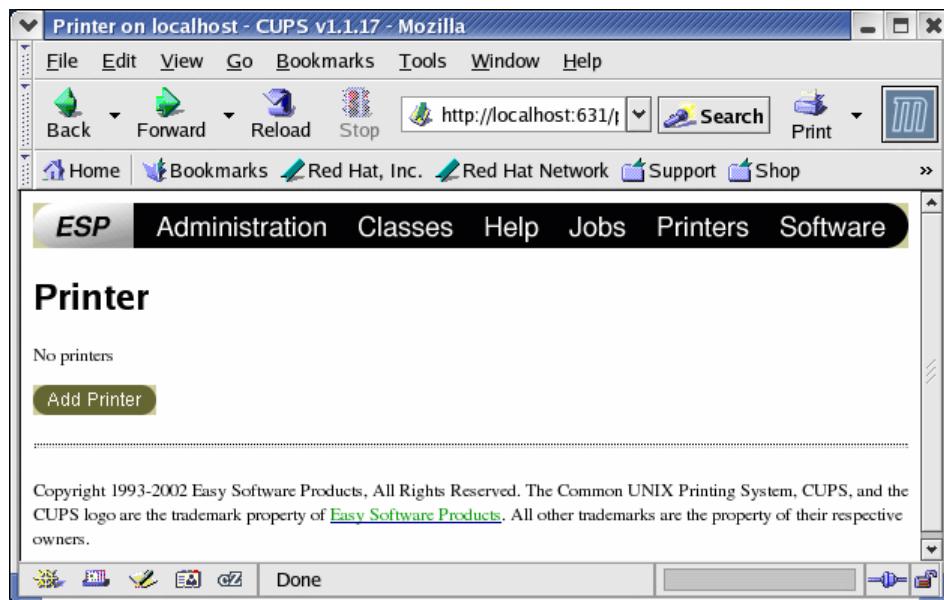


Figure 9-3 CUPS Web interface Manage Printers page

3. Click **Add Printer**. See Figure 9-4 on page 223.

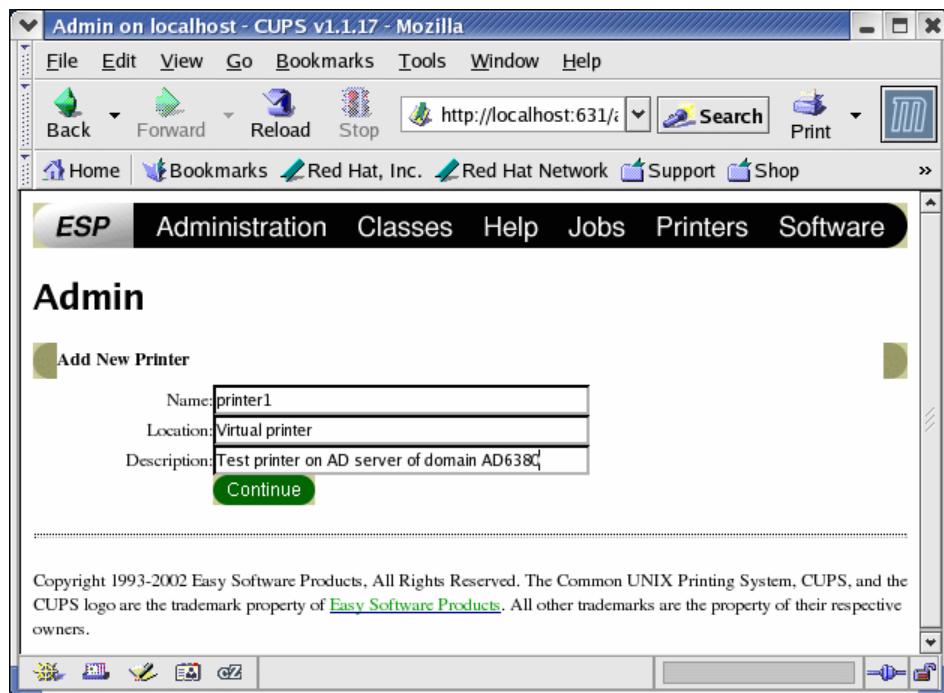


Figure 9-4 CUPS Web interface Add Printer page

4. You get a page where you enter the Name, Location, and Description of the printer. Then click **Continue**. See Figure 9-5 on page 224.

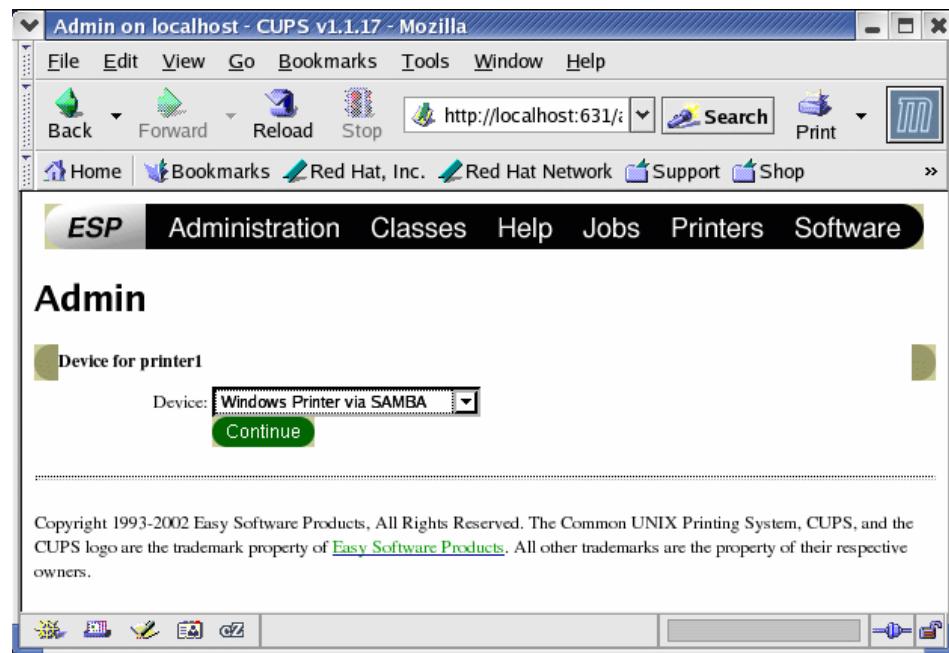


Figure 9-5 CUPS Web interface Device for the printer

5. On the next page, you choose the device for the printer. Choose **Windows Printer via SAMBA** and click **Continue**. See Figure 9-6 on page 225.

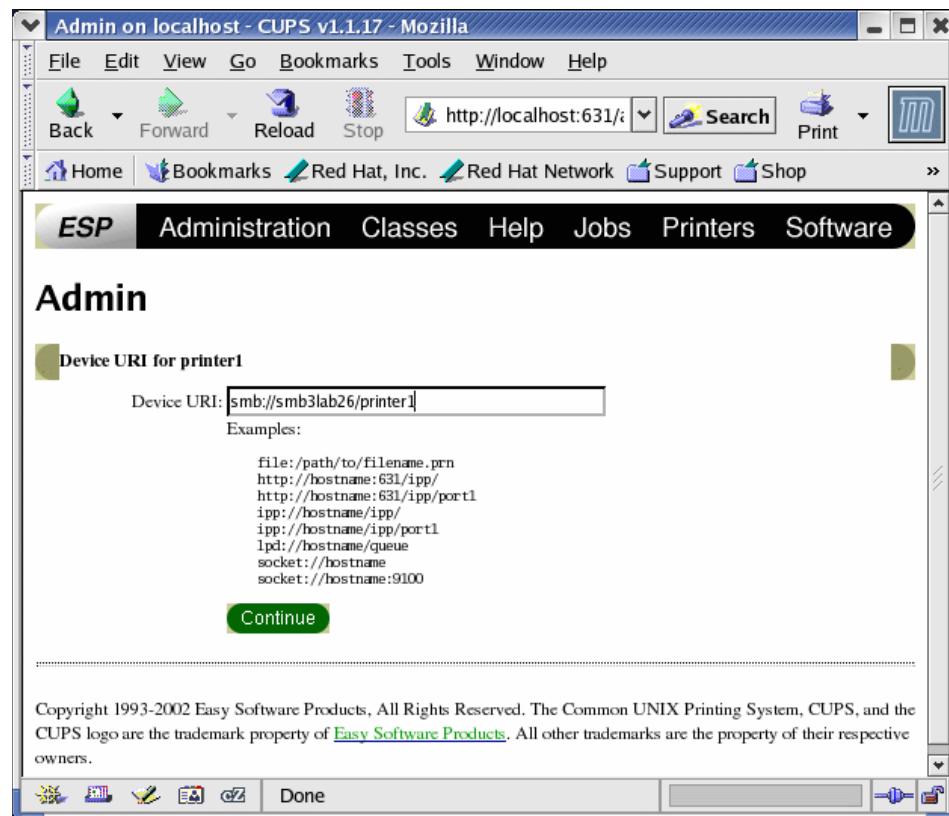


Figure 9-6 CUPS Web interface Device URI for printer

6. The next page asks for a device URI. Input:

smb://<domainname>:<password>@<printserver>/<printershare>

Or:

smb://<printserver>/<printershare>

Then click **Continue**. See Figure 9-7 on page 226.

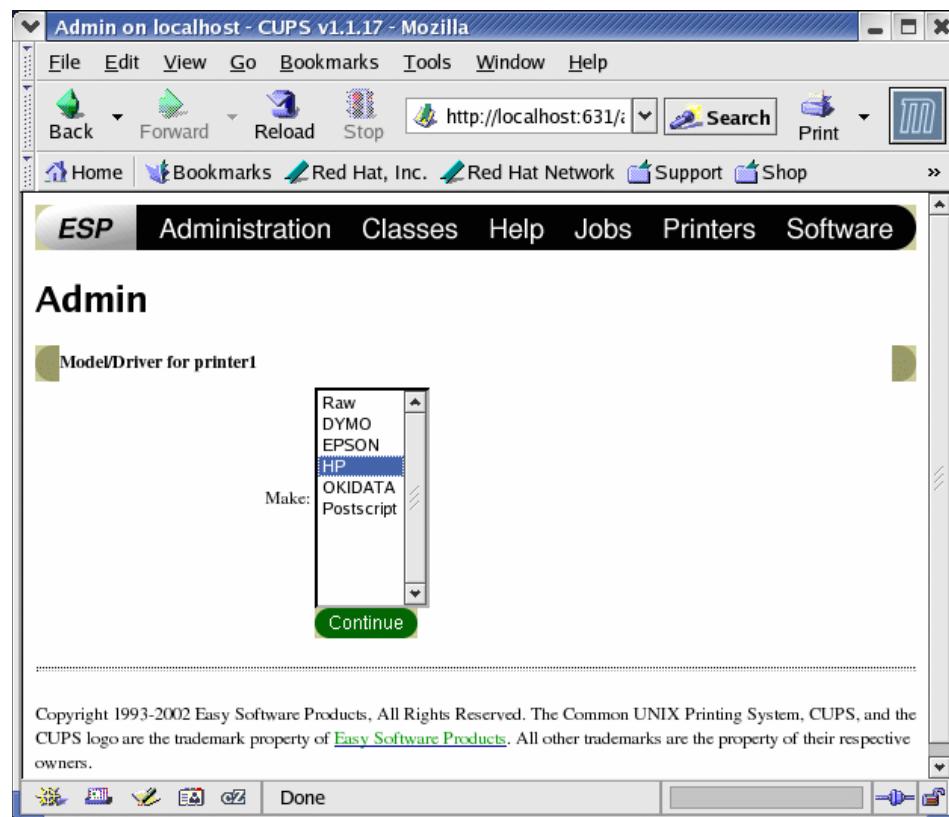


Figure 9-7 CUPS Web interface Model/Driver for printer

7. On the next page, choose a model and driver for the printer and click **Continue**. See Figure 9-8 on page 227.

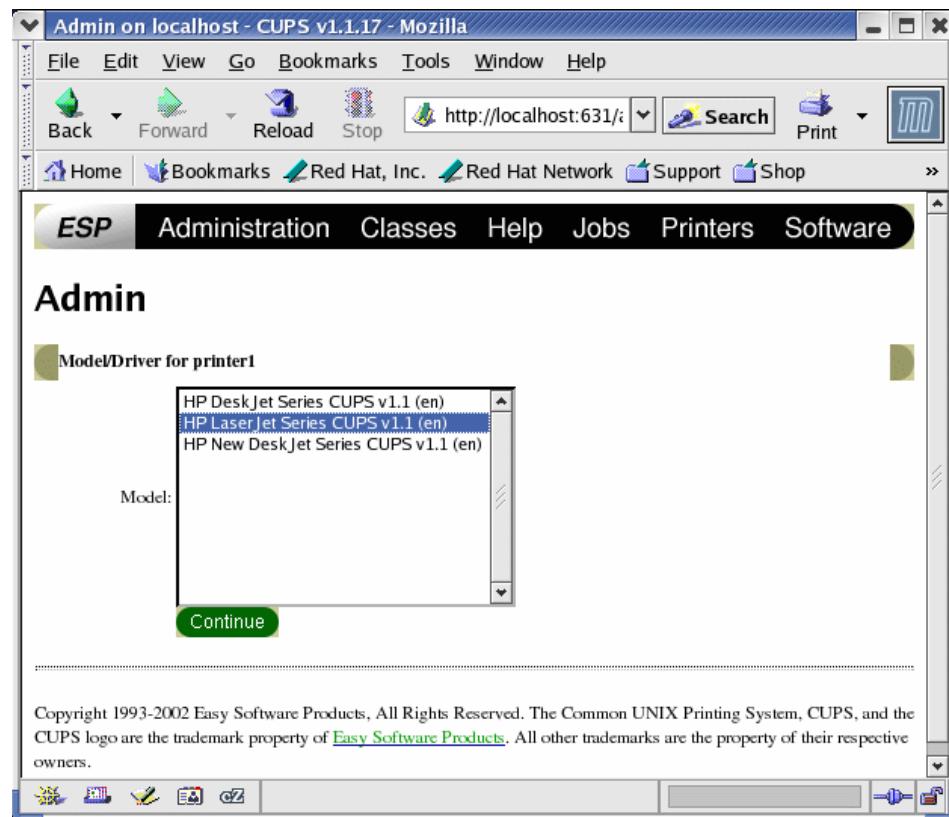


Figure 9-8 CUPS Web interface specific Model and Driver for printer

8. The next window gives a more detailed selection based on the model chosen on the previous screen. Choose the correct one and click **Continue**. See Figure 9-9 on page 228.

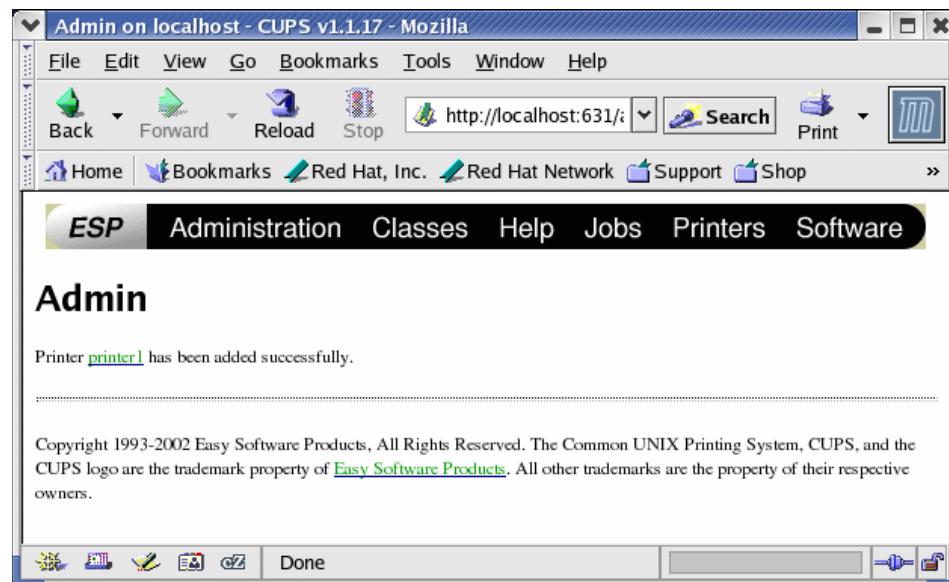


Figure 9-9 CUPS Web interface Printer added successfully message

9. You have added the printer to CUPS. See Figure 9-10 on page 229.

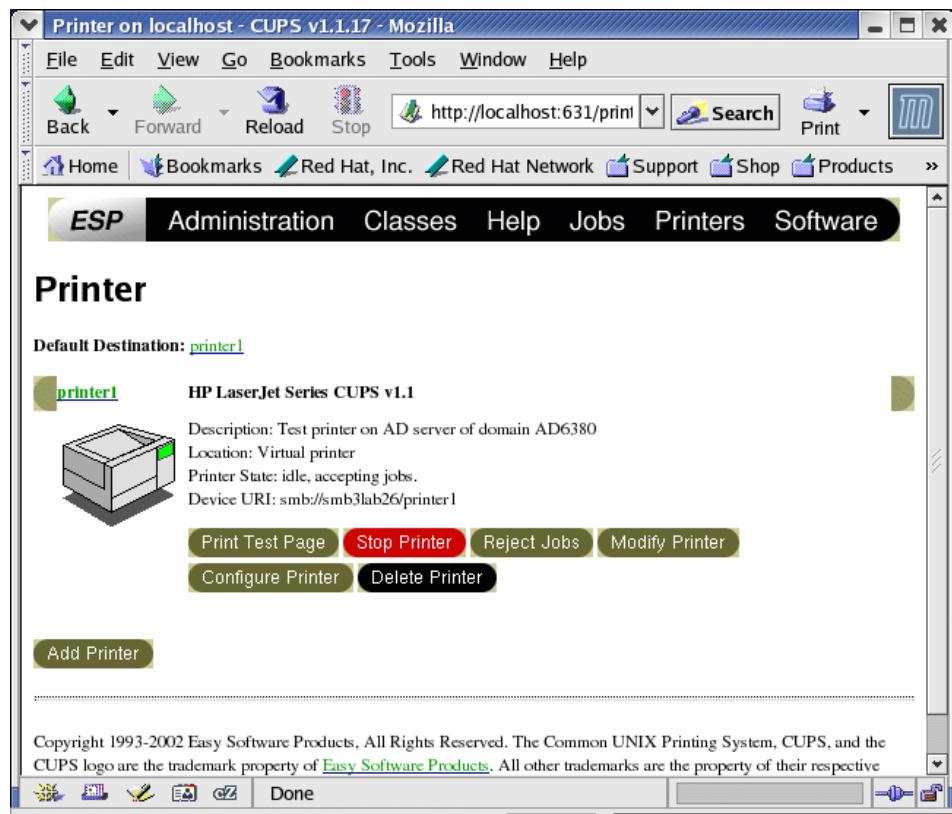


Figure 9-10 CUPS Web interface Manage Printers page with printer1 added

The screen captures in this section are created using a Red Hat Desktop client connected to an Active Directory domain AD6380 with domain controller smb3lab26.

To test the printer you added using `lpadmin` or the Web interface just print a file to it, or use the Print Test Page button on the Web interface.

Important: Using a username and password in the printer creation process will expose your password in the CUPS definition files. It might be best to use a special printer user or make printer shares available for everyone.



Part 4

Appendices

In part 4 of this book, we include:

Appendix A, “Linux glossary for Windows users” on page 233

Appendix B, “Using enterprise management tools” on page 255

Appendix G, “Application porting” on page 329

Appendix F, “Desktop automation and scripting” on page 321

Appendix E, “Client personalization” on page 313

Appendix C, “Automating desktop migration using Versora Progression Desktop”
on page 277

Appendix D, “Multi-station computing deep dive using Useful Desktop Multiplier”
on page 289



A

Linux glossary for Windows users

This appendix describes basic terms.

In this appendix, we discuss the following topics:

- ▶ “What does it all mean” on page 234
- ▶ “Common Linux Terms” on page 234

What does it all mean

For a Microsoft Windows user first delving into the world of Linux, there are many new terms to learn. This glossary explains briefly the meaning and significance of many of the terms, acronyms, and abbreviations common to Linux products. (Some of these terms are not specific to Linux, but still might be new to many Windows users.) This document should help lessen some of the confusion surrounding Linux; however, most common hardware, software, and communications terms are not included here, because they are easily found elsewhere.

Note: In the interest of keeping the definitions short, some might be oversimplified. They are not meant to be in-depth tutorials, but simply to provide a general explanation for a new user.

If you cannot find a word you are looking for here, there are many other sources of acronyms, abbreviations, and general computing terms (not all of them specific to Linux) from which to choose. Because some terms are likely to appear in one dictionary, but not another, and because some definitions can be clearer or more comprehensive in one source than in another, here is a selection to choose from, in alphabetical order:

- ▶ Free Online Dictionary of Computing
<http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?Free+On-line+Dictionary>
- ▶ IBM Terminology Web site
<http://www-306.ibm.com/software/globalization/terminology/>
- ▶ Lucent Technologies Glossary
<http://www.lucent.com/search/glossary>
- ▶ TechWeb TechEncyclopedia
<http://www.techweb.com/encyclopedia>
- ▶ Ugeek Technical Glossary
<http://www.geek.com/glossary/glossary.htm>
- ▶ Whatis.com
<http://www.whatis.com>

Common Linux Terms

Account Name	Same as Login ID, User ID, or Username. The name assigned to a user on a Linux system. Multiple users can
--------------	---

be set up on a system with unique account names, each with varying access (permission) levels. After Linux installation, account names are assigned by the Superuser, or root operator.

ACPI (Advanced Configuration and Power Interface)

The Advanced Configuration and Power Interface provides power management functionality on x86-based platforms.

ALSA (Advanced Linux Sound Architecture)

A framework for accessing sound cards and other audio devices under Linux. ALSA includes support for most popular audio chips and adapters. ALSA has replaced OSS on most distributions. (Also, see OSS.)

APM (Advanced Power Management™)

An industry standard for allowing the system processor and various components to enter power-saving modes, including suspend, sleep, and off. APM software is especially important for mobile devices, because it saves battery power.

Archive

A single large file containing multiple files, usually compressed to save storage space. Often created to facilitate transferring between computers. Popular archival formats include arj, tar, and zip.

Awk (Aho, Weinberger, and Kernighan)

A programming language useful for its pattern matching syntax, and often used for data retrieval and data transformation. A GNU version is called Gawk.

Background Process

A program that is running without user input. A number of background processes can be running on a multitasking operating system, such as Linux, while the user is interacting with the foreground process (for example, data entry). Some background processes—daemons, for example—never require user input. Others are merely in the background temporarily while the user is busy with the program presently running in the foreground.

Bash (Bourne Again SHell)

An enhanced version of the Bourne Shell. (Also, see Korn Shell.)

BDF Fonts

A variety of bitmapped fonts for the X Window System. (Also, see PostScript Fonts and TrueType Fonts.)

Bin	A directory containing executable programs, primarily binary files.
Binaries	Source code that has been compiled into executable programs. In the Linux world, some software is distributed as source code only; other packages include both source and binaries; still others are distributed only in binary format.
Boot Disk	A disk (floppy or CD) containing enough of an operating system (such as Linux) to boot up (start) the computer and run some essential programs from the command line. This might be necessary if the system was rendered non-bootable for some reason. A boot disk can be used to partition and format the hard drive, restore the Master Boot Record, or copy specific files, among other things.
Bootloader	An application that handles the initial startup of the computer. Bootloaders set up the initial environment, and then hand off the process to the selected operating system. (Also, see GRUB and LILO.)
Bot	Short for Robot. A program designed to search for information about the Internet with little human intervention.
Bourne Shell	A popular command line shell offering many advantages over the DOS command prompt. (Also, see Bash and Korn Shell.)
BSD (Berkeley Software Distribution) UNIX	UNIX distribution from University of California at Berkeley. (Also, see FreeBSD.)
Bzip2	A newer file compression program for Linux, providing smaller file sizes than Gzip. The file extension is usually .bz2.
CGI (Common Gateway Interface)	Used on Web servers to transmit data between scripts or applications and then return the data to the Web page or browser. CGI scripts are often created using the Perl language and can generate dynamic Web content (including e-commerce shopping baskets, discussion groups, and survey forms).
CHS (Cylinder/Head/Sector)	Disk information required by FDISK during partitioning.
Client	A machine that requests services (e-mail, for example) from a server.

Cluster	A network of workstations (PCs or other) running Linux. (Also, see Beowulf.)
Command Line Interface (CLI)	A full-screen or windowed text-mode session where the user executes programs by typing commands with or without parameters. The CLI displays output text from the operating system or program and provides a command prompt for user input.
Command Prompt	The DOS, Windows, and OS/2® term for the part of the command line interface where the user types commands. (Also, see Shell Prompt.)
Compile	To turn programming source code into an executable program.
Compiled Language	A language that requires a compiler program to turn programming source code into an executable machine-language binary program. After compiling once, the program can continue to be run from its binary form without compiling again. Compiled languages and programs tend to be faster than interpreted or p-code languages, but require an extra step of compiling before running the application. Examples of compiled languages are C and C++, COBOL, and FORTRAN.
Compiler	A program used to turn programming source code into an executable program.
Console Application	A command line program that does not require (or perhaps even offer) a graphical user interface to run.
Cron	A Linux daemon that executes specified tasks at a designated time or interval.
Daemon	A background process of the operating system that usually has root security level permission. A daemon usually lurks in the background until something triggers it into activity, such as a specific time or date, or time interval.
Desktop	The operating system user interface, which is designed to represent an office desk with objects on it. Rather than physical telephones, lamps, and in/out baskets, the operating system desktop uses program and data icons, windows, taskbars, and the like. There are many different desktop environments available for Linux, including KDE and GNOME, that can be installed by a user. (Also see

	the definitions for GUI, Window Manager, and X Window System in this section.)
Device Driver	A program that serves as an intermediary between the operating system and a device (such as ports, drives, monitors, or printers) defining to the operating system what capabilities the device has and translating the operating system commands into instructions the device understands.
Distribution	A packaging of the Linux kernel (core) with various user interfaces, utilities, drivers, and other software into a user deliverable. Often available as a free download or in a low-cost CD-ROM package. Popular distributions include Red Hat, SUSE, and Debian. Sometimes referred to as a “distro.”
Dpkg (Debian Package Manager)	A packaging and installation tool for Internet downloads, included with Debian Linux, but compatible with other distributions. It produces files with a .deb extension. Similar to RPM.
Emacs (Editing with MACroS)	A popular text editor, usually used as a console application. (Also, see Vi.)
Enlightenment	One of several user interfaces (window managers). For more about Enlightenment, see: http://www.enlightenment.org (Also, see GNOME, KDE, and X Window System.)
Environment variable	A variable used in scripts or console commands that refers to various environment settings. A common environment variable is \$HOME, which points to the current user's home directory.
Executable bit	The part of a file which determines if a file can be executed directly. Files without the executable bit are considered data files. Note that it is usually possible to execute files without the executable bit set if a helper application is used. (For instance, perl ./noexec.pl will still run the application noexec.pl, even if the executable bit is not set.)
File Extension	The trailing characters on a file name that are found after a period (.). The file extension usually describes what type of file it is. Unlike Windows, executables on Linux usually do not have a file extension.

File System	A set of programs that tells an operating system how to access and interpret the contents of a disk or tape drive, or other storage medium. Common file systems include: FAT and NTFS on Windows and ext3 and ReiserFS on Linux.
Filter	A program that reads data (from a file, program output, or command line entry) as input, processes it according to a set of predefined conditions (for example, sorted alphabetically) and outputs the processed data. Some filters include Awk, Grep, Sed, and Sort.
Finger	A Linux command that provides information about users that are logged on.
Foreground Process	In a multitasking operating system, such as Linux, the foreground process is the program that the user is interacting with at the present time (for example, data entry). Different programs can be in the foreground at different times, as the user jumps between them. In a tiered windowing environment, it is the topmost window.
FreeBSD (Free Berkeley Software Distribution)	Similar to Linux in that it includes many GNU programs and runs many of the same packages as Linux. However, some kernel functions are implemented differently. (Also, see BSD UNIX.)
Fsck (File System Check)	The command for scanning a disk for errors and fixing those errors if possible. Similar to the ScanDisk tool on Windows.
FTP (File Transfer Protocol)	A method of transferring files to and from other computers—often software repositories.
GCC (GNU C Compiler)	A high-quality C compiler governed by the GPL.
GIMP (GNU Image Manipulation Program), The	A popular image editor for Linux.
GNOME (GNU Network Object Model Environment)	One of several user interfaces (window managers) for Linux, built with GTK+. For more about GNOME, see: http://www.gnome.org

In verbal communication, the G is not silent, as in Guh-Nome. (Also, see Enlightenment, KDE, and X Window System.)

GNU (GNU is Not UNIX) Project

An effort of the Massachusetts Institute of Technology (MIT) Free Software Foundation (FSF) to develop and promote alternatives to proprietary UNIX implementations. GNU software is licensed under the GPL.

GNU/Linux

Same as Linux. So-called because many of the components included in a Linux distribution are GNU tools.

GPL (GNU General Public License)

A common usage and redistribution license. Any derivation of a work released under the GPL must also be released under the GPL or similar license. This includes application that interacts with a GPL library. For a copy of the GPL agreement, see:

<http://www.gnu.org/copyleft/gpl.html>

Grep (Global Regular Expression® and Print)

A tool that searches files for a string of text and outputs any line that contains the pattern.

GRUB (GRand Unified Bootloader)

A partition boot manager utility, capable of booting operating systems. GRUB provides a graphical menu to select which operating system to boot. GRUB has replaced LILO on most distributions. (Also, see LILO.)

GTK+ (GIMP ToolKit)

A powerful, fast open source graphics library for the X Window System on Linux, used by programmers to create buttons, menus, and other graphical objects. In verbal communication, it is called GTK. (Also, see GNOME, Motif, and Qt.)

GUI (Graphical User Interface)

The collection of icons, windows, and other on-screen graphical images that provides the user's interaction with the operating system. (Also, see Desktop and Window manager.)

Gzip (GNU zip)

The original file compression program for Linux. Recent versions produce files with a .gz extension. (A .z or .Z extension indicates an older version of Gzip.) Compression is used to compact files to save storage

	space and reduce transfer time. (When combined with Tar, the resulting file extensions might be .tgz, .tar.gz, or .tar.Z.)
Hard Link	Hard links are a cross between a shortcut to a file and a copy of a file. When hard linking, no data is copied, but a new entry to the original data is created. When the original file is deleted, the hard link pointing to the original data will remain. Hard links can only point to files on the same partition. (Also, see Symbolic Link.)
Home Directory	The directory the user is placed in after logging on, and where most (if not all) of a user's files are stored. Usually found in as a subdirectory of /home. Sometimes referred to as \$HOME, which is an environment variable that points to the current user's home directory. Also referred to as ~, which is a shell shortcut that points to the current user's home directory.
HTML (Hyper Text Markup Language)	The standard markup language for designing Web pages. Markup tags, or formatting commands, allow the Web page designer to specify highlighting, position graphics, create hyperlinks, and more.
HTTP (Hyper Text Transport Protocol)	The set of guidelines created for requesting and sending HTML-based Web pages.
Init	The first process to run immediately after the operating system loads. It starts the system in single-user mode or spawns a shell to read the startup files, and opens ports designated as login ports.
Interpreted Language	Unlike a compiled program, which is converted from source code to an executable one time, by a compiler, and then run from its binary form, an interpreted program is converted to binary on the fly each time it is run, by an interpreter program. Interpreted languages (and thus their programs) tend to be slower than compiled and p-code languages, and generally have limited authorization to low-level operating system functions or direct hardware access. On the other hand, they are often included along with operating systems, and are usually easier to program than compiled languages. Examples of interpreted languages are BASIC, Perl, Python, and REXX/Object REXX.

Java	An object-oriented programming language developed by Sun Microsystems™ to be operating system independent. Java is often used on Web servers. Java applications and applets are sometimes offered as downloads to run on users' systems. Java programming can produce applications, or smaller Java applets. While Java can be compiled to native code, it is typically compiled to bytecode, which is then interpreted. (Also, see JIT Compiler.)
Java Applets	Small Java programs that are embedded in a Web page and run within a browser, not as a stand-alone application. Applets cannot access some resources on the local computer, such as files and printers, and generally cannot communicate with other computers across a network.
JavaScript	A cross-platform World Wide Web scripting language, vaguely related to Java. It can be used as a server-side scripting language, as an embedded language in server-parsed HTML, and as an embedded language for browsers.
JDK™ (Java Development Kit)	A Java programming toolkit from Sun, IBM, or others, available for Linux and other operating systems.
JFS (Journaled/Journaling File System)	A file system that includes built-in recovery capabilities. Changes to the index are written to a log file before the changes take effect so that if the index is corrupted (by a power failure during the index write, for example), the index can be rebuilt from the log, including the changes.
JIT (Just-In-Time) Compiler	A compiler for interpreted languages that allows programs to be automatically compiled into native machine language on the fly, for faster performance of the program.
Journaling	Same as logging. Writing information to a journal (log) file as a method of tracking changes.
JVM™ (Java Virtual Machine)	Java run-time environment, required for the running of Java programs, which includes a Java interpreter. A different JVM is required for each unique operating system (such as Linux and Windows), but any JVM can run the same version of a Java program.

KDE (K Desktop Environment)	<p>One of several user interfaces (window managers) for Linux, built with Qt. For more on KDE, see:</p> <p>http://www.kde.org</p> <p>(Also, see Enlightenment, GNOME, and X Window System.)</p>
Kernel	The core of the operating system, upon which all other components rely. The kernel manages such tasks as low-level hardware interaction and the sharing of resources, including memory allocation, input/output, security, and user access.
Korn Shell	An enhanced version of the Bourne Shell, including extensive scripting support and command line editing. It supports many scripts written for the Bourne Shell. (Also, see Bash.)
LGPL (Lesser GPL)	A variation of the GPL that usually covers program libraries. Any application can interact with an LGPL library or application, but any derived library must be licensed under the LGPL or similar license. For a copy of the GPL agreement, see:
	http://www.gnu.org/copyleft/lesser.html
LILO (Linux LOader)	A partition boot manager utility, capable of booting operating systems. LILO is not constrained to booting just Linux. Most distributions has replaced LILO with GRUB. (Also, see GRUB.)
Link	An shortcut to a file or directory. (Also, see Symbolic link and Hard link.)
Linux	An open source UNIX-like operating system, originally begun by Linus Torvalds. The term “Linux” really refers to only the operating system kernel, or core. Several hundred people have contributed to the development of the Linux kernel. The rest of a Linux distribution consists of various utilities, device drivers, applications, a user interface, and other tools that generally can be compiled and run on other UNIX operating systems as well.
Log	To store application or system messages or errors. Also, a file that holds this information.
Lynx	A popular non-graphical (text-based) Web browser.
Macro	A set of instructions stored in an executable form. Macros can be application specific (such as a spreadsheet or

	word processing macro that performs specific steps within that program) or general-purpose (for example, a keyboard macro that types a user ID when Ctrl+U is pressed on the keyboard).
Man	The Linux command for reading online manual pages.
MBR (Master Boot Record)	The first physical sector on a bootable disk drive. The place where the system BIOS looks when the computer is first booted, to determine which partition is currently active (bootable), before reading that partition's first (boot) sector and booting from the partition.
Mesa	An implementation of the OpenGL (Open Graphics Library) API (Application Programming Interface). It provides standard guidelines and a toolset for writing 2D and 3D hardware-assisted graphics software.
MIME (Multipurpose Internet Mail Exchange)	A communications protocol that allows text e-mail messages to include non-textual (graphics, video, or audio, for example) data.
Motif	A graphics library for Linux, developed by the Open Software Foundation (OSF) and used by programmers to create buttons, menus, and other graphical objects for the X Window System. (Also, see GTK+ and Qt.)
Mount	Identify a disk drive to the file system. Hard drives, CDs, and floppies all need to be mounted before use. Removable media usually needs to be unmounted before ejecting the disk. Most modern distributions automatically mount and unmount removable media. It is also possible to mount disk images (such as ISO files), network devices, and even links to other parts of the file system.
Mount Point	The location on a file system where a hard drive partition, removable media, or other resource is mounted.
Multitasking	The ability of an operating system to run more than one program, or task, at a time. A cooperative multitasking OS, such as Windows 98, requires one application to voluntarily free up resources upon request so another application can use it. A preemptive multitasking OS, such as Linux, Windows NT-based systems, or OS/2, frees up resources when ordered to by the operating system, on a time-slice basis, or a priority basis, so that one application is unable to hog resources when they are

	needed by another program. (Also, see Multithreading and Time-sharing.)
Multithreading	The ability of an operating system to concurrently run programs that have been divided into subcomponents, or threads. Multithreading, when done correctly, offers better utilization of processors and other system resources. A word processor can make good use of multithreading, because it can spell check in the foreground while saving to disk and sending output to the system print spooler in the background. (Also, see Thread.)
NFS (Network File System)	A file system that allows the sharing of files across a network or the Internet.
Newbie	Someone new to the Internet, computers in general, or Linux specifically (for example, a “Linux newbie”).
Object-Oriented	A software development methodology that offers the programmer standard reusable software modules (components), rather than requiring the developer to write custom programming code each time. Using standard components reduces development time (because the writing and testing of those components have already been done by other programmers), and ensures a standard look and feel for programs using the same components.
OO	See Object-Oriented.
Open Source	A somewhat ambiguous term that refers to software that is released with its source code. The fact that the source code is provided does not necessarily mean that users can modify and redistribute the source code. The term is sometimes used interchangeably with “free software,” although they are not always the same. (Also, see Public Domain and Shareware.)
OSS (Open Sound System)	A device driver for accessing sound cards and other audio devices under Linux. It evolved from the Linux Sound Driver, and supports most popular audio chips and adapters. OSS has been replaced by ALSA on most distributions. (Also, see ALSA.)
OSS (Open Source Software)	See Open Source.

Owner	The user who has privileged access to a file; typically, the user who created the file.
P-code (Pseudo-code) Language	A type of Interpreted language. P-code languages are something of a hybrid, falling between compiled languages and interpreted languages in the way they execute. Like an interpreted language, P-code programming is converted to a binary form automatically when it is run, rather than having to be compiled. However, unlike a compiled language, the executable binary file is stored in pseudo-code, not machine language. In addition, unlike an Interpreted language, the program does not have to be converted to binary each time it is run. After it is converted to P-code the first time, the pseudo-code version is used for each additional execution. P-code languages (and thus their programs) tend to be slower than compiled languages and programs but faster than interpreted languages, and they generally have authorization to some low-level operating system functions but not direct hardware access. They are often included along with operating systems, and some p-code languages are easier to program than compiled languages. Examples of P-code languages are Java, Python, and REXX/Object REXX.
PAM (Pluggable Authentication Modules)	A replaceable user authentication module for system security, which allows programs to be written without knowing which authentication scheme will be used. This allows a module to be replaced later with a different module without requiring rewriting the software.
Panel	The name for the Linux equivalent of the Windows Taskbar.
Partition	A contiguous section of a disk drive that is treated by the operating system as a physical drive. Thus, one disk drive can have several mount points assigned to it.
PCF fonts	A variety of bitmapped fonts to be used with the X Window System.
PDF (Portable Document Format) files	Binary files created with Adobe Acrobat or other programs capable of producing output in this format. Used for producing operating system-independent documents, which can be viewed using Acrobat Reader or other

	programs, including Web browsers equipped with an Acrobat Reader plug-in.
Perl (Practical Extraction and Report Language)	A common programming language. It is often used on Linux Web servers for generating CGI scripts.
Permission	The authority to read and write files and directories, and execute programs. Varying permission levels can be assigned by the Superuser, or root operator, on a file-by-file, directory-by-directory basis or by account name (User ID). Permissions are often referred to in two forms of shorthand. When using shorthand, the first set regards the file owner, the second a file group, and the third is in reference to everyone. The first form of permission is to abbreviate the permission, with rwx meaning read, write, and execute respectively. The more common form uses a numerical value for permissions, where the number is the sum of permissions, with read equal to four, write equal to two, and execute equal to one. For instance, a file where the owner can read and write, the group can read, and everyone else who has no access would be abbreviated as 640.
PGP (Pretty Good Privacy)	A high-security, public-key data encryption program for Linux and other operating systems.
Port	The process of taking a program written for one operating system platform and modifying it to run on another OS with similar functionality. There is generally little or no attempt to customize the program to take advantage of the unique capabilities of the new operating system, as opposed to optimizing an application for a specific operating system.
Portable	A term referring to software that is designed to be used on more than one operating system with only minor modifications and recompilation.
POSIX (Portable Operating System Interface for uniX)	A set of programming interface standards governing how to write application source code so that the applications are portable between operating systems. POSIX is based on UNIX and is the basis for the X/Open specification of The Open Group.

PostScript	A page description language developed by Adobe Systems that tells a printer how to display text or graphics on a printed page.
PostScript Fonts	A wide variety of fonts that can be used with OS/2, Microsoft Windows, and the X Window System. Font files include those with .afm, .pfa, and .pfb extensions. Sometimes called Adobe Type 1 fonts, or ATM (Adobe Type Manager) fonts. PostScript fonts typically require a PostScript-compatible printer. (Also, see BDF Fonts and TrueType Fonts.)
Process	An executing program. (Also, see Multitasking and Multithreading.)
Public Domain	Software that is available to be used and modified by anyone, for any purpose, and might even be incorporated for distribution in commercial software. Public domain software is not copyrighted, and no rights are retained by the author. (Also, see Open Source and Shareware.)
Public Key Encryption	A method of data encryption that involves two separate keys: a public key and a private key. Data encrypted with the public key can be decrypted only with the private key and vice versa. Typically, the public key is published and can be used to encrypt data sent to the holder of the private key, and the private key is used to sign data.
Python	An object-oriented p-code programming language.
Qt	A powerful, fast open source graphics library for the X Window System on Linux, which is used by programmers to create buttons, menus, and other graphical objects. In verbal communication, Qt is pronounced the same as the word “cute.” (Also, see GTK+ and KDE.)
Queue	A list of tasks awaiting execution, as in “the print queue.”
RAID (Redundant Array of Independent/Inexpensive Disks/Devices)	A method of providing data redundancy, improved performance, and quick data recoverability from disk crashes, by spreading or duplicating data across multiple disk drives. Commonly used RAID types include RAID 0 (Data Striping), RAID 1 (Disk Mirroring), and RAID 5 (Striping with Distributed Parity). RAID configurations typically require identical drives (same capacity and even brand and model). RAID arrays appear to the operating system as a single device.

RC File	A script file containing the startup instructions for a program (an application or even the operating system). The file, to be executed automatically when the operating system is started, contains a list of instructions (commands or other scripts) to run.
RCS (Revision Control System)	A suite of programs that controls shared access to files in a group environment and tracks text file changes. Generally used for maintaining programming source code modules.
Rdev	A utility for obtaining information about a Linux system. It is used to query and set the image root device, the video mode, the swap device, and a RAM disk.
Root User	The user ID with authority to perform all system-level tasks. (Also called Superuser.)
Root Window	The underlying session in which the Linux desktop runs.
RPM (Red Hat Package Manager)	A packaging and installation tool for Internet downloads, included with some Linux distributions. It produces files with a .rpm extension. Similar to Dpkg.
Script	A set of commands stored in a file. Used for automated, repetitive execution. (Also, see RC File.)
SCP (Secure Copy)	A method for securely copying files between local and remote hosts. SCP uses SSH as a back-end. (Also, see SSH.)
Session	A complete interaction period between the user and the operating system, from login to logoff.
Shareware	A form of commercial software, where it is offered as “try before you buy.” If the customer continues to use the product after a short trial period, they are required to pay a specified, usually nominal, fee. (Also, see Open Source and Public Domain.)
Shell	A text-mode window containing a command line interface to the operating system.
Shell Prompt	The user input area of a shell. Whereas in a DOS shell the command prompt is designated by a Greater Than (>) symbol, in Linux it is usually a Percent (%) symbol, Dollar sign (\$) or other special character, depending on the shell used. (Also, see Command Prompt.)

Shell Script	A script designed to be run automatically when a shell is started.
Slash	The / symbol. Slash is used in file path names, instead of the backslash, \, used in the DOS, Windows, and OS/2 operating systems.
Source Code	Programming commands in their raw state as input by a programmer. Some programming languages allow the commands to be executed on the fly by a program interpreter. Other languages require the commands to be compiled into executable programs (binaries) before they can be used. In the Linux world, some software is distributed as source code only; other packages include both source and binaries; still others are distributed in binary format only.
Spool (Simultaneous Peripheral Operation On-Line)	To send data to a program that queues up the information for later use (for example, the print spooler).
SQL (Structured Query Language)	The language used for manipulating records and fields (rows and columns) in a relational database. Sometimes pronounced “sequel.”
SSH (Secure Shell)	A secure protocol for logging in to a remote machine. Many protocols can be “tunneled” through SSH, such that connections travel through a SSH connection between two machines instead of traveling unencrypted across the network.
String	A sequence of characters, as in a “search string.”
Superuser	Usually synonymous with root user.
Swap	To temporarily move data (programs or data files) from random access memory to disk storage (swap out), or back (swap in), to allow more programs and data to be processed than there is physical memory to hold it. Also called Virtual Memory.
Swap Space	Where swapped data is temporarily stored on disk. Linux uses a dedicated disk partition for swap space, rather than a specific swap file.
Symbolic link	An shortcut to a file or directory. Sometimes called a symlink. If the original file is deleted, the symbolic link will no longer work. (Also, see Hard link.)
Sync	To force all pending input/output to the disk drive.

Syslog	The Linux System Logger, where all system messages or errors are stored.
Tag	A command in a markup language, such as HTML, to display information in a certain way, such as bold, centered, or using a certain font.
Tar (Tape ARchive)	A file packaging tool included with Linux for the purpose of assembling a collection of files into one combined file for easier archiving. It was originally designed for tape backup, but today can be used with other storage media. When run by itself, it produces files with a .tar extension. When combined with Gzip, for data compression, the resulting file extensions can be .tgz, .tar.gz, or .tar.Z.
Tarball	A file created by the Tar utility, containing one or more other archived and, optionally, compressed files.
TeX	A popular macro-based text formatter. The basis for other such formatters, including LaTeX and TeX.
Text Editor	A program for editing text files. Similar to a word processor, but without most of the formatting functions (such as margins, italics, and fonts). Often used for writing or editing scripts, programs, and plain text files.
Text Formatter	A program that prepares a text document for printing, allowing the user to perform many layout functions, such as margins, headers, footers, indentation, pagination, and justification.
TFTP (Trivial File Transfer Protocol)	A simplified version of FTP without authentication or many other basic features of FTP.
Thread	A small piece of programming that acts as an independent subset of a larger program. A multithreaded program can run much faster than a monolithic, or single-threaded, program because several different tasks can be performed concurrently, rather than serially (sequentially). Also, threads within a single application can share resources and pass data back and forth between themselves.
Time-sharing	A method of allowing multiple users to share a processor by allocating each user a portion of the processor resources on a timed basis and rotating each user's processes within those time segments. (Also, see Multitasking.)

Touch	A command that changes the date and time stamp of a file without affecting the contents. If passed a file name that does not exist, touch will create an empty file.
TrueType Fonts	A wide variety of fonts designed to be printer-independent, unlike PostScript fonts. (Also, see BDF Fonts and PostScript Fonts.)
Tux	The name of the fictional Linux penguin mascot.
Unmount	The command for unmounting a hard drive partition, removable media, or other resource. (Also, see Mount.)
UNIX	UNIX began as a proprietary operating system developed by Bell Laboratories in the 1960s. It eventually spawned a number of mutually incompatible commercial versions from such companies as Apple (Mac OS X), Digital (Digital UNIX), Hewlett-Packard (HPUX), IBM (AIX®), NeXT (NeXTSTEP), and others.
UUCP	A set of programs and protocols that have become the basis for a worldwide network of UNIX computers. Named after the UNIX to UNIX Copy Program.
Vi	A popular text editor, usually used as a console application. Also used when discussing newer version, Vim. (Also, see Emacs.)
Virtual Desktop	A method for expanding the user's workspace beyond the boundaries of the computer screen. The desktop can be scrollable left and right, up and down, as though a larger desktop were positioned behind the glass screen and moved around to reveal icons, windows, and other objects that were "off-stage," or out of view. Alternatively, as with the KDE desktop, multiple buttons can be available, each of which displays an area of desktop equal to the size of the glass screen and which can each contain different objects.
Virtual Machine	Virtual Machines (VMs) are features of central processor chips that isolate an area of memory from the rest of the system. Because operating systems and applications run in a "protected mode" environment, if a program freezes in one Virtual Machine, it will not affect the operation of the programs and operating systems running outside of that Virtual Machine.
Virtual Memory	The process of using a portion of disk space as a temporary storage area for memory. Synonymous with Swap.

Widget	A graphical user interface programming object (such as a button, scroll bar, or check box) for the X Window System. (Also, see X Window System.)
Window Manager	The graphical user interface (GUI) that runs on top of the X Window System to provide the user with windows, icons, taskbars, and other desktop objects. (Also, see Desktop.)
Working Directory	Another name for the current directory, or the directory in which the user is currently working.
Workspace	Another name for the Root Window, or Desktop.
Wrapper	A program used to start another program.
X Window System	A graphical windowing environment for UNIX. The underlying program required by many user interfaces. (Also, see Desktop and Window Manager.)
X11	Version 11 of the X Window System.
XDM (X Display Manager)	User-friendly login front end for the X Window System.
XML (eXtensible Markup Language)	A powerful new markup language for designing data; similar to HTML, but allows programmers to define their own markup tags, or formatting commands.
Zip	A popular form of file compression and archiving available on many operating system platforms, including Windows and Linux. Popular tools include PKZip/PKUnzip and Zip/Unzip. Zipped files will have a .zip extension.
>	The redirection symbol; it is often used to send the output from a command to a text file. For example, the following command sends the current directory list to a file called output.txt, overwriting what had previously been in that file: <pre>ls -a > output.txt</pre> (Also, see Append Symbol and Piping Symbol.)
>>	The append symbol; it is often used to send the output from a command to a text file, appending the data to the end of the file, rather than replacing the existing content. For example, the following command sends the current directory list to a file called output.txt, and adds it to the end of the file. <pre>ls -a >> output.txt</pre>

Repeating the command will continue to add new data to the end of the file. (Also, see > and !.)

|

The piping symbol (the Shift+Backslash character, above the Enter key on a typical 101-key keyboard); it is often used to feed the output from one command or program to another. For example, to search for a previously entered command with the word “mcopy” in it, use the following command:

```
history | grep mcopy
```

(Also, see Append Symbol and Redirection Symbol.)

/

See Slash.

~

See Home Directory.



Using enterprise management tools

This appendix describes the system management tools available from two major Linux enterprise distribution vendors, and some from other sources as well. For each of these tools, this appendix describes their basic functionality, in what environments the tool might be useful, and pointers to more information. These tools provide extremely useful functions for facilitating a large migration.

This appendix discusses the following topics:

- ▶ “Why use enterprise management tools” on page 256
- ▶ “Red Hat Satellite server and Red Hat Network (RHN)” on page 257
- ▶ “Novell ZENworks Linux Management” on page 264
- ▶ “Webmin” on page 269
- ▶ “Other important tools” on page 272

Why use enterprise management tools

Just like you need to automate certain tasks when using other operating systems, it is also advantageous to do this when using Linux. Red Hat and Novell SUSE both provide options for automating client management tasks. These tools and a few others are introduced and discussed in this appendix.

You might or might not have used these tools, but you might want to consider their use when you migrate to Linux. These system management services enable a client or set of clients to be configured and monitored remotely using several optionally available management frameworks or applications. In the case of client deployments based on Red Hat Enterprise Linux and the RHN management framework, or SUSE Linux Enterprise Desktop and the Novell ZENworks framework, the management tools provide a set of key value-added options that can easily justify the subscription-based licensing fees.

The availability of enterprise management tools and how you plan on using them can strongly affect the cost justification models in a migration scenario. This is discussed in 2.1.2, “Costs related to Linux client” on page 32.

Internet standard technologies

There are published Internet standards that support community development of system management tooling. There are currently many active open source management tooling development projects that are based those standards. In the case of Web-Based Enterprise Management (WBEM), the goal is to provide a platform for unifying the management of distributed computing environments.

Web-Based Enterprise Management (WBEM)

Web-Based Enterprise Management (WBEM) and the Common Information Model (CIM) are the object-based architecture standards and the data model promoted by the industry consortia Distributed Management Task Force (DMTF). The WBEM services described here are implementations of these standards. For more information about these standards, refer to the DMTF Web site:

<http://www.dmtf.org>

WBEM services are made available for use by a system management application through a CIM object manager or CIM server. Providers are written that gather and manage system resources. These providers register their classes with the CIM server to enable these objects to be used by a management application or even other providers. There are increasing numbers

of applications adopting this new management standard. For more information about the components in the WBEM architecture, refer to the DMTF Web site:

<http://www.dmtf.org/standards/wbem>

Simple Network Management Protocol (SNMP)

Another commonly used remote system management service is the Simple Network Management Protocol (SNMP). SNMP is an Internet Engineering Task Force (IETF) standard and is perhaps the most widely implemented standard for system management. The data model used by an SNMP service is called a Management Information Base (MIB). Different sections of this MIB are supported by different devices and servers depending on the services available on the system. For more information about the SNMP service, refer to the IETF Web site:

<http://www.ietf.org>

SNMP services are implemented through a background server or daemon. This daemon listens for requests and if the object is supported in the set of MIB extensions maintained by the server, the action is performed and the response returned. The SNMP service also has the ability to send and receive system events called *traps*. And, with the deployment and wide spread adoption of secure SNMPv3, it is now even safer to use.

Red Hat Satellite server and Red Hat Network (RHN)

The Red Hat Network (RHN) is a systems management framework for your Linux infrastructure. Through a Web-based interface, it is possible to automate system updates, management, provisioning, and, monitoring. It provides a set of services through what is known as *modules*. See the following site for additional overview information about RHN:

<http://www.redhat.com/rhn>

Each of the major functions in RHN are defined in more detail here:

<https://rhn.redhat.com/rhn/help/reference/index.jsp>

Architectural and functional overview

There are three basic architectures for using RHN. These are the Hosted server model, the Proxy server model, and the Satellite server model. The architecture of the Red Hat Network is one where there is a central server from which all systems are managed. With the Hosted model, the RHN server is located at and hosted by Red Hat, with access through the Internet.

For supporting small numbers of systems and in scenarios with limited security considerations, the hosted model can provide an appropriate level of service and flexibility. When the number of systems in an enterprise grows or when security constraints prevent all systems from reaching the Internet, you should consider the proxy or satellite models. Of the three models, satellite provides the most flexibility and features.

Red Hat Network architectures:

- ▶ **Hosted model** - All systems connect to the RHN server at Red Hat through the organization's Internet firewall. All management information and services are exchanged directly with the RHN servers over the Internet.
- ▶ **Proxy model** - The managed systems connect to the locally administrated proxy server. The Proxy server aggregates data, caches content, and performs several tasks locally. It communicates with the RHN server on the Internet and can be added to a Satellite model to help scale content distribution. In this model, all management information for the RHN is stored on the RHN servers (or the locally hosted Satellite servers).
- ▶ **Satellite model** - Systems connect to the satellite server running locally. The satellite server functions as a locally connected RHN update server. It can receive updates for distribution to your network using Internet connection to RHN servers. It can also be loaded with updates manually, using physical media, thus eliminating the need for an external Internet connection for the satellite server (disconnected mode).

All of the models provide a core set of modules that can be grouped within the following functional areas:

- ▶ Update
- ▶ Management
- ▶ Provisioning
- ▶ Monitoring

Additional modules that add more functionality for Provisioning and Monitoring are available in the Proxy and Satellite models. For more details, see the Red Hat Network module and architecture comparison tables here:

<http://www.redhat.com/rhn/compare/>

In Figure B-1 on page 259, we show an example topology that demonstrates a tiered approach combining proxy and satellite servers. This approach might be best suited for larger, distributed enterprises.

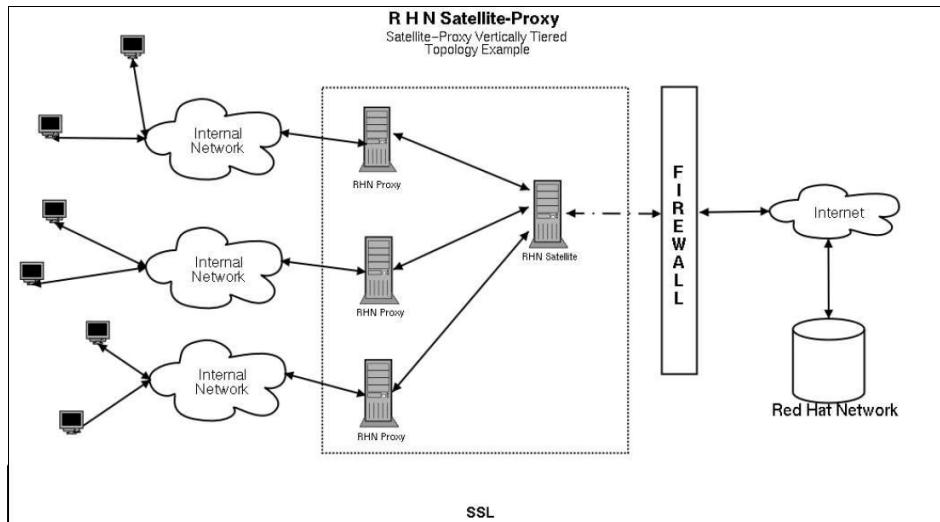


Figure B-1 ¹Red Hat Network - Example of combined Satellite-Proxy model

Satellite server

The technical architecture of the satellite server consists of a server application on top of a database to contain all of the data about systems, groups, entitlements, channels, packages, and errata. This application has a management Web interface. The communication with the clients is performed through the **up2date** application and several daemons running on the client.

Note: In Red Hat Enterprise Linux Version 5, **up2date** will be replaced by **yum**.

RHN Terminology

In this section, we describe terminology used in discussing the RHN/satellite server. These concepts include:

- ▶ System
- ▶ System group
- ▶ System set
- ▶ System Entitlement
- ▶ Channel
- ▶ Channel Entitlement
- ▶ Errata
- ▶ Action
- ▶ Activation key

¹ Copyright 2004 Red Hat, Inc. All rights reserved. Reprinted by permission.

We describe each of these concepts in some detail in the satellite server context. Some of these same terms exist as a concept in ZENworks as well, but might have a slightly different meaning.

System

A system can be any Red Hat Linux installed machine with **up2date** configured for connection to a server. A system is connected to a satellite server using a registration process that associates the system with a user ID on the satellite server.

System group

A system group is an entity in the satellite server that can contain systems. When created, the container is empty. Systems can be added to the system group. The system group is used in the satellite server to perform actions for multiple systems (the ones in the group) simultaneously. You can also have individual systems that belong to multiple system groups, that for instance might overlap by function and geography in your enterprise.

System set

Where the system group is more or less fixed, the system set can be used to select systems to perform an action on. These systems can be picked individually, or on a group level, or based on some property of the system. You can construct a system set “on the fly” in order to group systems together for applying a particular action.

Entitlements

Entitlements allow subscription access to the service level offerings in RHN. The quantity and types of available subscriptions are set by which RHN licenses have been purchased. You can have system level or software channel level entitlements.

Modules

The concept of modules allows for the choice of levels of service entitlement that the satellite server can perform. In the current Red Hat Network and satellite server, there are four modules:

- ▶ Update - The entry-level module, complimentary with Red Hat Network subscription, most appropriate for single or a few systems.
- ▶ Management - Includes Update module features plus additional features that enable more efficient system grouping, management, and channel definition to support larger numbers of systems.

- ▶ Provisioning - Includes Management module features plus additional features that provide for full Linux infrastructure lifecycle management.
- ▶ Monitoring - Requires the management module and a satellite server. Enables tracking and monitoring of all the Linux systems performance.

Note: For the most current information about RHN modules, refer to the Red Hat Web site:

<http://www.redhat.com/rhn>

With each higher level entitlement, more functions of the satellite server/RHN become available. You can change the module entitlement levels assigned to a particular system or group of systems whenever necessary.

Channel

A *channel* is a collection of updates. There are two types of channels: Base channels and child channels.

A *base channel* consists of a list of packages based on a specific architecture and Red Hat Enterprise Linux release. For example, all of the packages in Red Hat Enterprise Linux 4 for the x86 architecture are a base channel.

A *child channel* is a channel associated with a base channel, but it contains extra packages. This way, it is possible for an organization to use a base channel to create a child channel that contains the base packages and extra packages specifically needed by the organization.

Action

Actions can be performed on clients from the satellite server. Since the rhnd daemon is running as root, even a reboot can be scheduled from the satellite server. Some actions that can be scheduled are:

- ▶ Boot
- ▶ Package update
- ▶ Hardware properties update

Errata

Errata is the name for the collection of all updates and patches. Red Hat distinguishes between three types of errata:

- ▶ Security updates
- ▶ Bugfix updates
- ▶ Enhancement updates

These three types of errata each have their own priority for an application. Security updates should be applied immediately, Bugfix updates if necessary, and Enhancement updates when needed.

Activation key

An activation key is used to register the system, entitle the system to a service level, and subscribe the system to specific channels and system groups through the command line utility `rhnreg_ks`. These keys can only be created for Management and Provisioning entitlements. The keys are generated in the satellite server. For a single system, you can combine multiple activation keys (key stacking). Key stacking allows for a lower total number of keys required for multiple systems with differing subscription level requirements.

Sample update scenario

Consider a large multinational company, running Red Hat Desktop across the enterprise as their primary user client platform. Their IT department developers run the latest versions on test bed client platforms. As their enterprise applications are ported to the latest releases and pass Quality Assurance (QA) testing, they can then provide updates to their production client groups using RHN channels.

A simplified Red Hat Network topology focusing on the desktop channels is shown in Figure B-2 on page 263. A satellite server provides three base channels. The Red Hat Desktop channel serves as a base for set of child channels that are used for management development and testing (numbered 1-2-3® in the figure). The RHD base channel provides RHD Update 3 (the base channel would be labeled “RHD_U3”). So each of the child channels could be labeled as follows:

1. RHD_U3dev
2. RHD_U3test
3. RHD_U3prod

The IT department wants to deploy RHD Update 4 to their production desktop channel in the same way that they managed the Update 3 deployment. To do this, they follow the same pattern used to roll out Update 3 to the enterprise. They first create a new “RHD_U4dev” base channel on the satellite server. They can do this by cloning the Red Hat Desktop channel as provided by Red Hat, or they can create a new blank channel and push the appropriate packages to the channel from the installation media. This new RHD_U4dev channel is then used by their developers for testing and updating any applications.

Once the developers have completed their work on RHD_U4dev, the same channel is cloned as RHD_U4test, and the IT quality assurance team takes over. They test the packages and file bugs, and they might also request that the development team apply fixes to any packages included in the channel.

When the QA team is able to sign off on production readiness for the RHD_U4test channel, it can be cloned into a new channel, RHD_U4prod. This new production channel is then used to manage deployment of the U4 update to the enterprise.

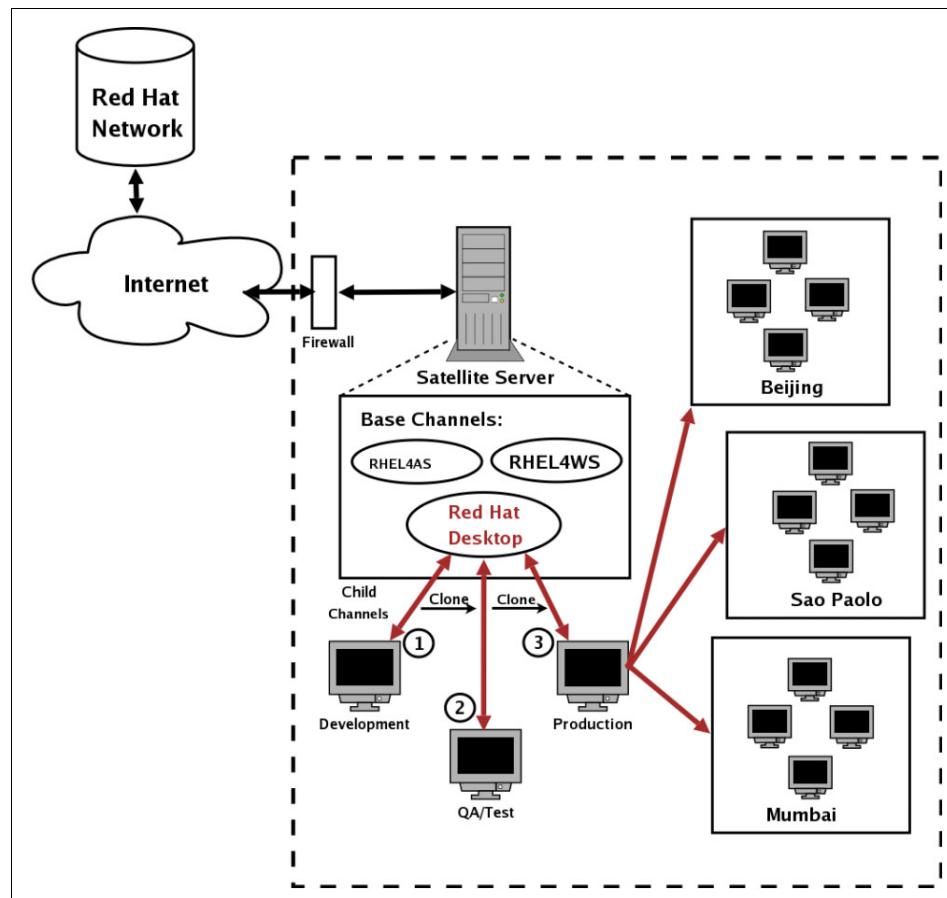


Figure B-2 Red Hat Network update topology

Novell ZENworks Linux Management

This section describes Novell ZENworks, now called ZENworks Suite. ZENworks Suite is Novell's application platform for deploying, managing, and maintaining systems in the enterprise. It can be especially useful for enterprise customers running a large number of Linux client installations within their organization.

ZENworks is the preferred tool for Novell/SUSE distribution, but can manage any client desktop, either Linux-based or Windows-based. The ZENworks Linux Management tools are the focus in this section.

Architecture and Functionality

ZENworks Suite contains Web-based tools to enable the administrator to remotely manage systems in an heterogeneous environment. There are currently modules in the suite for managing: assets, data, desktops, handhelds, Linux, desktop personality settings migration, servers, software, and patches. Their Linux Management solution uses policies and automation to manage and maintain Linux systems.

The ZENworks suite focuses on full lifecycle systems management. For a complete overview of the ZENworks Suite functionality, refer to the Novell Web site at:

<http://www.novell.com/products/zenworks/>

See Figure B-3.

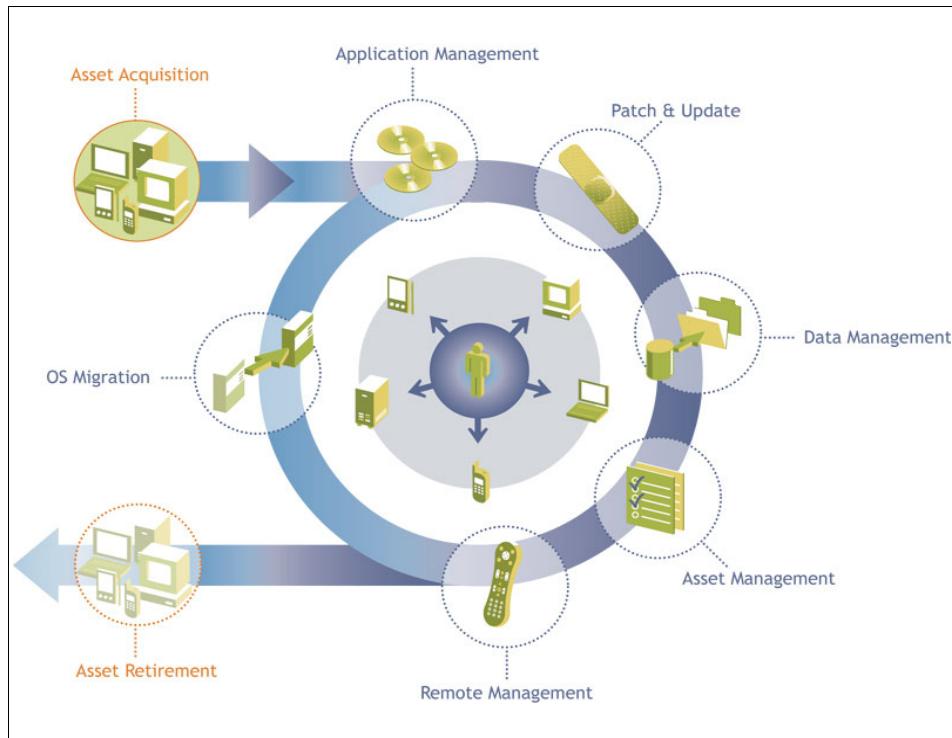


Figure B-3 ZENworks Suite lifecycle management²

ZENworks Linux Management

The management tasks available in the Linux Management module:

- ▶ Delivering packages
- ▶ Locking down client desktop settings
- ▶ Defining preboot activity
- ▶ Software and hardware inventory
- ▶ Manage remote clients
- ▶ Monitor events
- ▶ Generate reports

The system architecture of the ZENworks Linux management server is shown in Figure B-4 on page 266. The server consists of an application running on top of a database and using an http/https interface to communicate. The http/https

² Reproduced with permission from the “ZENworks 7 IT Management Suite” overview (<http://www.novell.com/products/zenworks/overview.html>)

interface runs the Web interface for management on one side and is used by the client to communicate with the server. The server can also be managed through the **rcman** command-line interface.

Previous to version 7, the client would run the **rcd** daemon. As of version 7, the daemon is now the **zmd** daemon. This daemon can be controlled by the **nove11-zmd** command.

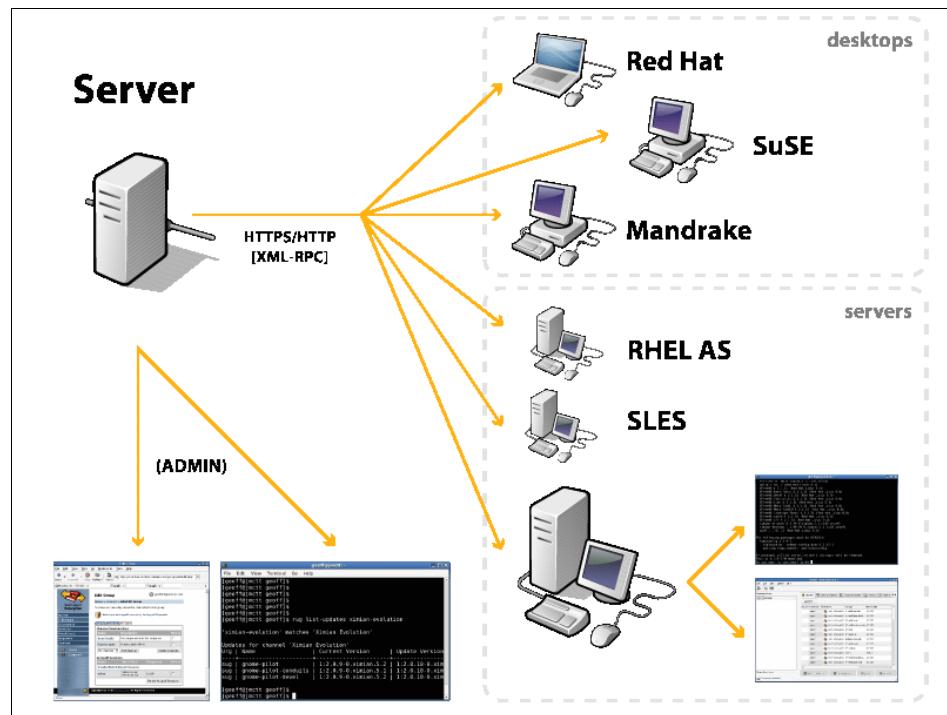


Figure B-4 ZENworks Linux management architecture³

ZENworks Terminology

In this section, we describe several terms used with ZENworks Linux management:

- ▶ Machine
- ▶ Group
- ▶ Machine set
- ▶ Channel
- ▶ Transaction
- ▶ Activation key

³ Reproduced with permission from "ZENworks Linux Management 6.5 Administrator's Guide" (<http://www.novell.com/products/zenworks/linuxmanagement/>)

We describe each of these concepts in some detail in the ZENworks Linux management context. Some of these same terms exist as a concept in the satellite server as well (see “RHN Terminology” on page 259), but might have a slightly different meaning.

Machine

Machine is the term used for a client system. The ZENworks server collects a lot of information about the system through the `zmd` daemon. This information can be used to search machines, but also to create sets on which to perform transactions.

Group

A group of machines is a collection of machines that can share certain properties, such as administrators or channels. The group is used to segment an enterprise into smaller entities, for example, to limit the scope of administrators.

Machine set

The machine set is like a group. Only one set exists and it is populated from all machines manually or by a search on properties of the machines. The machine set can contain all machines with a certain graphics card, or all machines needing a specific update and not having installed it yet. The set is a very powerful mechanism to perform a change across the enterprise.

Channel

A channel is a collection of RPM packages or package sets for a specific distribution or architecture. The package set is a collection of packages that is handled by ZENworks as a single package. It can have a version and dependencies.

Transaction

Transactions are generally updates that can be performed on machines, groups, or the machine set. The transactions can be scheduled. A feature in ZENworks, for example, is the ability to do a “dry run” of a transaction on the server to decrease the load on client machines.

Activation key

The activation key is the primary mechanism to register machines in ZENworks. The key is generated and certain properties are given to the key, such as groups, channels, and administrators. When a machine is activated using this key, it gets all these properties automatically in ZENworks. So the activation key used to activate the client more or less sets its role in the organization.

Usage examples

This section describes standard actions in the ZENworks Linux management server. This shows a small example of how the tool works and what it can do. We describe these actions:

- ▶ Creating a group
- ▶ Adding a channel with packages
- ▶ Creating an administrator
- ▶ Adding an activation key

An evaluation copy of ZENworks Linux management can be obtained from:

<http://download.novell.com>

On first login using the administrator account, the Web page of ZENworks Linux management looks similar to the window shown in Figure B-5 on page 269.

Different areas of the server are reached through navigation links on the left side of the page. These links are present on all pages to enable quick navigation through the Web-enabled administration of the ZENworks Linux management server.

Creating a group

A group is created by going to the Group administration page. After selecting Create New Groups, the necessary information is entered on the next page. After saving this page, the Edit page for the new group is reached. Through the tabs on this page, it is possible to edit all of the properties of the group, for example, administrators or channel permissions.

Adding a channel with packages

To work with channels, we go to the Channel administration page. Once there, click the Create New Channels link and on the next page, enter the Name, Description, and Alias for the new channel. After saving the new channel, the Edit page for this channel is shown. To add a package to the channel, we click the Add Package link. Saving this page adds the package for the selected platforms to the channel.

Create an administrator

First, go to the Account administration page. Once there, follow the link Create new administrator and fill in the needed information. Saving this information displays the Edit page, where the groups and channels that the administrator wants to administer can be added to the new administrator.

Add an activation key

The activation code pages are not mentioned in the navigation menu. To get there, you go to Server, and on that page, choose Activation codes. You then get the Activation administration page. Here you can create either a reusable or a single-use activation. The single-use activation becomes invalid after one machine has been activated using that code. The reusable activation key can be used many times and is the ideal way to activate a collection of systems with the same function. Following the Create New Reusable Activations link, we come to a page allowing you to enter the description and generate a value for the key. On the Edit page for the activation key, we can set the groups, channels, and administrators that are going to be set for every machine using this key to activate.

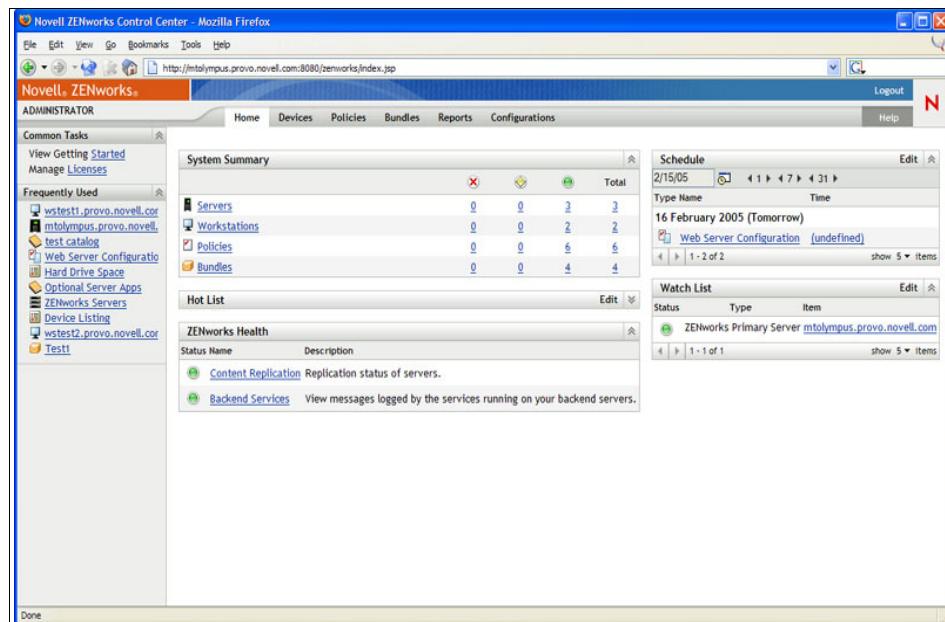


Figure B-5 ZENworks user interface window

Webmin

Webmin is a Web-based user interface for performing system administration on Linux clients. It also supports most popular UNIX operating systems, making it a good tool in a heterogeneous environment. It is a single system interface that runs in any browser that supports tables and forms.

Functionality

On Linux, Webmin supports well over 80 different modules that provide management user interfaces for just about anything you want to change on the client system, including many things that would be useful for servers as well. The modules are grouped into different manageable areas, including Webmin, System, Servers, Networking, Hardware, Cluster, and Others.

Just some of the modules supported:

- ▶ Webmin
 - Language and theme
 - Webmin configuration
 - Webmin users
- ▶ System
 - Bootup and shutdown scripts
 - Passwords
 - Software packages
 - Disk quotas
 - Users and groups
 - System logs
- ▶ Servers
 - Apache
 - DNS
 - DHCP
 - MySQL
 - PostgreSQL
 - Samba
 - FTP
- ▶ Networking
 - Network
 - Firewall
 - ADSL
 - NFS
 - Kerberos
 - IPSec
 - NIS
- ▶ Hardware
 - GRUB
 - CD Burner
 - Raid
 - LVM

- Partitions
- Printers
- Cluster
 - Heartbeat Monitor
 - Cluster users and groups
- Others
 - Command shell
 - System and service status
 - Upload and download

Usage examples

In this section, we describe a small subset of the basic functionality of Webmin. The standard actions we describe are:

- Adding a user to the managed client
- Remote command execution on the client system

Adding a user

Creating a user using Webmin is started by selecting the System icon from the top action bar. Presented in the System tab is an icon for Users and Groups. Clicking this icon presents the Users and Groups tab, showing all the current users of this system. Select the Create a New User link at the top and the Create User tab is shown. Simply fill in all the fields on this page and select the Create button at the bottom of the page to create the new user. Once this user is created, Webmin returns to the Users and Groups summary tab, where you can see the new user as a part of the system users.

Remote command execution

Being able to remotely execute a command on another server can be extremely useful. To run a remote command, start by selecting the Other icon from the action bar at the top of the page. The Other tab then shows all the modules available in this group. Select the Command Shell icon and the Command Shell tab shows. Enter a Linux command in the box provided, select the Execute Command button, and the command is run on the system to which you are connected. The output of the command is presented on the Command Shell page. To see where the current path is, try issuing `pwd` first. You can use the `cd` command to change location of the command shell to another directory.

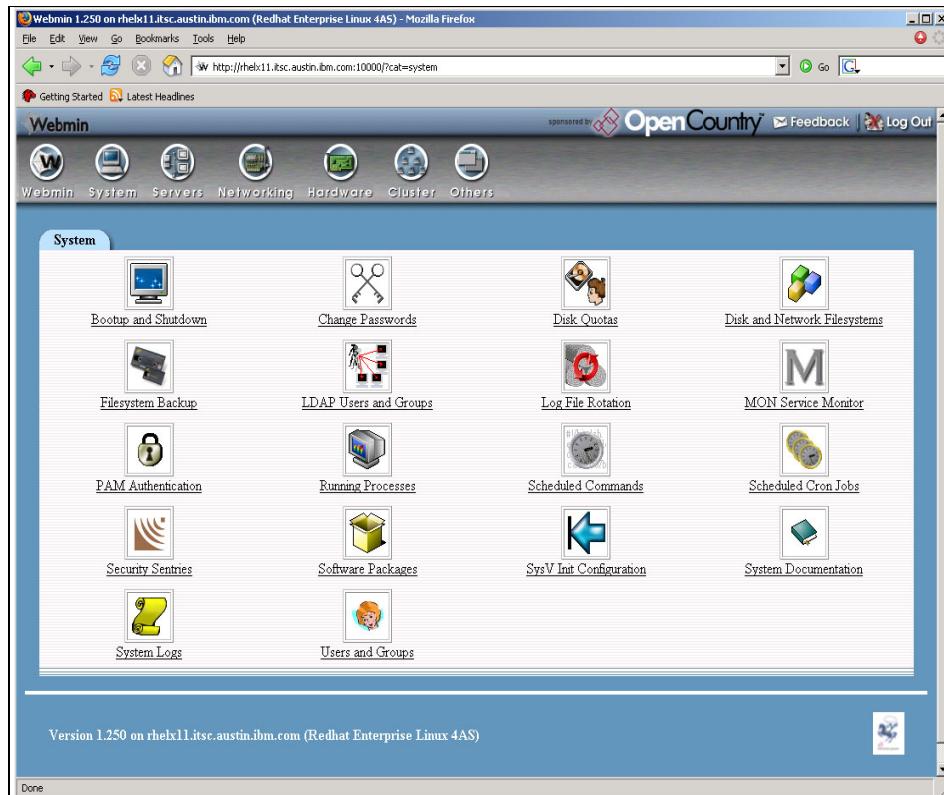


Figure B-6 Example of a Webmin user interface window

More Information

More information about Webmin, including project details, documentation, and access to the latest version available for download can be found at the Webmin Web site:

<http://www.webmin.com>

Other important tools

Tivoli management tools

Tivoli has a full set of management tools available for helping you handle the complexity of managing clients in an enterprise environment. Linux clients are supported by most of Tivoli's management tools, including tools that help monitor, manage, and deploy Linux client systems. Refer to the Tivoli Web site

for more information about the enterprise management tools offered and their features:

<http://www.ibm.com/software/tivoli>

Open Country OCM Suite

Open Country provides an enterprise level management tooling platform. It addresses many traditional large enterprise system management requirements. Some of those include: provisioning, system management, monitoring, updates, and asset tracking. For more information, see:

<http://www.opencountry.com>

OpenNMS

OpenNMS is an open source distributed enterprise network and system management platform. The OpenNMS framework is SNMP-based and handles discovering network nodes and services, polling for their availability, and recording the system information for immediate use or creating thresholds. OpenNMS also gathers and reports network and system service events. For more information about the project, to view a demo, or to download the platform itself, refer to the OpenNMS Web site:

<http://www.opennms.org>

SBLIM

SBLIM is an acronym for Standards-Based Linux Instrumentation for Manageability. This open source project provides standard Web-Based Enterprise Management (WBEM) components. The goal of this project is to provide a complete Open Source implementation of a WBEM-based management solution for Linux. WBEM-based management is a growing method of managing clients using a object-based approach defined by the Distributed Management Task Force (DMTF).

The SBLIM components or packages help to provide a unified management strategy among Linux clients with distribution specific configuration files and management tools. It also provides an abstraction to allow for simplification of common management tasks. The result is a common management method for handling system management tasks across different Linux clients.

You can read more about SBLIM project, the available packages, and the underlying technology at the SBLIM Web site:

<http://sblim.sourceforge.net>

WBEM-SMT

Within the previously mentioned SBLIM project, a component of particular interest is WBEM-SMT. WBEM-SMT stands for WBEM Systems Management Tasks. The goal of the WBEM-SMT component is to improve the management features and ease of use of the Linux operating system. WBEM-SMT provides a set of task-oriented user interfaces that exploit the SBLIM WBEM-based instrumentation and automate many aspects of configuration.

One of the tasks that WBEM-SMT has provided is an interface for configuring DNS. An example of the DNS task can be seen in Figure B-7.

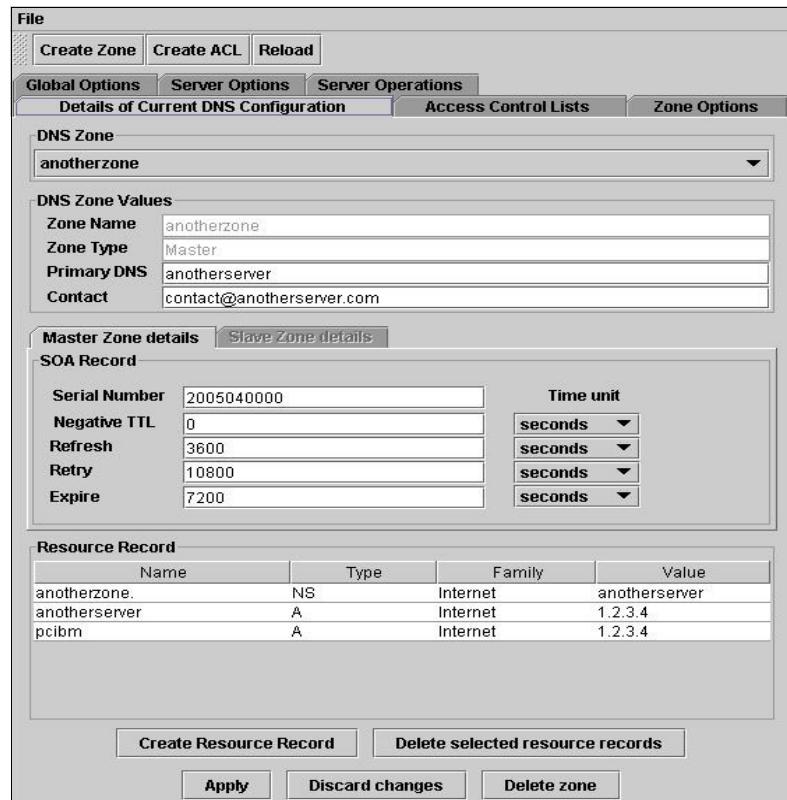


Figure B-7 Example of a WBEM-SMT user interface window

More information about the WBEM-SMT package can be found at the SBLIM Web site at:

<http://sblim.sourceforge.net/wbemsmt.html>



C

Automating desktop migration using Versora Progression Desktop

This appendix describes the benefits of using an automated migration tool, such as Versora's Progression Desktop. It then expands upon the three ways to run Progression Desktop; manually with the GUI, manually with the command line, and automatically with a systems management tool. Finally, we describe Progression Desktop's extensible architecture.

Benefits of an automated migration

A manual desktop migration is a time-consuming task. When dealing with a standard office or advanced office user, there are a myriad of technical considerations associated with the migration of data and application and system settings. A sample migration of e-mail, Web browser settings, and data is shown in Chapter 8, “Client migration scenario” on page 173. Since a manual migration is estimated to require several hours of technician time per desktop, it could feasibly take an organization weeks or even months to complete. A Windows to Linux migration requires a technician to be proficient on both platforms with an understanding of the differences in file structures, applications, application settings, and system settings.

By using an automated tool in a large migration, a test machine or lab can be used to completely lay out the migration strategy. Any issues with the migration can be fixed in the test lab and rolled into the migration process. This allows for the least amount of down time when it is time for the actual migration to begin. Use of automated tools provides predictable results that can be repeated simultaneously on desktops throughout the organization. Also, automated tools do not require the technician to have the same level of technical expertise required for a manual migration, because much of that knowledge is built into the program. When used in conjunction with systems management tools, certain automated tools can allow one technician to migrate up to one hundred desktops in the same time period that a single manual migration requires.

What is Progression Desktop

Progression Desktop by Versora is an example of a set of applications and tools that together can automatically migrate system settings, application settings, and data from Windows to Linux-based clients. The Web site is:

<http://www.versora.com>

System settings include settings such as desktop wallpaper, keyboard and mouse settings, screen saver settings, system sounds, network shares, dial-up connections, and fonts. System settings can be applied to both GNOME and KDE desktop environments. Application settings include e-mail-related settings such as messages, address books, and calendar entries; Web browser-related settings such as favorites, cookies, and the browser home page; word processing-related settings such as custom dictionaries, default templates, and AutoCorrect settings; and instant messaging settings such as accounts and buddy list preferences. In each of these application categories, settings can be migrated from any given source application to any of the available destination applications. Some example mappings can be found in Table C-1 on page 279.

Documents can be chosen by folder or by file, or even by file type. The documents can be copied as they are or sent through a conversion process based on file type to ensure complete compatibility on the target desktop. Progression Desktop also allows files to be excluded based on location or file type; for instance, a corporation might want to migrate the users' My Documents directories, but exclude all MP3 files or other files that violate corporate policies.

Table C-1 Some applications supported by Progression Desktop

Category	Source applications	Destination applications
E-mail	Microsoft Outlook Microsoft Outlook Express Mozilla Mail Mozilla Thunderbird	Novell Evolution KMail Mozilla Mail Mozilla Thunderbird
Web browser	Microsoft Internet Explorer Mozilla Mozilla Firefox	Konqueror Mozilla Mozilla Firefox
Word processor	Microsoft Word OpenOffice.org Writer	OpenOffice.org Writer
Instant messaging	AOL Instant Messenger GAIM	GAIM Kopete

How to migrate with Progression Desktop

There are three ways to migrate clients with Progression Desktop:

- ▶ Using the provided GUI
- ▶ Using the command line with templates
- ▶ Using a systems management tool to automatically invoke the command line with templates

Each of these methods follows the same basic steps, as shown in the process diagram in Figure C-1 on page 280. First, the Windows-based executable of Progression Desktop is run on the source desktop. This process retrieves information from each of the selected source applications and stores these settings into a neutral source. This neutral source, called a Platform Neutral Package (PNP), then must be made available to the destination machine. This step is accomplished by copying the file to a network share, or directly to the destination machine. Finally, the Linux-based executable of Progression Desktop is run on the destination machine. This process gathers information from the PNP and applies the settings to the appropriate location for each of the target applications.

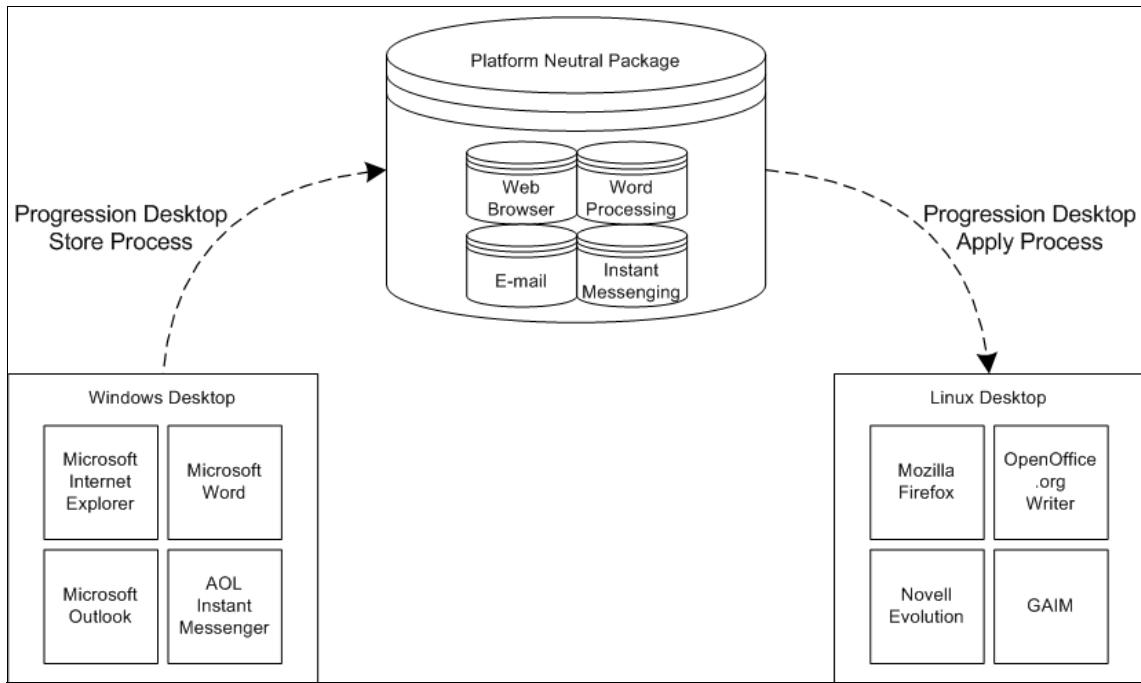


Figure C-1 Progression Desktop neutralization process

GUI

The GUI allows users to perform the migration via a “Next-Next-Finish” style wizard interface. The user is presented with a series of questions that ask which system settings, application settings, and files from the Windows desktop should be captured and stored for migration. For instance, in Figure C-2 on page 281, a list of detected applications is shown, and the user is choosing to migrate settings from Microsoft Outlook, AOL Instant Messenger, Microsoft Internet Explorer, and Microsoft Word.

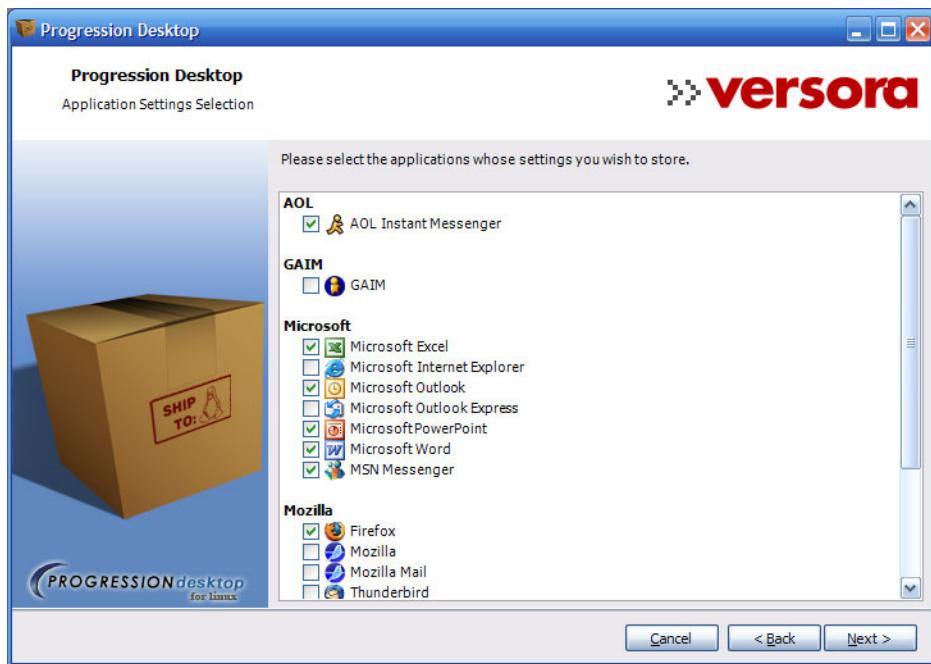


Figure C-2 Progression Desktop storing settings on Windows

Next, on the Linux desktop, the user interfaces asks to which destination applications and locations those settings should be applied. In Figure C-3 on page 282, the application is asking for destinations for settings that were stored in Figure C-2. The user has chosen to map Microsoft Internet Explorer to Mozilla Firefox, AOL Instant Messenger to GAIM, Microsoft Outlook to Novell Evolution, and Microsoft Word to OpenOffice.org Writer.



Figure C-3 Progression Desktop applying settings to Linux

Performing a migration using the GUI is the simplest approach. However, because this approach requires installing Progression Desktop on each machine (either off of the CD or a network share), this option does not scale well to large environments. This approach is probably best suited for migrations of smaller groups of systems (ten to twenty). For larger groups, you should consider the other approaches.

Command line (with templates)

Templates are defined policies, which tell the Progression Desktop engine what items are to be saved from the Windows machine and what items are to be applied to the new Linux machine. Templates are used as part of an automated migration processes. They also provide a method of efficient migration without having to step through the Progression Desktop user interface each time you want to perform a migration on a specific machine. Templates are simple XML files, which can be created by editing the default templates provided, or by capturing a temporary template that is created during a migration with the GUI.

Once the template has been created, a simple batch file or shell script can be created, which installs Progression Desktop automatically and then runs it using

the specified template. This process can then be run with a single command on each workstation or run using a logon script, as your environment allows. This option is probably best suited for migrations of up to forty systems.

Systems management tool

It is possible to manage the migration of large environments through the use of third-party systems management tools. Migrations managed by a systems management tool use the same command line approach discussed in the previous section. The systems management tool needs to be configured to automatically install and run Progression Desktop on both the source and destination machines. For information regarding one systems management tool, IBM Tivoli Provisioning Manager, see the IBM Redbook *Provisioning On Demand Introducing IBM Tivoli Intelligent ThinkDynamic Orchestrator*, SG24-8888, available at:

<http://www.redbooks.ibm.com/abstracts/sg248888.html?Open>

For information about how to set up a customized workflow with which to run Progression Desktop or other custom jobs with IBM Tivoli Provisioning Manager, see the IBM Redbook *Developing Workflows and Automation Packages for IBM Tivoli Intelligent ThinkDynamic Orchestrator*, SG24-6057, available at:

<http://www.redbooks.ibm.com/abstracts/sg246057.html?Open>

The general steps of configuring a systems management tool to trigger a migration process are similar on most platforms:

1. Create a template.

The first step in a migration is to create a template; for larger migrations, you might choose to have different templates for different departments.

2. Create the Platform Neutral Package.

You then create a job, which automates the installation of Progression Desktop, runs it using the specified template, and then copies the resulting Platform Neutral Package (PNP) to a network server.

3. Apply the Platform Neutral Package.

The apply job would be similar, where the systems management tool copies the PNP from the server, installs Progression Desktop, and then runs the apply process using the same template.

If your systems management tool allows for multiple machines to be included in the same job, you could schedule the apply job to start as soon as the store finishes. If you are planning on reusing hardware (an “in-place” migration), some systems management tools can even be used to install the Linux operating system in between the store and apply phases, leading to a complete automated

migration and installation with the click of a button. Because this option requires the most setup before migrating, it is best suited for migrations of more than forty machines.

Progression Desktop architecture

Some of the benefits to developing on Linux are the wide variety of higher-level languages available on the Linux platform. When developing for Progression Desktop, the majority of the engine code needed to be portable between the Windows and Linux versions. When looking for a high-level language that was equally supported on Windows and Linux, that provided robust run-time services, and that was able to natively access Windows and Linux specific technologies such as the Windows registry or GNOME GConf, the development environment that made the most sense to use was Microsoft Visual C#® (pronounced “C sharp”). While first created by Microsoft as a competitor to Java, the Mono project has implemented the language and run time to provide a Linux compatible implementation as well. However, because a native user interface for all platforms was desired, a C#-based GUI program was not suitable for KDE-based desktops at the time of this writing. Instead, the Linux GUI uses another high-level, flexible language; Python. Python was chosen because in addition to its object-oriented nature and dynamic run-time flexibility, the PyGTK and PyQt libraries for developing graphical applications are quite mature. By creating common Python base classes for each step and deriving a GTK and Qt version from those classes, Progression Desktop manages to provide a native look-and-feel in its Linux-based GUIs with only a minimal amount of duplicated code.

One of the major design goals of Progression Desktop was extensibility. With many unique choices of applications in both Windows and Linux for each of the common and targeted migration data sets, Progression Desktop needed to be able to migrate to several different configurations. For instance, a user who had been running Microsoft Outlook might want to migrate to Evolution, while another user might migrate from Mozilla Mail to Thunderbird. In order to achieve this goal and be able to run on both Windows and Linux, Progression Desktop utilizes a plug-in-based architecture and XML data files running on the .NET platform. The three main technologies developed for this are the PNP file format, the Settings Packages, and the plug-ins such as Transforms and Accessors.

PNP Files

Progression Desktop stores extracted settings in a Platform Neutral Package, or PNP. PNP files are a standard Zip file, which can be opened with your favorite Zip file editor, such as FileRoller, Ark, or WinZip. Figure C-4 on page 285 shows

an example PNP file open in FileRoller. Inside of the Zip file are XML files describing the PNP, your previous environment, and the settings that were stored. Stored settings are grouped by category, with each application being a root level node in the XML file. The PNP also contains subfolders, which are either files that are referenced by a setting, or files that were explicitly selected by the user when storing.

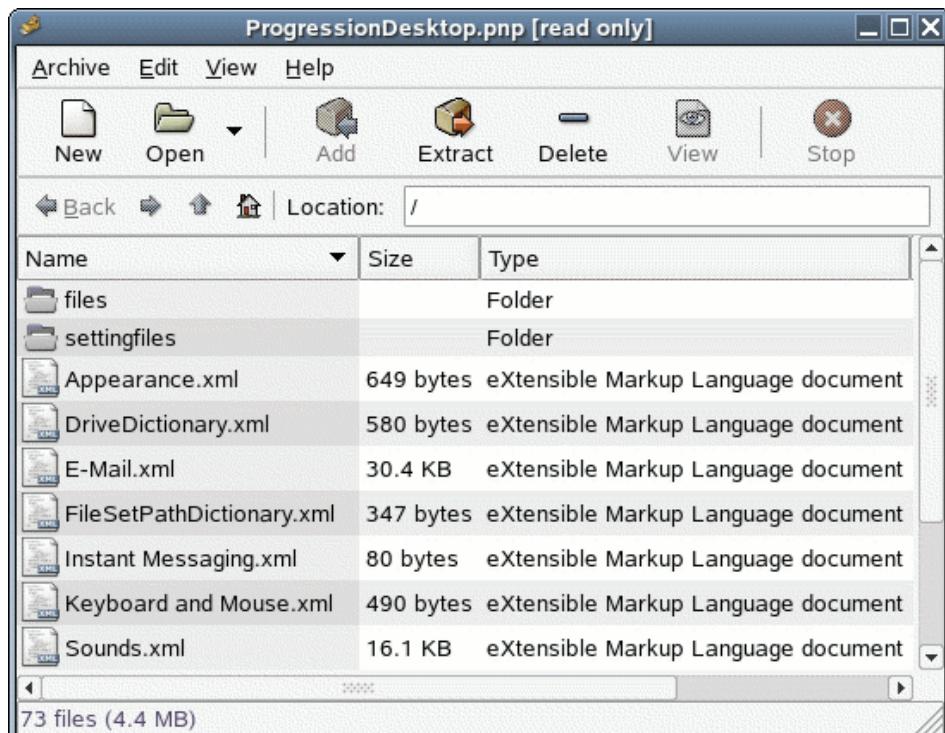
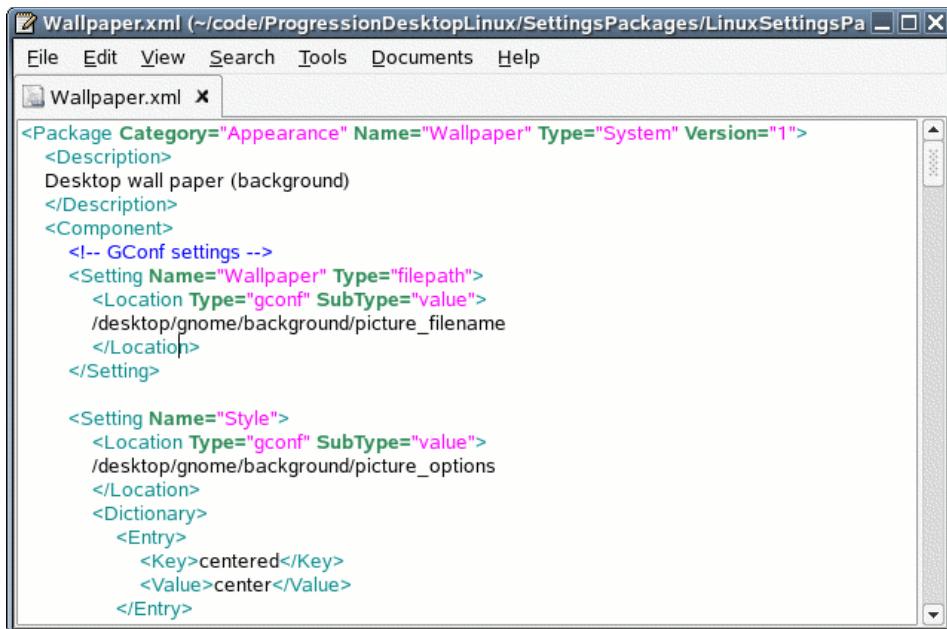


Figure C-4 An example PNP file open in FileRoller

Settings Packages

Progression Desktop is extended to support additional applications through Settings Packages. A Settings Package is an XML file that describes an application and its settings. Settings Packages provide a flexible data definition format that can be used to migrate anything from custom in-house applications to well known third-party applications. Versora-maintained Settings Packages are bundled into a single DLL in the root folder, but additional settings packages might be added in the "SettingsPackages" subfolder. This gives users the ability to easily extend their own migration without having to contract out to Versora for new application support. Settings Packages include information that tells

Progression Desktop how to detect if the application is installed, as well as where settings for this application are stored. Figure C-5 shows a simple settings package, including the necessary values to apply desktop wallpaper and its location to the GNOME desktop.



```
<?xml version="1.0"?>
<Package Category="Appearance" Name="Wallpaper" Type="System" Version="1">
  <Description>
    Desktop wall paper (background)
  </Description>
  <Component>
    <!-- GConf settings -->
    <Setting Name="Wallpaper" Type="filepath">
      <Location Type="gconf" SubType="value">
        /desktop/gnome/background/picture_filename
      </Location>
    </Setting>

    <Setting Name="Style">
      <Location Type="gconf" SubType="value">
        /desktop/gnome/background/picture_options
      </Location>
      <Dictionary>
        <Entry>
          <Key>centered</Key>
          <Value>center</Value>
        </Entry>
      </Dictionary>
    </Setting>
  </Component>
</Package>
```

Figure C-5 A simple settings package describing the GNOME wallpaper

In order to avoid a complex mapping from each individual source application to an appropriate destination application, Settings Packages map settings into a common neutral format. Progression Desktop then automatically translates settings between those with the same in the source and destination packages.

For instance, in order to migrate personalization settings from Internet Explorer to Konqueror, both packages have a setting named “Homepage”. In Internet Explorer’s case, the “Homepage” setting is defined as coming from the registry at a specific path, while in Konqueror the setting points to a path of an INI file and the section and key inside that INI. Once those two paths have been found and named the same, Progression Desktop handles the details of extracting the setting from the registry, storing it inside of the PNP, and then inserting the value into an INI on the destination. If the setting had pointed to a file, then Progression Desktop would also automatically extract the associated file and insert it in the appropriate place on the destination machine. Settings Packages can also include a Dictionary, which is a mapping of specific settings to neutral types. For instance, Internet Explorer uses integers between 0 and 4 to specify how often a page should be checked for updates, while Konqueror uses words such as Cache

and Refresh. Once the dictionary is specified, Progression Desktop does all of the work of mapping values between application-specific values.

Plug-Ins

While Settings Packages provide extra flexibility, sometimes more control over the migration process might be necessary. Progression Desktop has a plug-in architecture that allows you to extend its capabilities further. Plug-ins can be written in any .NET language and are automatically used by Progression Desktop when placed in the plug-ins directory. There are several types of plug-ins that Progression Desktop can handle, including Transforms, Accessors, and FileConverters.

Transforms

Transforms are used in cases where a simple translation needs to be performed between applications. For instance, one application might store how often to run an auto-save feature in minutes, while the destination application stores that number in seconds. When extending Progression Desktop to migrate your in-house applications, you can use Transforms if your application's settings logic differs between the Windows and Linux versions.

Accessors

Accessors allow a settings package to retrieve structured data from any data store. Some Versora-provided Accessors include the ability to read and write from the Windows registry, GConf, and INI files. When extending Progression Desktop to migrate your in-house applications, you can write an Accessor if your applications use a nonstandard format when storing settings.

FileConverters

FileConverters are used to convert any file type dynamically. For instance when Windows shortcut files are stored, instead of storing the binary .lnk file, an XML file containing the information about that shortcut is stored. Then when applying to a Linux environment, a .desktop file is created, pointing to the new path for the document that the shortcut had been pointing to. When extending Progression Desktop to migrate your in-house applications, you can write a FileConverter to convert documents which cannot be opened in the target Linux environment.

Enterprise Source License

Progression Desktop is licensed under Versora's Enterprise Source License. The license grants users of Progression Desktop access to the source code, and the right to read and modify it for internal use. While this license does not allow for redistribution of Progression Desktop, it does ensure the user that the user

has access to extend the behavior of Progression Desktop to meet the user's migration requirements.



D

Multi-station computing deep dive using Useful Desktop Multiplier

In this appendix, we provide more detailed information related to multi-station deployment strategies using Useful Desktop Multiplier and IBM IntelliStation Pro systems.

We present the following topics:

- ▶ “Deploying multi-station solutions on the IBM IntelliStation platform” on page 290
- ▶ “Software requirements and installation considerations” on page 296
- ▶ “Deployment considerations” on page 298
- ▶ “Additional system management considerations” on page 303
- ▶ “Case study one: General office desktops for a 25-user office” on page 306
- ▶ “Case study two: Transactional desktops: Public computers for a city library” on page 308
- ▶ “Understanding how multi-station computing works: Exploiting the flexibility of X Window System” on page 309

Deploying multi-station solutions on the IBM IntelliStation platform

The amount of computing power available to the user of a computer system is determined by a number of factors including CPU speed, system memory, and storage. Those measures also apply to multi-station systems, with some additional considerations.

As with single-station PCs, the number of simultaneous users or processes that can be supported is bound not just by the amount of computational power that is available on the PC, but also by how long it takes to service a particular process and the number of concurrent running processes. Obviously, the total number of concurrent processes increases with the number of users sharing a multi-station computer. Less obvious is how those processes and other factors interact to affect the user experience on each station.

In this section, we consider a number of hardware and software factors that can determine the total number of users who can effectively work from a single multi-station Linux computer.

Important: In deploying a multi-station computing solution, you must take care to properly test client interactivity and overall performance. Defining the acceptable level of client performance supported by the multi-station host platform with the maximum number of clients all logged on and running their key business applications simultaneously, and provisioning the host systems to meet that goal, is a critical design requirement for success of the solution. See “Processor requirements” on page 293.

Hardware requirements

This section describes the hardware requirements and related considerations for deploying multi-station Linux desktop systems.

Due to the widespread availability of multi-headed video cards and USB input devices, many current personal computer systems can be configured for two local users simply with the addition of a new set of user input and output devices including a keyboard, mouse, and monitor. By adding video cards, several users can be simultaneously supported on properly configured systems.

A multi-station computer can be built using standard PC components; however, more robust, higher performing platforms, such as the IBM IntelliStation A Pro and IntelliStation M Pro workstations, are ideal. Most common configurations incorporate dual-head cards using ATI or NVIDIA chipsets, a generous amount of RAM, and an x86, EM64T, or AMD64 processor and motherboard with a

number of free PCI/AGP/PCI-E expansion slots. Criteria for specifying equipment to meet each of these requirements is described in more detail below.

Dual-head video cards

Although multiple single-head video cards can also be used, dual-head video cards are preferred for multi-station systems both to maximize the use of workstation expansion slots, as well as for taking advantage of the cards' on-board screen mirroring or screen spanning capabilities. In general, any recent single or dual-head video card is compatible with multi-station configurations. However, cards based on ATI's FireGL workstation or Radeon consumer chipsets, or NVIDIA's Quadro workstation or GeForce consumer chipsets are known to work well and provide good driver support for multi-station configurations and other multi-monitor setups. Some video cards might work well individually but not in concert with other video cards installed in the same system. Be sure to test combinations of video cards in all anticipated multi-station configurations prior to establishing standard configurations. See Figure D-1 on page 292.

Note: Some video cards do not support 3D rendering on multiple screens simultaneously. If 3D video rendering is critical to the utility of the multi-station systems, individual models of video cards should be tested for 3D compatibility.

Userful's Desktop Multiplier system does not currently support 3D graphics acceleration software.

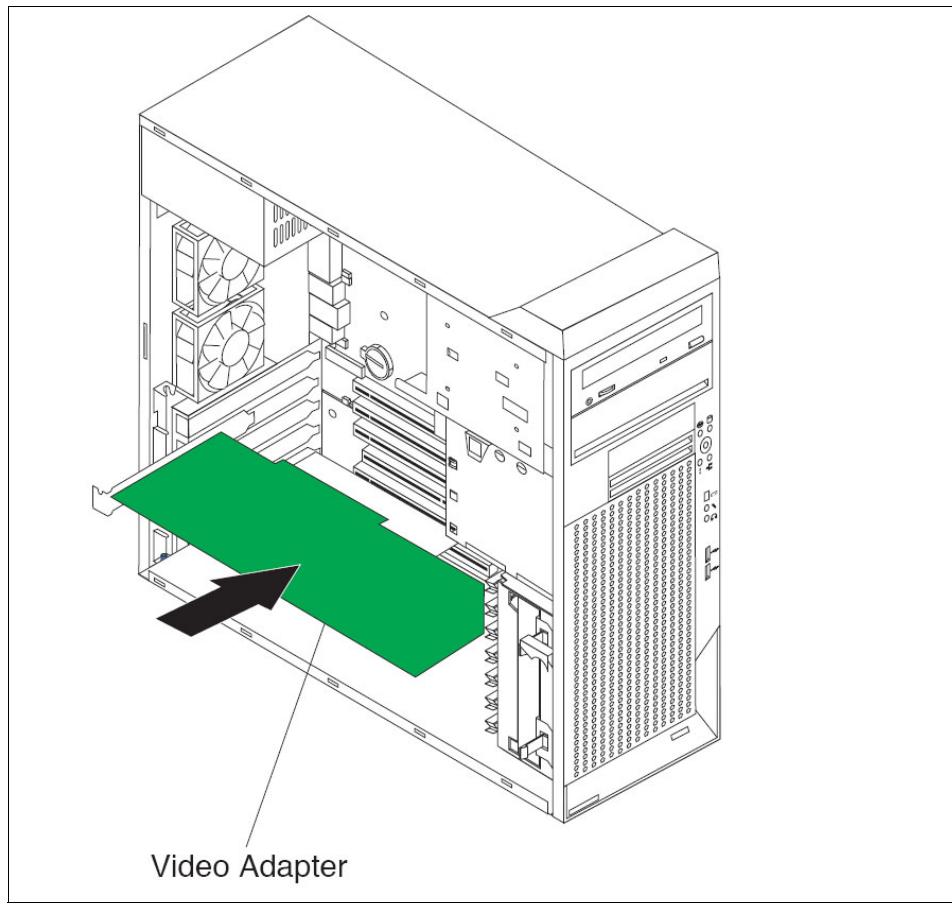


Figure D-1 Adding a video card to an IBM IntelliStation A Pro workstation

PCI/AGP/PCI-E expansion bus

The number of video cards (and hence, video heads) that you can install in a workstation computer is limited by the number of expansion slots on the workstation, and to a lesser extent, the bandwidth of the expansion bus. A single 32-bit PCI bus at 33 MHz is theoretically capable of transferring data at 133 MBps, or approximately the data required to display full-screen software-decoded DVD movie streams simultaneously to six video heads with overhead (720 pixels x 480 pixels x 2 bytes/pixel x 24 frames per second x 4 video heads=99.5 MBps). This worst case PCI bus utilization scenario far exceeds normal workstation video display usage.

In new workstations, each lane of a PCI Express expansion bus can carry up to 250 MBps, which supports the data transfer for more than ten software decoded

DVD movie streams. With up to 16 lanes per PCI Express video card, the number of available expansion slots, rather than the bus bandwidth, is most often responsible for limiting the total number of video displays that can be connected to a multi-station system. Current IntelliStation models typically have three to five free PCI-X or PCI Express expansion slots, supporting a total of 6 to 10 single-monitor stations, or 3 to 5 dual-monitor stations per computer. In either configuration, performance of the multi-station system is not strictly limited by the expansion bus or video cards.

When deploying high-density desktop clusters to many users within the same area, PCI hardware expansion units can allow additional video cards and peripheral devices to be added to facilitate as many directly connected users as the systems memory, CPU, and PCI bus bandwidth permit.

Processor requirements

The optimal number of users supported by a particular CPU in a multi-station system cannot be reduced to a simple formula, because overall performance issues are largely dependent on the nature of usage. A IntelliStation system with a dual-core 2.6 GHz processor could reliably support over 20 users, each simultaneously running “low utilization” applications such as terminal emulators or simple data entry applications. The same system could support far fewer users, perhaps 4-8 users, with full graphical environments performing office tasks such as word processing and Web browsing. The same system used by programmers who frequently compile and test code could likely support 2 to 4 users without noticeable lag.

One method to estimate CPU usage is to examine the CPU load average, that is the number of tasks waiting for execution time at any given point. On a single core processor system with no HyperThreading, a load average of 1.0 indicates that one process is always waiting for execution time. This is the ideal condition. A load average of more than 1.0 indicates that some processes are waiting for execution time, while a load average of less than one indicates that CPU cycles are being unused or idle.

To determine the optimal multi-station system configuration for a particular usage pattern, profile the load average for that class of users at peak usage on their current single-user systems. For best results, the load average during peak usage of all users on a multi-station system should not exceed 2.0 per processing core. A load average value of 2.0 during peak usage indicates that the CPU is, in general, able to serve all regular priority user processes as required by user applications, while not ignoring the usually more numerous background system processes. Faster CPUs or additional processing cores can increase the available computational power to decrease the amount of time it takes to service a process, and therefore increase the number of users that the system can support.

Userful's Desktop Multiplier software shares the opportunity for processor time at the X server layer equally for all users, but it does not attempt to modify the way in which the underlying Linux kernel handles processes and threads. Since most line-of-business applications do not spawn multi-threaded CPU-intensive processes, the combination of a dual-core processor and an SMP kernel should easily provide acceptable levels of performance for each client on the multi-station system.

As with a single-station Linux desktop or thin client system, if a CPU-intensive application or event is triggered by a user on a multi-station system, all other running user processes might experience a negative performance impact. Hardware-related CPU usage spikes arising from processes running in the system context at a higher priority than user processes can negatively affect the processes of all users. Various scheduling utilities exist to set priorities for application and system-level events, such as `nice` and `renice`, which are standard process management tools available with most Linux distributions. There are numerous other special priority schedulers available from various sources.

Persistent storage

When considering local storage for multi-station deployments, hard disk speed and cache size tend to be more important factors than total disk size. On multi-station computers with networked user file storage, the swap partition and temporary user files are unlikely to ever occupy more than 10 GB in total, but that usage would be the result of many hundreds of small files. Many user applications and system processes store temporary working files in the `/tmp` directory. With a single user, relatively few processes read and write files to that directory, and it is unlikely to grow to more than 1 GB in size with any regularity.

As with a single user system, responsiveness of applications such as a Web browser or OpenOffice benefits from a high-performance hard disk. Applications often make and rely on many small temporary files for each open document, which tie application responsiveness to reading and writing these small files from the hard disk quickly. On a multi-station, multi-user system, overall performance benefits greatly from a large disk cache and high spindle speed on the hard disks on which `/tmp` and swap files are hosted. If persistent local storage is required, local storage capacity should be provisioned according to the needs of each user, as with any other desktop computer system.

Memory

Assessing memory requirements for multi-station Linux setups is no different from assessing the memory requirements for a single user Linux workstation, with the exception that the underlying operating system memory image is only loaded once. Red Hat Enterprise Linux Workstation Version 4 loads a

full-featured configuration in under 256 MB of non-swapped memory while Ubuntu Linux loads a complete Linux operating system with GUI in under 350 MB of non-swapped memory. Each additional user only requires an additional 20-40 MB of system memory to expand the scope of USB device handlers, X (see “Understanding how multi-station computing works: Exploiting the flexibility of X Window System” on page 309), and the window manager, because no additional copies of X, the kernel, or any other system software need to be loaded. As with single station systems, each user application has memory requirements that can be profiled with standard tools such as ps, top, and the contents of /proc/<process>/status.

Individual user applications such as OpenOffice Writer and Firefox browsers use no more or less memory on a multi-station system than on a single user system, which is easily profiled with utilities such as ps and top, and such application memory requirements are easily provided on a multi-station system. In a typical office deployment, each user typically requires between 256-512 MB of non-swapped memory for running local applications.

Device support

Common devices such as keyboards, mice, barcode scanners, magnetic stripe readers, and most USB storage devices have mature multi-station (or multi-user) aware driver support. For other devices, multi-station device support can currently be limited by driver module implementations that are not multi-user aware. For example, many Web cameras and USB audio devices function in multi-station configurations, but access to these devices is shared by all users.

While non-multi-user aware devices can be used in multi-station systems, the issue of multiple users contending for access to such devices should be reviewed and fully tested before deploying such devices. A simple procedure outlining access to a shared device such as the workstation’s CD burner is feasible for most devices.

If controlling access to shareable devices is a requirement of the workplace environment to meet policy or regulatory requirements, integrated solutions are available. For example, Useful’s Desktop Server system (which integrates Desktop Multiplier functionality with Fedora Core software and specialized access control features) can quickly form logical workstations and assign exclusive access to almost any non-privileged resource to a particular station, overcoming many limitations of a non-integrated solution.

Shared access to removable storage

As mentioned, managing access to shared devices on a multi-station computer can present unique technical and management challenges. Because most computer systems are not configured with multiple local users in mind, they do

not include multiple optical or floppy drives. Therefore, a combination of policies and technology enhancements might be needed to ensure appropriate access and data security.

Providing shared access to removable storage devices means that any user on the local multi-station system can access the contents of any disk or USB memory key connected to the shared system. This can pose a security concern if the enterprise data handling policy permits confidential information to be stored by employees on removable media and could raise user concerns with regard to identifying and working with multiple connected devices of the same kind. Denying access to removable storage devices at the system level solves both of these concerns, but can frustrate employees who occasionally access data stored on removable media.

Prior to deploying multi-station systems, be sure to inform users about best practices and policies regarding shared drive access.

Selecting your IBM IntelliStation model

The technologies that enable multi-station computing are based on standard PC hardware so a variety of hardware configurations are capable of providing stations to multiple users. However, there are distinct advantages to selecting and implementing multi-station solutions based more scalable workstation class platforms.

IntelliStation Pro systems with Intel Xeon® or AMD dual-core Opteron processors can be configured for up to 16 GB of memory along with fast storage subsystems to help ensure acceptable levels of performance for all client workstations. Such systems are appropriate for supporting a moderate number of client workstations to power users, or higher numbers of client workstations for users of lightweight applications such as terminal emulators.

Software requirements and installation considerations

The software requirements for multi-headed or multi-station Linux are easily met by most Linux distributions. Typically, multi-station Linux solutions require Linux Kernel Version 2.6, X.Org version 6.8.2 or later, or Xfree86 4.3.0 or later and a recent Gnome or KDE build.

Important note about X.org: Some older versions of X.org, including 6.8.2-31 are incompatible with multiple installed PCI video cards, and they might fail to launch an X environment. The current version of X.org (6.8.2-37) does not exhibit this problem.

Desktop Multiplier system requirements

In addition to this information, visit the Useful Desktop Multiplier product home page at the following address for the most up-to-date information about system requirements, installation, and answers to frequently asked questions:

<http://useful.com/products/dm>

As of this writing, Useful Desktop Multiplier is tested to be compatible with the following Linux distributions:

RPM-based distributions:

- ▶ Red Hat Enterprise Linux WS, Fedora Core 2, 3, and 4, SuSE 9.1, 9.3, and 10, Mandrake 10.0 and 10.1, Novell Desktop Linux 9 and 10, and CentOS 4

DEB-based distributions:

- ▶ Ubuntu 5.x and 6.x, Linspire 5.0, Xandros v3, and Debian 3.1 (Sarge)

Desktop Multiplier might also be compatible with other Linux distributions that support .deb and .rpm packages. Instructions for installing on non-supported distributions are available at:

<http://useful.com/products/dm-config>

Before adding Desktop Multiplier multi-station software to an existing x86 or EM64T Linux installation, sufficient video cards and USB ports must be installed on the shared computer to connect at least one monitor, one keyboard, and one mouse for each station. Since there are only three to six free USB ports on the IntelliStation, we recommend USB keyboards with integrated hubs. Powered USB hubs can be used to minimize the number of USB devices that need to be directly connected to the computer.

Note: Underpowering the USB bus can cause unpredictable behavior of USB devices. If extension cables are used, ensure that the distance between keyboard or other USB device and computer or powered hub does not exceed 3 m (10 ft.).

Desktop Multiplier software supports display resolutions of 640x480, 800x600, 1024x768, and 1280x1024 on each monitor.

Recommended BIOS settings:

- ▶ Enable all USB ports.
- ▶ Enable Legacy USB support. (Sometimes called "USB keyboard support"); otherwise, only the PS/2 keyboard works during the Linux boot process.

- ▶ Disable unnecessary integrated peripherals.
- ▶ Installing extra video cards increases load on the PCI bus; therefore, disabling any unneeded devices (for example, serial ports, parallel ports, and so forth) in the BIOS can improve performance and compatibility.
- ▶ Disable on-board video (see notes above).
- ▶ For installation on NLD 9 or kernels below 2.6, disable USB2 if you are planning to use hubs.

Desktop Multiplier software installation

Desktop Multiplier is available as an .rpm and as a .deb package and includes installation scripts for supported Linux distributions. Following the basic package installation, a reboot is required, at which point a configuration wizard helps to complete the installation process.

To download Desktop Multiplier and access detailed installation instructions, visit:

<http://useful.com/products/dm>

Deployment considerations

This section examines various deployment considerations relating to multi-station systems. In general, multi-station computers are best deployed in environments where the shared system's CPU and memory usage are not expected to be intensive, such as in most fixed function or provisioned office systems where each user runs a limited number of concurrent applications.

For general office deployments where the primary computer use consists of an e-mail client, several Web browser and office suite windows, and perhaps a data entry or terminal emulation application, the processing, storage, and networking capabilities of even modestly equipped desktop computers far exceed the demands of individual users.

Storage, printing, and external service considerations

The basic requirements for deploying multi-station desktop Linux systems differ little from those deploying standard multi-PC environments. Few or no adjustments to file, print, and other network services are necessary to accommodate access from multi-station computers. In both single and multi-user variations of the desktop Linux environment, the use of local storage for persistent data such as user documents is discouraged in favor of centralized, managed storage.

Remote file systems

Users of multi-station Linux desktops require access to remote file systems in the same manner that users of single-user Linux desktops do.

Access to remote file systems is implemented in the underlying Linux operating system, which is multi-user aware, rather than in X or the desktop environment, posing few additional challenges for multi-station Linux desktop deployments. In the majority of cases where remote file systems are accessed on a per user basis (rather than per workstation), those file systems can be securely and privately mounted by individual users on a multi-station Linux system in the same fashion.

Because most storage solutions deployed according to best practice are already multi-user aware, and because such awareness is not a characteristic of the physical layers of connectivity, adjustments to file and print management are typically not required to accommodate access from multi-station computers.

Multi-station Linux desktops use the same standard authentication or directory services (such as NIS, LDAP, or local authentication) as those provided for single station Linux desktops.

Network services

Most network-enabled services are unaffected by consumption from clients using multi-station computers. For example, in a UNIX-type network environment, sockets and client/server connections are all multi-user aware and many services can be accessed by clients behind a NAT router or firewall. Most services do not need to be reconfigured to accept multiple client connections from a single IP address, in either single or multi-station configurations.

Note: All users on a particular multi-station system share a single IP address in the default configuration, although integrated solutions such as Desktop Server and DiscoverStation can optionally bind unique IP addresses to individual users. Be sure to review any applications that depend on unique IP addresses for identification purposes for possible issues.

Network, electrical, and physical infrastructure considerations

Multi-station computers substantially reduce network, electrical, and physical infrastructure requirements, resulting in substantial management and cost savings in all areas.

Network infrastructure requirements

Deploying multi-station Linux provides the advantage of simpler network configurations with fewer IP addresses and network devices to manage.

In most multi-station deployment scenarios, a single standard 100 Mbps Ethernet connection to the multi-station computer provides sufficient bandwidth for all users to access network-based data. Although each individual multi-station computer can conceivably require as much bandwidth as individual workstations do for the same number of users, it is frequently the case that users are fully productive on less than 1 Mbps of bandwidth, as evidenced by the success of telecommuting over home broadband connections.

We recommend gigabit network connections to the multi-station systems if many users are expected to simultaneously access bandwidth-intensive resources such as streaming video or large file access. IntelliStation and most other workstation-class PCs include an integrated gigabit network interface, providing sufficient bandwidth for each user on the shared system without noticeable network lag.

Electrical requirements

Traditional single-station workstations such as IntelliStation require approximately 550 W per user, with the computer consuming 450-500 W and the LCD monitor consuming 50 W through a total of two electrical outlets. A five station configuration reduces this requirement to under 150 W per user, eliminating 400 W of consumption and one electrical outlet per user sharing the multi-station computer. For new deployments, the reduced power consumption of multi-station systems can lead to substantial savings in facility upgrade costs and reduced power consumption.

Consult your workstation and display manuals for detailed power requirements.

Office layout

Multi-station configurations are best suited to office environments where users work in close proximity, for example, open plan offices where cables and stations can be easily installed. Multi-station configurations are less suited for enclosed office deployments because the cabling and distance requirements become more technically challenging. Extension cables and KVM over copper solutions can help to enable effective multi-station deployments in otherwise restrictive office layouts. Multi-user, multi-display configurations are spatially suited to almost any office or production environment. The small footprint of modern LCD monitors allows at least two moderately sized monitors to occupy the physical desktop space of a traditional CRT monitor.

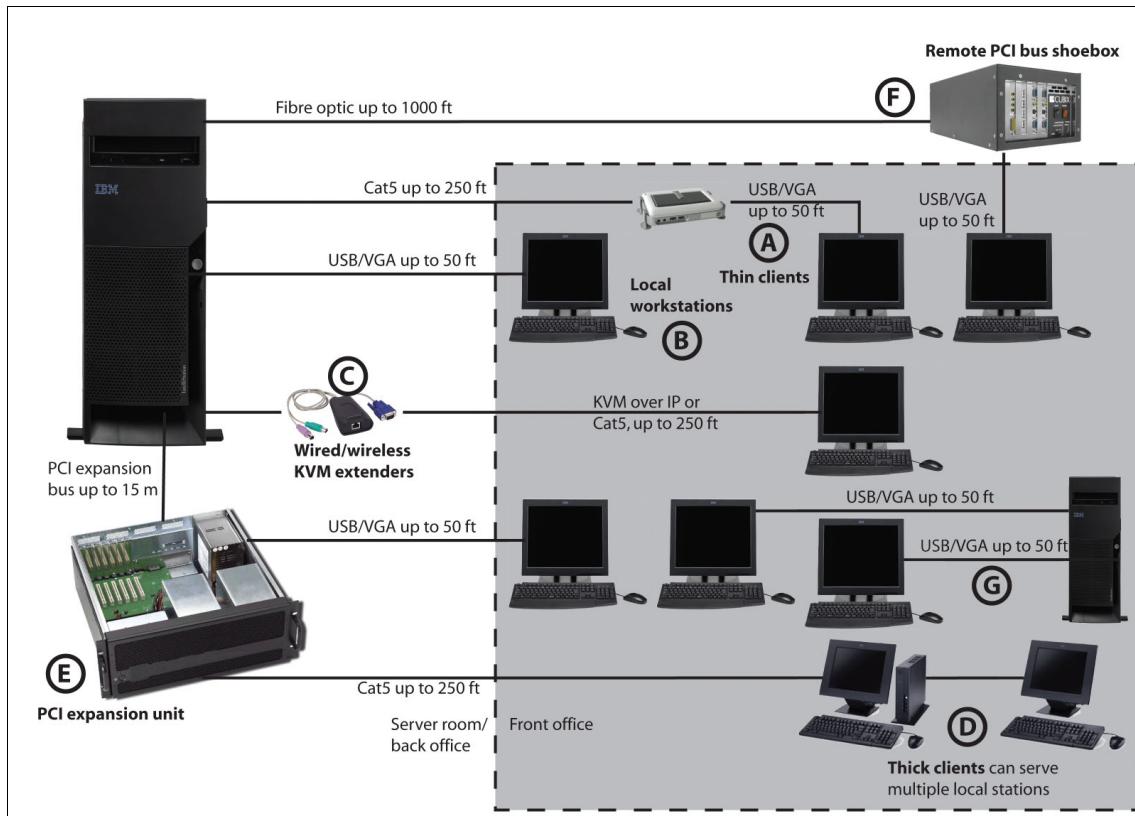


Figure D-2 Multi-station hardware topology options based on a high-end IntelliStation multi-station platform

Figure D-2 illustrates some of the Multi-station topology options discussed so far. Key components in the figure are described as follows:

- ▶ A: Managed multi-station disk images support thin clients alongside regular multi-station desktops.
- ▶ B: Local workstations connect to the IntelliStation using standard USB/VGA cables.
- ▶ C: KVM options allow mice, keyboards, and monitors to be located up to 250 feet away from the IntelliStation.
- ▶ D: Thick clients with minimal CPU, memory, and PCI expansion slots can serve desktops to local stations, while application processing occurs on the central IntelliStation computer.
- ▶ E: PCI expansion bus units allow stations to be deployed into user environments, while the IntelliStation is managed in a secure environment.

- ▶ F: PCI expansion bus units are also available with fiber optic interconnects, allowing workstations to be deployed up to 1000 feet away from the IntelliStation.
- ▶ G: IntelliStations can support multiple local clients, while sharing applications and data with other systems.

Complementary technologies

A variety of complementary technologies can extend the flexibility of multi-station deployments. This section discusses several of the key technologies that can be used to logically complement and physically extend a multi-station computing platform.

Adding more stations using PCI bus expansion

If the number of available PCI or PCI-Express expansion slots in the shared system is a limiting factor, PCI bus expansion hardware allows additional video cards and peripheral devices to directly link to the shared system. PCI expansion units with up to 13 additional PCI slots can be daisy-chained or fanned out from the single shared system to facilitate as many extra PCI cards as the computer's PCI bus bandwidth permits. PCI expansion units can also be used in conjunction with PCI extension technologies over fiber-optic cable, effectively enabling an unlimited number of PCI expansion slots at almost any distance from the shared server system, using a single fiber-optic connection.

Connecting remote stations

KVM over CAT5 cable and PCI extension technologies can effectively support connection of client workstations to the multi-station platform at distances of up to 250 feet using copper¹, or up to 1,000 feet or more using fiber². In a multi-station scenario, one or more workstations of a multi-station computer system can be deployed to hostile or remote environments where deploying an entire computer can be impractical or even impossible. These units can typically be combined and linked in a variety of ways to accommodate virtually any deployment scenario.

Remote desktop clients and services

Multi-station technologies combined with remote desktop client software enables a single multi-station platform to support multiple connected clients that can run virtually any kind of operating system interface or application available on remote hosting servers. For example, remote Linux desktop services based on RDP,

¹ The hardware KVM units combine (multiplex) the electrical signals carrying keyboard, mouse, and video data into a single data stream.

² PCI extension over fiber-optic cabling allows the complete PCI bus subsystem to be extended to as many individual PCI cards as the computer's bus bandwidth can support.

XDMCP, VNC, NX, and so forth can be layered into a multi-station client computing design just as easily as in a traditional thin client design.

Tip: As soon as you move above hardware level considerations, multi-station and thin client architectures can become complimentary instead of competing design strategies.

See 7.2, “Remoting tools” on page 154 for more details.

Additional system management considerations

Software maintenance, system management tools, security, and privacy are common concerns among all managed desktop environments; however, a few basic additional considerations pertaining to multi-station Linux should be noted.

Software updates

Software packages and updates provided by the vendor of the installed Linux operating system typically do not account for multi-station systems. For most multi-station deployments, this fact can be both advantageous and annoying.

Updating most user application and system component packages does not affect an existing multi-station installation. The exceptions are software packages that directly affect X, the login managers, and configurations for those subsystems.

The following kinds of updates can affect Desktop Multiplier settings:

- ▶ Updates that change GDM settings might prevent graphical login from more than one station.
- ▶ Updates that change the runlevel of the system might cause Desktop Multiplier to exit prematurely (Desktop Multiplier defaults to runlevel 5).

However, to compensate for these cases, on the next system reboot, Desktop Multiplier will attempt to automatically restore the GDM configuration file should it be overwritten by an OS update, software package, or service pack.

The Desktop Multiplier configuration can be reset to defaults with the following command: `rm -f /etc/X11/useful.Mxorg.conf`. After doing this, a software wizard creates a new configuration file on the next reboot. In situations of extreme modification due to vendor updates, reinstalling Desktop Multiplier might be necessary.

Also, USB keyboard assignments can be reset by running the following command: `/opt/useful/bin/usbinput-reconf.sh`

Management tools

Multi-monitor and multi-station system setups can at times pose additional management challenges. Even though the one-to-one relationship between users and their GUIs is preserved, some system management software might not be aware that multiple monitors or other interface devices are in use.

Tools, such as Novell ZENworks system management software or OCM Suite from OpenCountry, typically work seamlessly in multi-station setups, while vendor specific tools might or might not. Be sure to thoroughly test management tools in a variety of multi-user configurations prior to deployment.

Software and support

Multi-station desktop Linux can reduce the total number of software and support subscriptions required in many deployment scenarios by reducing the total number of CPUs deployed. Depending on the particular software licensing agreements in force, licensing and support subscription costs can be reduced substantially. Deploying multi-station systems also substantially lowers the total system support load because there could be significantly fewer desktop systems to manage.

System shutdown

System shutdown and physical access to controls on multi-station computers are important considerations where multiple individuals share the same computer. Restarting a shared system without notice could result in loss of work or data for multiple users at once. Useful's Desktop Multiplier removes the shutdown option from the default greeter theme on Fedora Core. Best practices suggest similar modifications should be made to all multi-station deployments to avoid the accidental loss of work. As an example, Useful's modified greeter theme for Fedore Core is available from:

<http://www.useful.com/no-shutdown-gdm-greeter>

Multi-language support

Desktop Multiplier supports international keyboard layouts and allows each user to have different keyboard settings as needed. However, GDM and KDM currently do not allow users to change keyboard layouts³. As a result, users cannot switch keyboard layouts until after they log in. The result is that users

might be unable to type their username and password to log in if the keyboard layout is incorrect.

If a multi-station computer is serving users who need to type using multiple languages or use different keyboard layouts, the system and the greeter should be started using a common language layout, which can then be changed by the individual users after logging in. Desktop Multiplier software supports full internationalization of its configuration tools with translations available in most commonly used languages. Support for additional languages are typically available within several business days of a request for supporting a new language.

Security considerations

Software security on multi-station Linux poses few significant challenges beyond those of single-user Linux workstation deployments. The key security difference between a single-user Linux workstation and multi-station Linux is that multiple users share access to the same computer system; therefore, its storage, processes, programs, and other hardware are also shared. Any system compromise or corruption during one user's activity can potentially affect all other users.

Sharing systems is not explicitly a problem if the users never modify their workstation environment, for example, by installing browser plug-ins and applications, or by adding or compiling shared libraries. But because multiple users are able to simultaneously run potentially untrusted applications on the same computer, a physical or electronic compromise of one user's environment could result in the compromise of all local user environments.

In order to minimize the effect of user configuration changes, we recommend that users of multi-station systems never are given local root or administrator level access, and that users are made aware of how to install software in their own private directories without adversely affecting the shared multi-station computer environment. In this manner, no user can have access to another user's temporary files or running processes, and no user can modify the shared environment.

Privacy considerations

As discussed above, sharing a computer could result in the inappropriate, inadvertent, or deliberate disclosure of confidential information. The potential for sharing data and the sometimes complicated access controls required to prevent

³ The default keyboard layout is set in the X.org configuration file and is generally guessed from the default system language.

unintentional data sharing could pose problems for compliance with regulatory or internal policies.

Depending on specific corporate policies, it might or might not be a problem if different employees within the same security context share a computer with other employees who have legitimate access to the same information. Access to the same computer by employees from different security contexts, however, should be actively prevented by only sharing multi-station computers among employees of the same function. In many cases, existing policies that apply to the handling of confidential information about shared or multi-user access mainframe computers or file servers can map identically or be easily adapted to multi-station environments.

Case study one: General office desktops for a 25-user office

This case study shows a multi-station solution for adding employees at a branch.

Requirement

A 15-employee branch office of an IT company needs to expand its open plan office to accommodate ten additional employees. The office also has an executive suite with four workstations.

The office is located in an older building with aging electrical infrastructure with relatively few wall outlets, low mains amperage, and most network cables must be run through the plenum space due to the poor quality of installed copper lines. Local construction bylaws concerning older buildings prevent the addition of new electrical cabling without renovating the electrical infrastructure of the entire building.

Each employee's workstation either ran Windows or a Red Hat or Fedora Linux distribution at various patch levels and was maintained individually as needed. User files were maintained locally with no effective security measures, centralized data storage, or backup plan. Several Windows workstations were required for print document design and production, accounting functions, and for access to occasionally used specialized Windows applications.

Solution Design

The office deployed six multi-station systems based on Fedora Core 4 and Desktop Multiplier, each with four or six video heads. Four of the quad-user systems were retrofitted from the existing inventory of workstations by adding video cards and memory, while the two six-user systems required new computers based on AMD's Athlon 64 processor. All systems contained enough memory to provide between 256 and 512 MB of RAM per attached client workstation.

In the new deployment, each multi-user system serves four to six users. One of the systems was fitted with four dual-head video cards to accommodate four dual-display users. A central server running Linux was added to provide authentication services and remotely mounted home directories for each user.

The four users in the executive suite were provided with client workstations all connected to another multi-station host system.

Access to optical drives was preserved by positioning multi-station systems (each with a single optical drive) close to the users who anticipated using them most frequently. Deploying external USB optical drives (for approximately \$30 more than the cost of an additional internal drive) is an option if concurrent and frequent access become necessary in the future.

Minimal changes to the office layout were required to accommodate the new configuration. The most drastic changes were the removal of individual workstations from under most employees' desks, and the deployment of storage, backup, and intranet servers based on hardware from the former workstation computers. No additional electrical or network cabling was required to add workstations for the ten new users. An important consideration because installing new cabling and electrical outlets is costly.

With the savings from reduced computer management costs, the office was able to expand the deployment beyond the initial plan to have one workstation for each employee, and the office outfitted every available desk with a workstation. The extra workstations increased flexibility in work patterns and improved collaboration among employees. In addition, an RDP server was set up to provide access to key Windows applications from anywhere in the office using the Open Source rdesktop software at:

<http://www.rdesktop.org/>

The move to a consistent software platform and patch level reduced demands on the IT staff. The hardware-related savings allowed the office to increase their hardware refresh cycle from four years to two years, ensuring all users have the latest high performance computer hardware. As a result, single image desktops eased management, central storage, and backup-related overhead. The

deployment also freed up ports on the existing 24-port managed switch for future network growth. The workstation redundancy built into the new setup also allows anyone to work on any other station for collaboration, or to minimize employee down time in case of hardware failure.

Case study two: Transactional desktops: Public computers for a city library

A multi-branch city library experienced a period of significant growth due to population increases in their area. Library branches were added and modernized to increase services with a focus on electronic access. The highest growth rate was in public computer use. Its eight branches saw a 400% increase in computer hours logged between 1998 and 2003.

Requirement

Since it began deploying public access computers, the library has prided itself on offering its patrons an excellent service. Unfortunately, tight budgets coupled with high hardware, software, and support-related costs were forcing tough choices on the library. If the branches wanted to meet the immediate demand for computer access for all their patrons, not all of the computers could have the latest software packages. Even free software updates (such as the latest version of Macromedia Flash) were often not up-to-date because staff time was always in limited supply. Many stations lacked floppy drives, all stations lacked CD burning capabilities, and not all stations could run word processors. For years, the library was forced to pick and choose from a limited set of options. They simply could not provide everything patrons wanted on every computer. This created a heterogeneous hardware and software environment that was difficult to maintain and support.

Helping patrons find the correct station to print a resume or download attachments to disk also placed an unwanted burden on the reference desk staff. Supporting the wide array of hardware and software combinations on 100 public access computers across eight branches became a headache for the IT staff. In addition, there were never enough machines to go around and there were often long lineups for computers, particularly during peak hours.

Solution Design

In 2003, the library's Manager of Systems and Technical Services decided to deploy public access computing solutions for their branches based on Useful's Desktop Multiplier technology. The library's multi-station solution was built by

augmenting standard PCs with multiple dual head video cards with the target of supporting four to eight users per computer.

Replacing a large number of individual workstations with more tamper-resistant multi-station Linux systems drastically lowered support, licensing, and power usage costs for the library. The multi-station desktop approach also enabled the library to implement a consistent hardware and software platform at all branches, which reduced the burden on IT staff. Further savings were achieved through lowered hardware and software failure rates due to the reduction in the total number of systems they needed to support.

Additional multi-station case studies

You can read additional case studies and customer references for multi-station Linux deployments at:

<http://useful.com/customers/case-studies>

Understanding how multi-station computing works: Exploiting the flexibility of X Window System

The X Window System (also known as X11 or X) is a windowing system for computer displays, which provides the framework for the graphical user interface found in most Linux distributions. X is responsible for drawing and moving windows on the display and interacting with input devices such as a mouse and keyboard. X does not specify the appearance or behavior of the user interface. Xorg, the current reference implementation of X, is the most commonly bundled and distributed version, although implementations such as XFree86 are also popular. For more information, see:

http://en.wikipedia.org/wiki/X_Window_System
<http://wiki.x.org/wiki/>

The design of X

The design of X is based on a client/server model, in which the X server communicates with client programs. The X server accepts requests to display graphical output (windows) from the window manager and returns user input (keyboard, mouse, and so forth) to the window manager. The window manager running above X (such as Enlightenment or twm) controls the placement and display of application windows. On top of the window manager, GUI widget toolkits (such as Qt and GTK) and desktop environments (such as KDE and Gnome) provide the windows with their specific appearance and visual styles.

Applications, such as Web browsers, text editors, and so forth, request that the window manager provide windows with specific user controls (check boxes, buttons, scroll bars, and so forth) in specific positions. The window manager, in conjunction with any GUI widget toolkits in use, then composes the requested window for the user.

The X server is capable of keeping track of multiple video cards, monitors, and input devices. Therefore, an X server is capable of logically interfacing with a large array of displays. In a multi-display configuration, X can be configured to present the displays as individual displays or as one large display, or as multiple logical groupings of one or more displays. See Figure D-3.

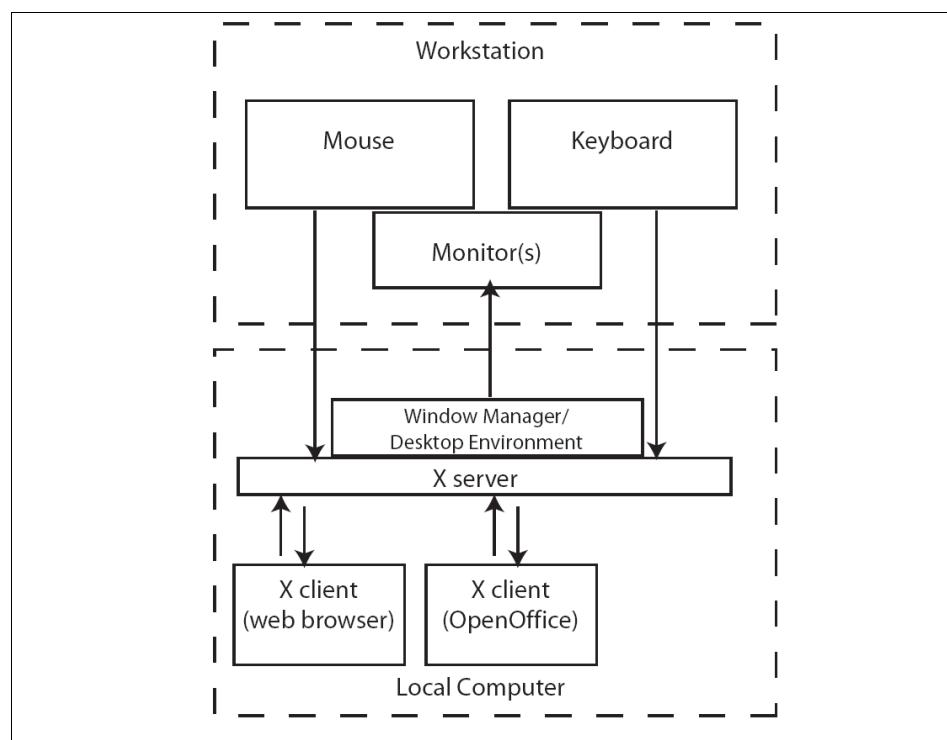


Figure D-3 Standard single-user X server

Figure D-3 shows a typical single-station X implementation that uses a single X server to handle a single keyboard/mouse input and outputs a single desktop to one or more monitors.

See Figure D-4 on page 311.

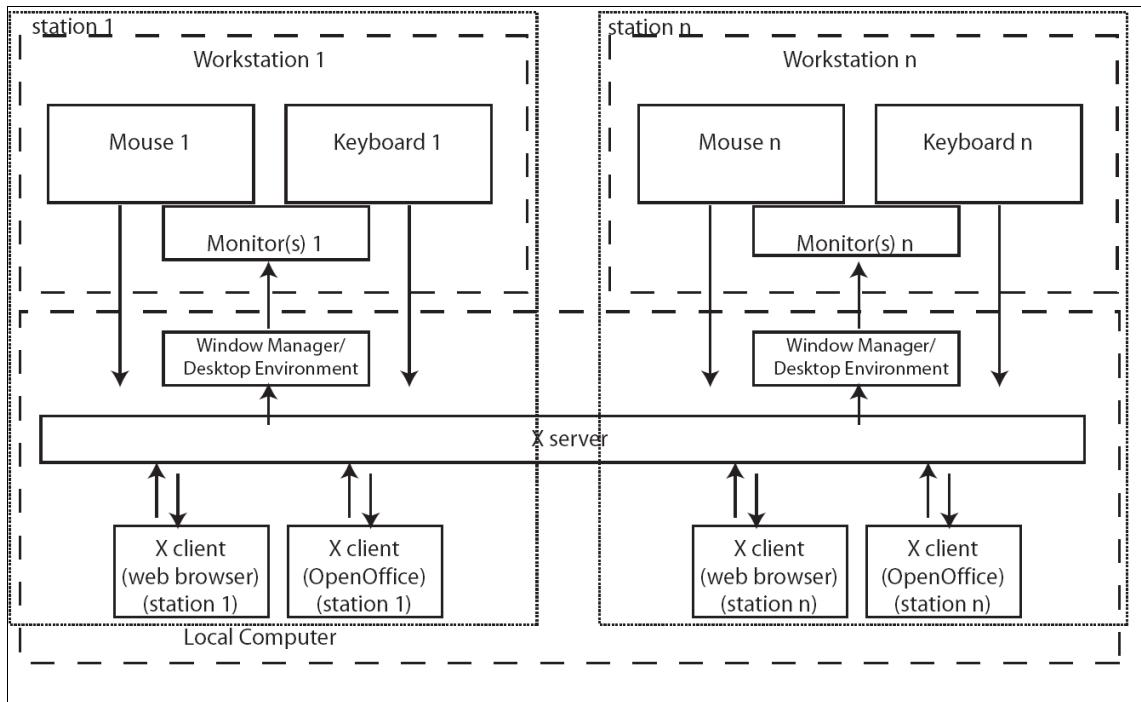


Figure D-4 Desktop Multiplier using one X server for all client workstations

Figure D-4 shows a multi-station X implementation that uses a single X server to handle multiple sets of keyboard/mouse inputs and outputs one desktop for each user to one or more associated monitors. This is how Desktop Multiplier operates.

And Figure D-5 on page 312 shows a multi-station X implementation in which one X server handles each set of keyboard/mouse inputs and outputs one desktop for each user to one or more associated monitors. This is how “Ruby”, also known as “Backstreet Ruby”, operates. For more information, see:

http://www.tldp.org/HOWTO/XFree-Local-multi-user-HOWTO/about_bruby.html

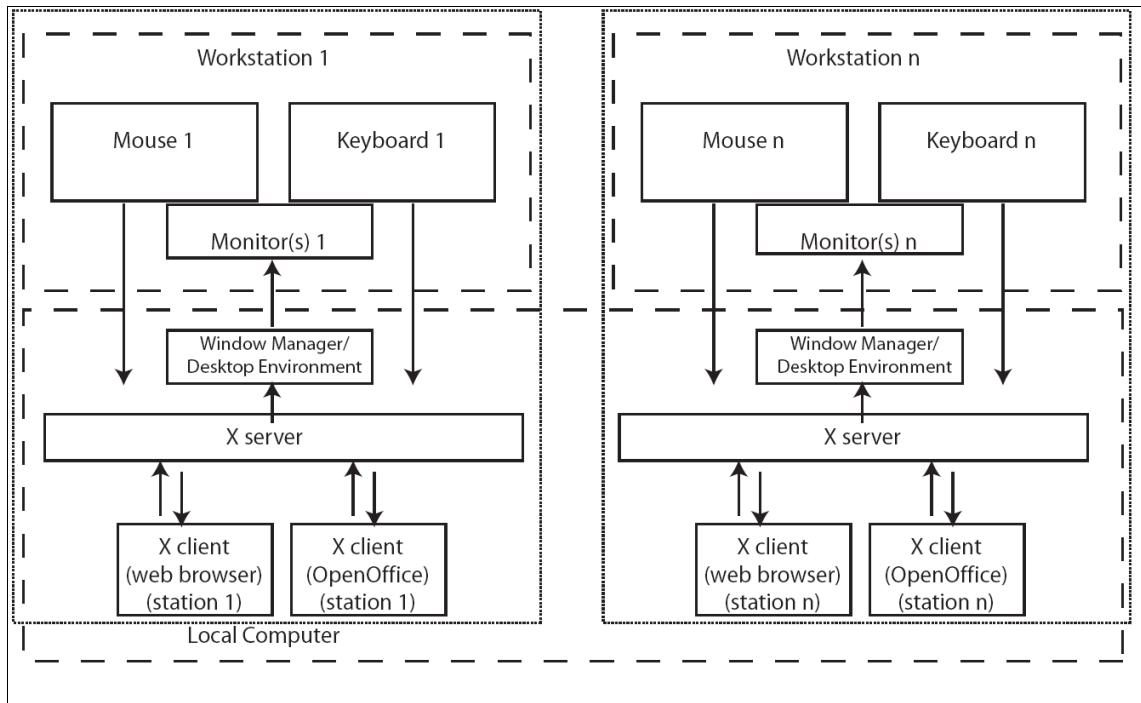


Figure D-5 Other multi-station approaches with separate X servers for each workstation

In a standard single user environment, the user runs a single instance of X on top of which rests the remainder of the GUI. In a multi-station environment, a single instance of X can be set up to handle multiple users (Desktop Multiplier) or multiple instances of X can be run (as in the case of Ruby).



E

Client personalization

This appendix compares the methods for storing client-side personalization data between Windows and Linux. Because this book focuses on migrating clients from Windows to Linux, migration of client-side personalization data could also be an important requirement for the planning and implementation phases of a project.

Microsoft Windows client personalization

The primary container for personalization data in Windows is called the user's *profile*. In Windows 2000 and Windows XP, profile data for the user "username" is typically stored in:

`c:\documents and settings\username`

In Windows NT, profile data for the user "username" is typically stored in:

`c:\winnt\profiles\username`

Windows uses these locations to store various kinds of personalization data. For example:

- ▶ User documents
- ▶ Application data and settings
- ▶ Temporary Internet files
- ▶ Internet favorites

The profile directory name uses the user ID, but can also have a more detailed distinction, because sometimes the computer name or the name of the domain is appended. This happens if a user exists already and a new one is created with the same name but integrated in the domain. So one username can exist several times on a machine, connected with the domain name or the local machine name or other combinations. The consequence is that Windows has to distinguish between user `username.localcomputername` and `username.localdomainname`.

Some settings are stored in subfolders of the user's profile. For instance, many settings are stored in a folder called "Application Data." However, the majority of settings are stored in the registry. The Windows registry contains two main "hives". The "Local Machine" hive contains system-wide settings and hardware information. The "Current User" hive contains settings that are specific to a certain user. The registry has its own permissions layer, which is similar to file system permissions. The registry can only be accessed through APIs provided by Microsoft, because the binary file representation is too complex to manually edit. Because many settings are stored in the registry, this can lead to a single point of failure; a corrupted registry can render an entire profile unusable.

Linux client personalization

Considering the modular structure of Linux, it becomes clear that profiles must differ substantially from the Windows profiles. Because there is no registry in Linux, all settings must be saved in another way. System-wide settings in Linux are normally configured using readable text files in the `/etc` directory. These files frequently have extensions such as `.conf` or `.profile`.

The file /etc/profile sets the system-wide environment and startup programs per user login. These system-wide defaults can be extended by profiles in the local or home directory of the user. The local .bash_profile (Bourne Again SHell) is one example on Red Hat Enterprise Linux; the .profile file on SUSE Linux is another example.

Structurally, Linux stores user profile data in a method somewhat similar to the way that Windows does. All individual user settings are stored inside the user's home directory, which is sometimes referenced as \$HOME. The home directory for the user "username" is usually found at:

/home/username

Linux does not implement a registry the way Windows does. Instead, user profile information is stored in configuration files within the individual user home directories. These files are usually created by specific applications that need to store user personalization data required by that application.

One advantage of these files is that they are often plain text formatted files, and thus, human readable. Another is that the content can be automatically manipulated by scripts if you want. Also, because each application stores data in separate files, a corrupted file only affects that application. Because these files are standard files, the standard file permissions apply. By modifying permissions on these files, it is possible to prevent unwanted changes by the user. This could have the effect of "locking down" certain aspects of the user's login environment.

Most applications create a hidden folder, beginning with a dot or period (.), in which configuration files for those applications are created and maintained. Frequently, these files are grouped together with similar applications, such as all Mozilla-based applications storing settings in \$HOME/.mozilla.

Desktop personalization: KDE Desktop

In KDE, desktop personalization is based on information found in a standard directory structure within the user's home directory. The contents of the desktop, including icons and other data, is usually found in:

/home/username/Desktop

Application data settings, such as e-mail, bookmarks, and other preferences, are usually stored in:

/home/username/.kde/share/config

When KDE looks up a setting, it first checks settings found in the user's home directory. If the setting is not found there, then it looks in the global configuration directory. Red Hat typically places the global configuration files in the

/usr/share/config directory. Other distributions might place these files in different locations. Because users start out with no customized settings in their home directory, any changes that are made to the global settings files are inherited by all new users created after making the changes.

KDE uses plain text files for all its configuration options. This makes it easy to configure KDE using scripts and editors. Three of the main files used to configure the desktop are:

- ▶ kdeglobals
- ▶ kdesktoprc
- ▶ kickerrc

The most recent versions of KDE ship with an application called KConfigEditor. KConfigEditor allows you to edit these files in a GUI application. KConfigEditor can also be used to change some settings for GNOME applications as well. An example of the KConfigEditor interface is shown in Figure E-1.

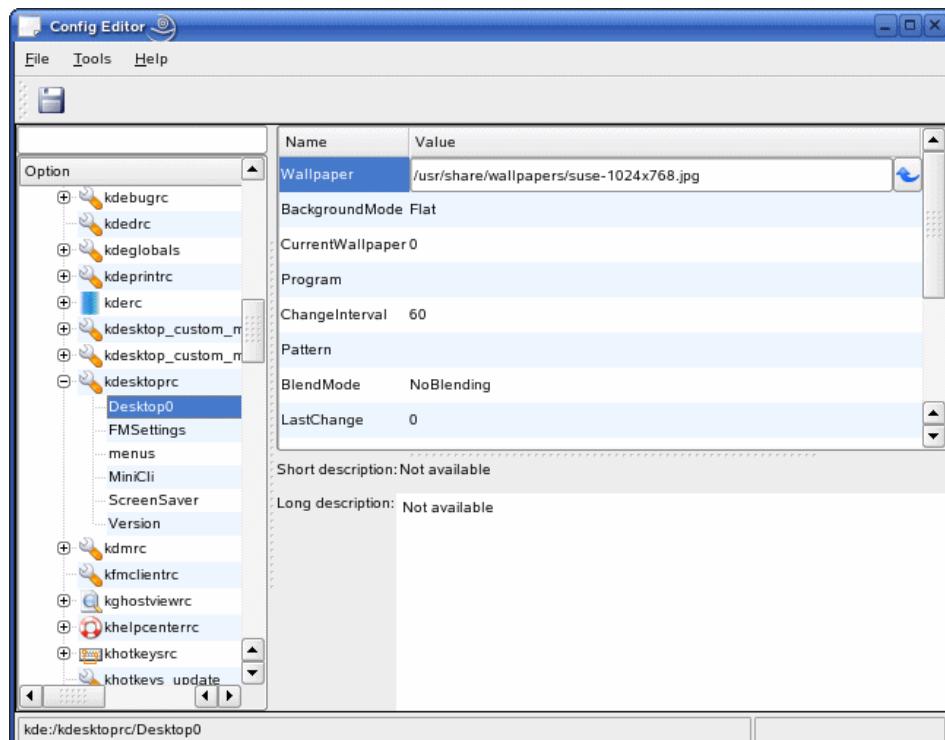


Figure E-1 KConfigEditor editing kdesktoprc

Desktop personalization: GNOME Desktop

As with the KDE desktop, desktop personalization files in GNOME are stored in a directory structure within the user's home directory. Just like KDE, modern versions of GNOME store desktop icons and other data in:

/home/username/Desktop

The task of making general configuration changes to the GNOME desktop is extremely different than it is for KDE. While KDE uses simple text files, the GNOME desktop uses a database called GConf to manage all of its settings. In order to make settings available from different applications and also from tools within the GNOME Desktop, there is a daemon running that notifies all involved applications when a value has changed and that manages all access between applications and the configuration sources. This daemon is called gconfd-2 and runs in one instance for each user.

Two tools are provided to edit GConf settings directly: gconf-editor and gconfcontrol-2. gconf-editor is a graphical tool, which lets you browse and change the settings stored in GConf. An example of using gconf-editor to browse to the **desktop/gnome/background** key is shown in Figure E-2 on page 318. Many values also include a description of what setting it controls, and what data is relevant.

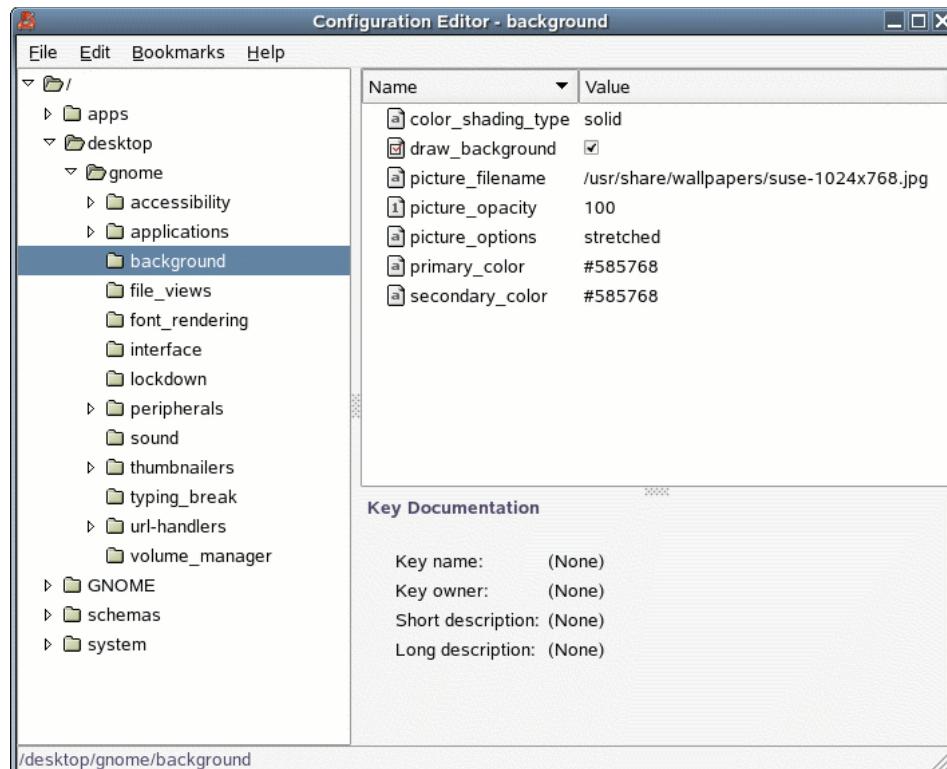


Figure E-2 Editing gconf entries with the graphical gconf-editor

gconftool-2 is a command line application, which provides the access to the same information and is quite useful for scripting purposes. The number of configuration options that are available to the gconftool-2 tool are vast and too numerous to list in this book. To display the current options that are set for the GNOME desktop, enter the following command:

```
gconftool-2 -R /desktop
```

Although we do not cover all options, we show how we customize our desktop and explain each of the commands that are used. To list all the settings associated with the desktop background, enter the following command:

```
gconftool-2 -R /desktop/gnome/background
```

This displays the following default values:

```
color_shading_type = solid
secondary_color = #585768
primary_color = #585768
picutre_filename=/usr/share/wallpapers/suse-1024x768.jpg
```

```
picture_options = stretched  
picture_opacity = 100  
draw_background = true
```

The following code changes the wallpaper to a corporate logo:

```
gconftool-2 --type string --set /desktop/gnome/background/picture_filename  
/usr/share/backgrounds/images/corporate_logo.jpg
```

Immediately after running the command, the wallpaper should change to the newly specified path.

For further information about how to make mandatory changes to GConf, see 7.1.2, “GNOME lockdown options” on page 151.



F

Desktop automation and scripting

In this appendix, we present details about several scripting and automation capabilities of Linux applications. These capabilities allow users and administrators to fully control their environment and automate many desktop management and configuration tasks.

Scripting languages

Open source scripting languages for Linux such as Perl, Python, Ruby, ECMAScript, or Tcl, have been ported to many diverse operating systems, including Windows and Mac OS X. The typical Windows scripting language, Visual Basic, and the related project, Visual Basic for Applications (VBA), are not directly available for Linux; although, there are similar projects available either through commercial offerings (such as REALbasic, which is discussed in Appendix G, “Application porting” on page 329) or through free implementations (such as StarBasic in OpenOffice.org¹):

<http://www.thefreecountry.com/compilers/basic.shtml>
http://documentation.openoffice.org/HOW_TO/various_topics/VbaStarBasicXref.pdf

All of these Basic dialects are certainly relevant from a compatibility point of view (for example, for Excel macro migration), but they do not play a central role in the foundation of the GNOME or KDE desktop environments, because there are so many other natural choices there.

Shell scripting

The traditional way of scripting on Linux (and UNIX) machines involves shell scripts. Shell dialects include bash, ksh, and tcsh. Shell scripts are the Linux equivalent of batch files on Windows, except that they are significantly more powerful. Shell scripts are frequently used for startup routines (such as those found in /etc/init.d) or as wrappers around more complex programs (such as Mozilla or OpenOffice.org). More information about programming with shell scripts can be found at:

- ▶ Linux Shell Scripting Tutorial: <http://www.freeos.com/guides/lstt>
- ▶ Advanced Bash-Scripting Guide: <http://www.tldp.org/LDP/abs/html/>

Perl

Perl (Practical Extraction and Report Language) is a flexible and powerful scripting language. Perl was originally developed for text manipulation, and now it can be used for much more. An impressive collection of third-party modules are available at the Comprehensive Perl Archive Network² (CPAN). While enthusiasts say that Perl’s philosophy of “TMTOWTDI” (“There’s More Than One Way To Do It”) makes Perl easy to use to program, others can make a legitimate claim that the same philosophy makes Perl code very difficult to read and maintain.

¹ http://en.wikipedia.org/wiki/StarOffice_Basic

² <http://www.cpan.org>

Python

Python is a modern object-aware scripting language that has remarkable power while still maintaining clear syntax. There are built-in libraries, which provide abstractions for many platform specific features (such as file access and threading), and many built-in or third-party open source libraries for graphical applications (such as PyGTK, PyQt, and wxPython, and even bindings for Mac OS X and Microsoft Foundation Classes (MFC)). New modules can be written in either Python, C, or C++. While an entire application can be written in Python, it is also frequently used as an extension language for applications that have a built-in plug-in system.

Red Hat uses Python extensively in the installation process and at many other places. Tools, such as kudzu (which is also used for hardware recognition in the Debian-based Knoppix distribution), are available as Python modules, too. While using **kudzu --probe** is straightforward, the Python wrapper allows system administrators to write their own hardware recognition scripts in a much more elegant fashion, as seen in Example F-1.

Example F-1 Hardware recognition with kudzu and Python

```
#!/usr/bin/env python

import kudzu

print kudzu.probe(kudzu.CLASS_SOCKET, kudzu.BUS_PCI, kudzu.PROBE_ALL)
print kudzu.probe(kudzu.CLASS_NETWORK,kudzu.BUS_PCI, kudzu.PROBE_ALL)

devices = kudzu.probe(kudzu.CLASS_UNSPEC, kudzu.BUS_UNSPEC, kudzu.PROBE_ALL)
for dev in devices: print dev
```

Embedded Scripting Languages

Most Linux scripting languages have wrappers for the GTK+, Qt, and wxWidgets libraries, and even the whole KDE/GNOME APIs. In fact, it is possible to write simple desktop applications or panel applets in a scripting language in order to allow for rapid application development. Several applications also have embedded languages, which allow for user-customized actions. Such customization is often useful to tailor these applications for your exact business needs. The language used for these embedded capabilities often varies by application. For example, Perl and Python are used in Gnumeric and The GIMP,

JavaScript, and XUL are used in Mozilla and Mozilla-based projects (such as Firefox and Thunderbird), and Star Basic is used in OpenOffice.org.

Qt Script for Applications

Trolltech decided to give ECMAScript (the standardized version of JavaScript) a dominant role in Qt by releasing the Qt Script for Applications toolkit (QSA), which is tightly integrated into their framework:

<http://www.trolltech.com/products/qsa/>

QSA plays the same role as VBA for Windows applications and is very easy to use for Qt programmers. QSA is released under a commercial license for Linux, Windows, and Mac platforms. In addition to that, QSA is also licensed under the GNU GPL for free software development.

KJSEmbed

An very similar approach is the KDE JavaScript Engine, KJSEmbed. KJSEmbed has bindings for Qt, KDE, and DCOP interprocess communication mechanisms. You can find KJSEmbed, which is licensed under the LGPL, and a lot of documentation at:

<http://xmelegance.org/kjsembed>

<http://www.sourcextreme.com/presentations/KJSEmbed/>

In Example F-2, you see how easy it is to write to a simple HTML browser with JavaScript using this toolkit.

Example: F-2 Basic HTML browser with KJSEmbedd

```
#!/usr/bin/env kjscmd

var url = 'http://www.kde.org/';
if (application.args.length)
    url = application.args[0];

var mw = new KMainWindow();
var box = new QHBox(mw);
mw.setCentralWidget(box);

var view = Factory.createROPart("text/html", box, "view");
view.connect(view.child(0),
             'openURLRequest(const KURL&,const KParts::URLArgs&)',
             'openURL(const KURL&)')
view.openURL(url);

mw.resize(650,500);
mw.show();

application.exec();
```

Kommander

The developers of the Quanta+ Web development framework introduced a new toolkit called Kommander, which you can use to design KDE Dialogs and widgets by using **kmdr-editor** (a program derived from Qt Designer) and executing them with **kmdr-executor**. Kommander uses XML-based GUI description files with embedded DCOP scripts, which can be written in any language able to speak the DCOP protocol (such as Bash, Python, Perl, Java, or JavaScript). Kommander allows you to write KDE dialogs by using the powerful DCOP mechanisms itself. It is already used at many places inside Quanta+ (DCOP is also used for the inter-component communication in the KDE PIM framework Kontact). Take a look at Figure F-1 to see how the integrated DCOP development with **kmdr-editor** looks and can be used for your own purposes. For more information:

<http://kommander.kdewebdev.org>
<http://quanta.kdewebdev.org>

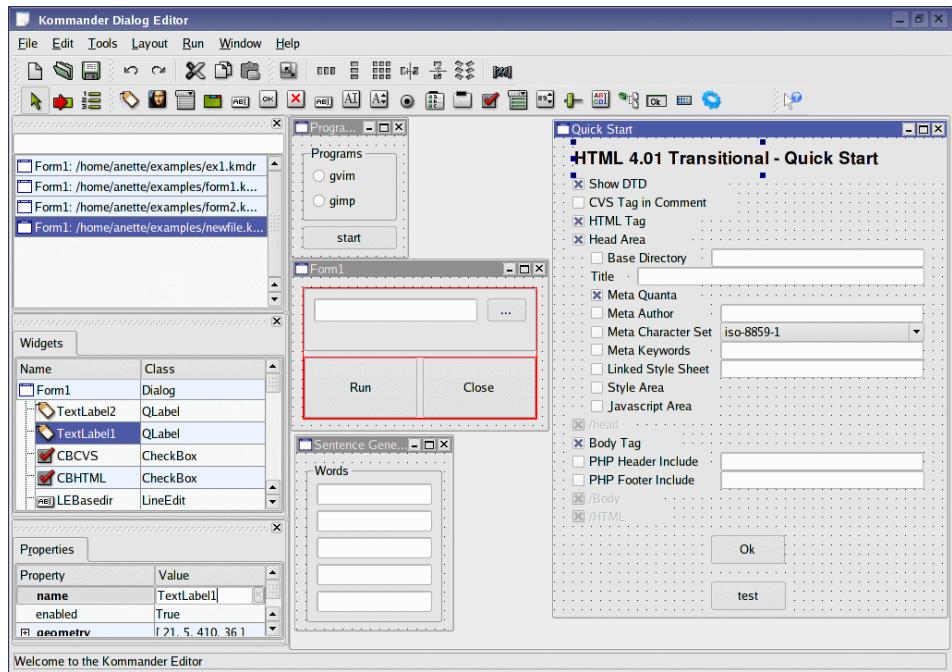


Figure F-1 Integrated DCOP development with Kommander

Desktop interprocess communication

Sometimes you want to control multiple applications from one authority. Or, perhaps you want to standardize on one programming language so that customizations do not need to be written in each of the various embedded languages. Many Linux-based applications support a form of interprocess communication, which allows you to control the application programmatically.

The GNOME developers originally used ORBit2 (a highly optimized CORBA implementation), but lately, they have focused on D-BUS for future development projects.

DCOP is the standard KDE Desktop Communication Protocol. You can use it for all kinds of KDE desktop automation and scripting purposes. It is based on the Inter Client Exchange (ICE) protocol and uses UNIX sockets instead of remote calls. It has its roots in a CORBA implementation, but it was completely redesigned for desktop needs because the KDE developers were not happy with the CORBA performance at that time. Let us take a closer look at DCOP, because this is a proven technology that is available today. You can also use DCOP for KDE applications running on the GNOME desktop.

DCOP

When you log in to your KDE session, a program called **kdeinit** is started (by **startkde**). It triggers other applications, such as **dcopserver** and the KDE daemon **kded**, which are responsible for system configuration management and caching (syscoca), interprocess communication, and a lot more. If you really want to understand how this process works and which environment variables and configuration files can influence it, you should read the *KDE for System Administrators Guide* available at:

<http://www.kde.org/areas/sysadmin/>

You can explore your desktop environment and make DCOP calls with the command-line tool **dcop**, its graphical counterpart **kdcop**, or with the DCOP language bindings using Perl, Python, Ruby, Java, or any other language supported through the **kdebindings** package. The **dcop** command syntax is as follows:

```
dcop [ application [object [function [arg1] [arg2] [arg3] ... ] ] ]
```

Desktop automation with DCOP

To begin with your desktop automation work, just call **dcop**, check the output, and go further down the pipeline. In Example F-3 on page 327, you can see how this works in practice. After listing all applications registered with the DCOP server (some of which include process IDs, such as konsole's process ID 366), we open a new Konsole session and display the first session name afterwards. Next, we list our e-mail accounts, check e-mail, compact all e-mail folders, and open a new KMail composer window with a few predefined values.

Example: F-3 DCOP in action

```
% dcop
konsole-366
kmail
...
% dcop konsole-366 konsole newSession
% dcop konsole-366 session-1 sessionName
% dcop kmail KMailIface accounts

% dcop kmail default checkMail
% dcop kmail default compactAllFolders
% dcop kmail default openComposer info@kde.org "" "" "Subject" "DCOP Rocks" 0
```

In Example F-4, we open our favorite URL in Konqueror and generate another browser window with two preloaded tabs. The next line switches Konqueror to full screen mode, which is useful for presentations or unattended Kiosk mode.

Finally, we switch to desktop 2 by using the published interface of the KDE window manager **kwin**.

Example: F-4 More DCOP magic

```
% dcop  
konqueror-21209  
kwin  
...  
% dcop konqueror-21209 konqueror-mainwindow#1 openURL http://www.ibm.com/linux  
% dcop konqueror-21209 default openBrowserWindow http://www.kde.org  
% dcop konqueror-21692 konqueror-mainwindow#2 newTab http://www.linux.com  
  
% dcop `dcop konqueror-21209 konqueror-mainwindow#1 action fullscreen` activate  
  
% dcop kwin default setCurrentDesktop 2
```

DCOP is an extremely powerful mechanism and can even be used with XML Remote Procedure Calls (through the XML-RPC to DCOP bridge). By using standard UNIX security measures for the ICE communication sockets, users should not be able to attack DCOP sessions from other users.



G

Application porting

When planning for a Linux client migration, you might discover early on in the assessment process that you have custom-developed applications that do not run natively on a Linux-based client (any Microsoft Visual Basic application fits into this category).

This appendix provides information and links for learning more about the application porting process. This topic alone could easily fill an entire redbook. We introduce the topic here for the sake of completeness (many Windows-to-Linux client migration projects have application porting considerations).

An extensive online guide for planning porting and migration efforts is provided by Novell at this site:

http://developer.novell.com/wiki/index.php/Porting_and_Migration

GTK+

GTK+ is a cross-platform graphical toolkit that is used by applications such as The GIMP, GAIM, and the GNOME desktop. GTK+ is written in C, although there are language bindings for C++, Python, Perl, and many more languages. GTK+ is licensed under the LGPL, and it can be used in both open source and commercial applications. More information about GTK+ can be found at:

<http://www.gtk.org>

Qt

Qt is a cross-platform graphical toolkit developed by Trolltech and is used by the KDE desktop and related applications. Qt is written in C++, although there are language bindings for C, Python, Ruby, and many more languages. Qt is dual-licensed under the GPL and a commercial license. Therefore, you can develop GPL compatible applications free of charge, but to develop commercial applications, you must purchase a license from Trolltech. More information about Qt can be found at:

<http://www.trolltech.com>

REALBasic

REALbasic, provided by REAL Software, Inc., is a cross-platform software development environment that also provides an extremely effective path for porting Visual Basic applications, so that they can run natively on a Linux client. A technical white paper is available that provides a detailed description of the porting process using REALbasic. The white paper can be accessed at the following URL:

<http://www.realbasic.com/vb>

wxWidgets

Formerly known as “wxWindows”, wxWidgets provides an open source C++ GUI framework for cross-platform programming. It lets developers create applications for Win32, Mac OS X, GTK+, X11, and so forth. Applications, which use wxWidgets, appear to be native on each platform that it targets. wxWidgets is licensed under the wxWidgets License, and can be used free of charge in both open source and commercial applications. More information about wxWidgets can be found at:

<http://www.wxwidgets.org>

<http://en.wikipedia.org/wiki/WxWidgets>

Some older tutorials are available at the IBM Developerworks Web site that explain how to use this toolkit in detail:

- ▶ “*Looking through wxWindows: An introduction to the portable C++ and Python GUI toolkit*”:

<http://www-106.ibm.com/developerworks/library/l-wxwin.html>

- ▶ “*Porting MFC Applications to Linux: A step-by-step guide to using wxWindows*”:

<http://www-106.ibm.com/developerworks/linux/library/l-mfc>

Mono and the .NET Framework

The Mono project is an open source implementation of the Microsoft .NET Framework. Mono is based off of the ECMA standards for the .NET Common Intermediate Language (CIL) and the C# language. The majority of the .NET standard libraries have been reimplemented on Mono. As of Mono version 1.1, WinForms, the graphical library used by most applications, which targets the Microsoft .NET Runtime, is only partially supported. However, work on WinForms has been rapidly progressing and is expected to be completed in Version 1.2. Mono also includes support for GTK#, a graphical toolkit that works on both Linux and Windows.

Both Mono and .NET support programming in more than just C# with languages such as Visual Basic.NET, IronPython, Nemerle, and Boo. Languages which target the Common Language Runtime (CLR) can access the entire framework, and they are seamlessly interoperable with other languages, even in the same application.

While some .NET applications can contain calls into native APIs, the majority of any given .NET application can easily be ported to Mono. More information about Mono can be found at:

<http://www.go-mono.com>

Some tutorials about porting from Microsoft .NET to Mono can be found at:

http://developer.novell.com/wiki/index.php/Porting_and_Migration

Related publications

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 338. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *Linux Handbook A Guide to IBM Linux Solutions and Resources*, SG24-7000
- ▶ *Domino Web Access 6.5 on Linux*, SG24-7060
- ▶ *OS/2 to Linux Client Transition*, SG24-6621
- ▶ *Provisioning On Demand Introducing IBM Tivoli Intelligent ThinkDynamic Orchestrator*, SG24-8888
- ▶ *Developing Workflows and Automation Packages for IBM Tivoli Intelligent ThinkDynamic Orchestrator*, SG24-6057

Other publications

These publications are also relevant as further information sources:

- ▶ *Printing with Linux on zSeries Using CUPS and Samba*, REDP-3864
- ▶ *Migrate Exchange 5.5 to Domino on Linux*, REDP-3777
- ▶ *Linux: Why It Should Replace Your Windows NT Domains*, REDP-3779
- ▶ *Open Your Windows with Samba on Linux*, REDP-3780
- ▶ *Using Asset Depot for Inventory Management*
- ▶ *The IDA Open Source Migration Guidelines*, netproject Ltd © European Communities 2003
- ▶ *Solution Guide for Windows Security and Directory Services for UNIX* available from Microsoft

Online resources

These Web sites and URLs are also relevant as further information sources:

- ▶ Red Hat Network Architecture
<http://www.redhat.com/software/rhn/architecture/>

- ▶ Webmin
<http://www.sourceforge.net/projects/webadmin>
<http://www.webmin.com>
- ▶ Big Brother/Big Sister
<http://www.sourceforge.net/projects/big-brother>
<http://www.sourceforge.net/projects/bigsister>
<http://www.bb4.org>
- ▶ Nagios
<http://sourceforge.net/projects/nagios>
<http://www.nagios.org/>
- ▶ Copies of the GNU GPL Licenses
<http://www.gnu.org/copyleft/gpl.html>
- ▶ Free Software Foundation (FSF)
<http://www.gnu.org/fsf/fsf.html>
- ▶ OpenOffice.org (Writer, Calc, Impress)
<http://www.openoffice.org>
- ▶ The GIMP
<http://www.gimp.org>
- ▶ GAIM
<http://gaim.sourceforge.net>
- ▶ rdesktop
<http://sourceforge.net/projects/rdesktop/>
- ▶ NoMachine NX
<http://www.nomachine.com>
- ▶ Knoppix
<http://www.knoppix.com>
- ▶ Personal Computing Support: Linux for IBM Personal Systems
<http://www-306.ibm.com/pc/support/site.wss/MIGR-48NT8D.html>
- ▶ Linux Standard Base
<http://www.linuxbase.org>
- ▶ KDE
<http://www.kde.org>
- ▶ SuperKaramba
<http://netdragon.sourceforge.net/>

- ▶ GNOME
<http://www.gnome.org>
- ▶ Significant GNOME desktop applications
<http://www.gnomefiles.org>
- ▶ GDesklets
<http://gdesklets.gnomedesktop.org/>
- ▶ SVG Clip Art Library
<http://www.openclipart.org>
- ▶ W3 standard SVG
<http://www.w3.org/Graphics/SVG/>
- ▶ Theme Web sites
 - <http://themes.freshmeat.net>
 - <http://www.customize.org>
 - <http://www.kde-look.org>
 - <http://art.gnome.org>
 - <http://www.crystalgnome.org>
- ▶ KDE OpenOffice integration and Mozillux projects
 - <http://www.polinux.upv.es/mozilla>
 - <http://kde.openoffice.org>
- ▶ Success story of Linux usability on the desktop
 - http://www.linux-usability.de/download/linux_usability_report_en.pdf
 - <http://www.userinstinct.com/viewpost.php?postid=gnome26review>
- ▶ UniConf
<http://open.nit.ca/wiki/>
- ▶ InterMezzo
<http://www.inter-mezzo.org>
- ▶ Linux at IBM Solutions page
<http://www.ibm.com/linux/solutions>
- ▶ Red Hat Network (RHN)
<http://www.redhat.com/software/rhn>
- ▶ ZENworks Linux Management
<http://www.novell.com/products/zenworks/linuxmanagement/>
- ▶ ndiswrapper project
<http://ndiswrapper.sourceforge.net>

- ▶ PC model types suitable for Linux
<http://www.ibm.com/pc/support/site.wss/MIGR-48NT8D.html>
- ▶ Common UNIX Printing System (CUPS)
<http://www.cups.org>
- ▶ Scanner Access Now Easy (SANE)
<http://www.sane-project.org>
- ▶ UNIX SmartCard Driver Project
<http://smartcard.sourceforge.net>
- ▶ Project gphoto2 - Supporting digital cameras
<http://www.gphoto.org>
- ▶ For winmodems, some Linux drivers are available for some models
<http://linmodems.org>
- ▶ Mozilla, Firefox, Thunderbird
<http://www.mozilla.org>
- ▶ iFolder
<http://www.novell.com/products/ifolder>
- ▶ rsync
<http://samba.org/rsync/>
- ▶ Kiosk Admin Tool
<http://extragear.kde.org/apps/kioskttool.php>
- ▶ Kiosk readme
<http://webcvs.kde.org/cgi-bin/cvsweb.cgi/kdelibs/kdecore/README.kiosk>
- ▶ Mono
<http://www.mono-project.com>
- ▶ openMosix
<http://www.openmosix.org>
- ▶ PXELinux
<http://syslinux.zytor.com/pxe.php>
- ▶ Red Hat Linux hardware probing library
<http://rhlinux.redhat.com/kudzu>
- ▶ *Introduction to Stateless Linux, Proposal for the Fedora Project*
<http://people.redhat.com/~hp/stateless/StatelessLinux.pdf>

- ▶ Purchase printing software supporting Linux
<http://www.easysw.com/printpro>
- ▶ Try a Windows PPD file provided by Adobe
<http://www.adobe.com/products/printerdrivers/winppd.html>
- ▶ *Samba HOWTO Collection* section 6.3
<http://samba.org/samba/docs/Samba-HOWTO-Collection.pdf>
- ▶ Red Hat Web page, Red Hat Network Architecture
<http://www.redhat.com/software/rhn/architecture/>
- ▶ Red Hat Web site
<http://www.redhat.com/rhn>
- ▶ *ZENworks Linux Management Administrator's Guide*
<http://www.novell.com/products/zenworks/linuxmanagement/>
- ▶ Evaluation copy of ZENworks 6.5 Linux management
<http://www.novell.com/products/zenworks/eval.html>
- ▶ REALbasic for porting Visual Basic applications so that they can run natively on a Linux client white paper
<http://www.realbasic.com/vb>
- ▶ wxWidgets provides an open source C++ GUI framework for cross-platform programming - project Web site
<http://www.wxwidgets.org>
- ▶ “Looking through wxWindows: An introduction to the portable C++ and Python GUI toolkit”
<http://www-106.ibm.com/developerworks/library/l-wxwin.html>
- ▶ “Porting MFC Applications to Linux: A step-by-step guide to using wxWindows”
<http://www-106.ibm.com/developerworks/linux/library/l-mfc>
- ▶ Free BASIC Compilers and Interpreters
<http://www.thefreecountry.com/compilers/basic.shtml>
- ▶ QSA overview
<http://www.trolltech.com/products/qsa/>
- ▶ *KDE for System Administrators Guide*
<http://www.kde.org/areas/sysadmin/>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Symbols

.NET 331
/etc/fstab 182
/etc/hosts.allow 179
/etc/hosts.deny 179
/etc/profile 315

A

AbiWord word processor 79
acceptance 68
ACPI 96, 235
Active Directory 195
Active Directory. See Windows domain
ActiveX® controls 185
Activity Explorer 160
administration planning 86
ALSA 235
anti-virus 66
APM 96, 235
application development
 platform 159
Application porting 329
application updates 88
applications
 portal-based 41
Archive 235
archivers 55
as-is-analysis 50
assessment
 client hardware 63
 client IT environment 63
 client software 65
 data dependencies 66
 infrastructure 67
Asset Depot 63
ATI 291
Awk 235

B

backup 90
 Tivoli Storage Manager 90
Bash 235

BDF Fonts 235
Big Brother/Big Sister 36
Boot Disk 236
Bootloader 236
Bot 236
Bourne Shell 236
bridging applications 53, 55
 separate retraining from migration 58
BSD 236
Bzip2 236

C

Cairo (vector graphics system) 112
card readers 96
CDROM
 bootable 59, 334
CGI 236
CIM object manager 256
CIM server 256
Citrix 56
Citrix Metaframe 100
CLI 237
Client 236
client environments 5
client personalization 313
CLR 331
Cluster 237
Command Line Interface (CLI) 237
Common Language Runtime 331
Common Unix Printing System. See CUPS
communications plan 57
configuration files 315
CORBA 111
cost 32
 application 33
 hardware 33
 license 32
 management 33
 support 32
CRM 55
CUPS 73, 219
 lpadmin 220
 samba integration 74

web interface 221
custom applications 3

D

 Daemon 237
 D-BUS (message bus system) 112
 DCOP 325
 Desktop automation 327
 interprocess communication 326
 Debian 7
 Debian Package Manager 238
 desktop
 design 38
 standardization 75
 themes 80
 Desktop automation with DCOP 327
 Device Driver 238
 DHCP 67, 75
 digital cameras 96
 Distributed Management Task Force 256
 Distribution 238
 distributions 7
 task oriented 37
 dmask options 183
 DMTF 256
 DNS 75
 document library 159
 Dpkg 238
 dual boot 102
 Dual-head video cards 291

E

 ease of use 38
 ECMA 331
 eGatherer 63
 Emacs 238
 end user survey 52
 Enlightenment 238
 enterprise management tools 255
 Environment variable 238
 ERP 55
 Evolution 186
 Example client migration 174
 Exchange 186
 Exchange connector 186

F

 File Extension 238
 file managers 55
 File System 239
 file systems 58
 file types 52
 Finger 239
 firewall 30
 Foreground Process 239
 Free Software Foundation (FSF) 7
 FreeBSD 239
 Fsck 239
 FSF (Free Software Foundation) 7
 fstab 182
 FTP 7, 239
 functional continuity 53

G

 GCC 239
 GConf 317
 gconf-editor 317
 editing entries with 318
 gconftool-2 152, 318
 GDesklets 79
 getent 201
 GIMP 239
 glossary 233
 GNOME 239, 317
 desktop environment 122
 Nautilus 55
 GNU 240
 GNU C Compiler 239
 GNU General Public License 110
 GNU General Public License (GPL) 6, 32
 GNU Image Manipulation Program 239
 GNU Network Object Model Environment 239
 GNU zip 240
 GNU/Linux 240
 goals 43
 GPL 110, 240
 graphical programming 122
 graphics
 SVG libraries and programs 80
 Grep 240
 GRUB 240
 GStreamer (streaming media framework) 112
 GTK+ 240, 330
 GUI 240

Gzip 240

H

HAL (hardware abstraction layer) 112
Hard Link 241
hardware support 64
Home directories 73
Home Directory 241
Host Publisher 68
HTTP 5
Human factors 56

I

IBM Productivity Tools 159
IBM Tivoli Intelligent ThinkDynamic Orchestrator 283, 333
IBM Workplace
 Managed Client productivity tool 159
ICA Client 56
iFolder 99
Image manipulation 185
impact of changes 57
infrastructure 67
Init 241
Instant Messaging (IM) 159
IntelliStation 290
Internet Engineering Task Force 257
Interpreted Language 241
ISO-9660 7
ISV 65
IT personnel, 57
ITEF 257

J

Java 6
Java desktops 6
JFS 242
joining a Windows domain 196
Journalized/Journaling File System 242
Journaling 242

K

K Desktop Environment 243
KConfigEditor 316
KDE 243
 desktop environment 122
 kdeglobals 316
kdesktoprc 316
kickerrc 316
kiosk framework 82
Konqueror 29, 55
Kerberos 197
 krb5.conf 197
Kernel 243
kernel modularity 35
kickerrc 316
KJSEmbed 324
Knoppix 59, 334
Kommander 325
Korn Shell 243
kudzu 323
KVM 302

L

LDAP 195, 204
 ldap.conf 206
Lesser GPL 243
LGPL 243
Linux
 why you need it 6
live CD 59, 334
look and feel
 customization 58
Lotus Domino 66
Lotus Notes 6, 66

M

Mac OS X 323
manageability 34
management tools
 enterprise 255
massive parallel processing (MPP) 7
message queueing 67
messaging
 security 29
MIB 257
Microsoft Internet Explorer 29
Microsoft Office 2–3, 66
Microsoft Outlook 66
Microsoft Project 66
Microsoft Windows
 client personalization 314
migrating applications 84
migration
 assessing client software 65

automated 187
context 4
goals 43
justification 25
path 5
pilot 44
modular structure 35
Mono 331
mount 72, 182, 214
 CIFS 215
 cifs 72
 smb 72
 smbfs 214
 Windows shares 72
Mozilla 6
 browser security 29
MPP (massive parallel processing) 7
multimedia players 55
Multi-station computing 289
 Deployment considerations 298
 Processor requirements 293

N

Nagios 36, 334
Name Service Switch 201
 ldap 206
 nsswitch.conf 201
 winbind 201
Nautilus
 connect to the Windows shares 187
ndiswrapper 95, 335
NetBIOS 197
network services integration 69
nice 294
NoMachine NX 56, 100, 334
Novell Connector 186
Novell Evolution 186
nsswitch.conf 180
NT4. See Windows domain
NVIDIA 291
NX 303

O

OASIS 160
Open Document Format (ODF) 159
open standards 5
OpenDocument format 160
OpenOffice.org 54

Opera browser 29
operating system updates 87
organizational readiness 41
OSS
 open source software philosophy 38
Outlook Web Access 186

P

PAM 72, 180, 208
 pam_ldap 210
 pam_mkhomedir 212
 pam_mount 72, 216
 pam_winbind 180–181, 208, 213
pam_mount 200
PCI bus expansion 302
Perl 322
Personal Communications 68
personalization 314
 GNOME Desktop 317
 KDE Desktop 315
 Linux client 314
pervasive technologies 5
pilot group 57
planning
 communications 57
 organizational and human factors 49
plotters 96
port scanning 66
ported applications 55
POSIX
 standard 6
 thread library 7
Postscript Printer Description 184
ppd 184
printer 64
printers and authentication 74
profile 314
profiles
 Windows NT 314
Progression Desktop
 Architecture 284
protocol 67
PyGTK 323
 PyQt 323
Python 323

Q

Qt 330

Script for Applications 324

R

rcd 266
rcman 266
rdesktop 56, 334
RDP 56, 302
RDP sessions 56
REALBasic 330
RealPlayer 55
Red Carpet 93
Red Hat 7
Red Hat Network. See RHN
Redbooks Web site 338
 Contact us xxii
remote access 35
 execution 35
 management 36
 support 36
remote administration 88
Remote Display Protocol 56
removable storage 295
renice 294
retraining considerations 57
RHN 36, 91, 257
 Hosted 91, 258
 management tool 34
 Proxy 91
 Satellite 91, 258
RHN modules 92
 Management 92
 Provisioning 92
 Update 92
roll-out 89

S

Samba 195–196
 as domain controller 69
smb.conf 196, 199
smbclient 215
smbmount 72
symbolic links 73
Satellite server 257
 action 261
 activation key 262
 architecture 257
 channel 261
 entitlement 260

errata 261
system 260
system group 260
system set 260
scalability 6
scanners 96
Scripting languages 322
 Embedded 323
security
 browser 29
 desktop 27
 firewalling 30
 messaging-client 29
 user fencing 30
Server Message Block (SMB) protocol 69
Service Oriented Architecture (see SOA)
Shell scripting 322
Simple Network Management Protocol 257
Simple Network Time Protocol 198
Single Sign On 72
SLIP 7
smb.conf 179, 199
smbmount 182
SNMP 257
SNTP 198
SOA 5
SOAP 5
stability 6
standardizing the desktop 75
strategy 3
survey 50
surveying user data 52
SUSE 7
SVG libraries and programs 80
symbolic links 73
system_auth 181

T

Task oriented distributions 37
TCP/IP
 networking stack 6
Telnet 7
Terminal server 100
themes 80
Tivoli Configuration Manager 36
Tivoli Intelligent ThinkDynamic Orchestrator 283, 333
tsclient 185

U

up2date 259
updates 35
usability 77
usage patterns 50
user roles 68
Useful Desktop Multiplier 289
utility applications 54

printing services 73

shares 72

Windows NT 314
migrating from 4
profiles 314
Windows Terminal Server 56, 184
WinForms 331
wxPython 323
wxWidgets 330

V

Versora Progression Desktop
Architecture 284
viewers 55
viruses 90
worms 30
Visual Basic 3, 66, 329
VMware 100–101
VNC 178, 303
vncserver 179

X Window System
layers 119
X11 75
Xdmcp 178
XDMCP, 303
xine 55
XML 5, 151
xmms 55
Xnest 122

W

WBEM 256
Web applications 55
Web browser 159
Web services 5–6
Web-Based Enterprise Management 256
Web-based Enterprise Management 256
WebDAV 186
Webmin 36, 334
Win32 3
winbind 179, 195, 199
 idmap 89
 idmap file 89
 idmap_ad 203
 idmap_rid 203
 separator 71, 200
 wbinfo 201
winbind daemon 199
window managers 121
Windows domain 70, 196
 Active Directory 197
 join 198
 Service for Unix 71
 Services for Unix 204
 authentication 70
 domain controllers 69
NT4 196
 join 196

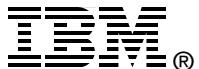
Yast online Update 37
yum 259

Z

ZENworks Linux Management 34, 36, 93, 264
 Activation key 269
 Administrator 268
 Channel 268
 Group 268
ZENworks Linux management
 Activation key 267
 Channel 267
 Group 267
 Machine 267
 Machine set 267
 Transaction 267



Linux Client Migration Cookbook, Version 2



Linux Client Migration Cookbook, Version 2

A Practical Planning and Implementation Guide for Migrating to Desktop Linux



Redbooks

**For any organization
that is exploring or
planning for a Linux
desktop migration**

**Provides in-depth
detail on the
technical and
organizational
challenges**

**Includes methods
for planning and
implementation**

The goal of this IBM Redbook is to provide a technical planning reference for IT organizations large or small that are now considering a migration to Linux-based personal computers. For Linux, there is a tremendous amount of "how-to" information available online that addresses specific and very technical operating system configuration issues, platform-specific installation methods, user interface customizations, and so forth. This book includes some technical "how-to" as well, but the overall focus of the content in this book is to walk the reader through some of the important considerations and planning issues that you could encounter during a migration project. Within the context of a pre-existing Microsoft Windows-based environment, we attempt to present a more holistic, end-to-end view of the technical challenges and methods necessary to complete a successful migration to Linux-based clients.

This second version of the Linux Client Migration Cookbook builds on the content strategy we started with in the first version. Although anyone interested in using Linux on the desktop could benefit from different portions of this book, our primary audience for this book is existing business IT environments that need to begin an evaluation of desktop Linux, or in a broader sense, any organization whose strategy is to move toward greater adoption of open source software and open standards.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**