Running Linux Applications on AIX

AIX affinity with Linux
AIX Toolbox for Linux Applications
Porting and source compatibility

Luis Ferreira
Janethe Co
Jan-Rainer Lahmann
Gonzalo Quesada

ibm.com/redbooks
International Technical Support Organization

Running Linux Applications on AIX

June 2001
## Contents

**Preface** ................................................................. xi  
The team that wrote this redbook ........................................... xi  
Special notice ..................................................................... xiv  
IBM Trademarks ................................................................. xv  
Comments welcome ............................................................ xv  

**Chapter 1. Introduction** .................................................. 1  
  1.1 AIX ............................................................... 2  
  1.1.1 Current version and status ........................................ 2  
  1.2 Linux .................................................................. 4  
  1.2.1 Brief history ...................................................... 4  
  1.2.2 About Linux’s copyright .......................................... 5  
  1.2.3 The GNU Project and the Linux kernel ....................... 5  
  1.2.4 Different flavors of Linux ......................................... 6  
  1.3 Linux at IBM ........................................................ 7  
  1.3.1 Linux applications on AIX ....................................... 8  
  1.3.2 IBM’s UNIX-based operating system strategy ............... 8  
  1.4 Future trends and directions ......................................... 10  

**Chapter 2. AIX Toolbox for Linux Applications** .................. 13  
  2.1 Overview ......................................................... 15  
  2.2 Additional information .............................................. 16  
  2.3 Design of the Toolbox ............................................ 17  
  2.3.1 Directory structure ........................................... 17  
  2.3.2 System variables ............................................. 20  
  2.4 Components ....................................................... 21  
  2.4.1 Development utilities ......................................... 21  
  2.4.2 User environment utilities and applications ............... 21  
  2.4.3 Binaries and sources (rpm and srpm) ....................... 22  
  2.5 Installation methods ............................................. 22  
  2.5.1 AIX instalip .................................................. 24  
  2.5.2 RPM Package Manager ....................................... 30  

**Chapter 3. Toolbox installation** ....................................... 37  
  3.1 System requirements ............................................. 38
3.2 Installation procedure
3.2.1 Installing the RPM Package Manager
3.2.2 Preparing to install GNOME, KDE2 and other applications
3.2.3 FTP tools
3.2.4 Installing the Toolbox base
3.2.5 Using the RPM Package Manager
3.2.6 Installing KDE2
3.2.7 Installing GNOME
3.3 Useful URLs

Chapter 4. Source compatibility: Linux-compatible APIs on AIX
4.1 Writing portable code
4.2 Linux-compatible APIs and LSB functions on AIX
4.3 File macro supported values
4.3.1 File access modes
4.3.2 File descriptor flags for fcntl
4.3.3 File modes
4.3.4 Poll macro values
4.4 Signal values

Chapter 5. Package building and porting
5.1 Compiler installation and requirements
5.2 Rebuilding Toolbox packages
5.2.1 Building packages with rpm
5.2.2 Rebuilding a Toolbox RPM
5.3 Compiling open source software
5.4 Using libtool to handle shared libraries
5.5 Examples
5.5.1 Rebuilding and updating the wget package

Chapter 6. User and administration differences
6.1 Desktop and graphical applications
6.1.1 The XWindow System
6.1.2 The KDE desktop
6.1.3 The GNOME desktop
6.1.4 Package managing using KDE or GNOME
6.1.5 CDE desktop
6.2 Available shells
6.2.1 Overview of shell startup files
6.3 Commands and syntax differences
6.3.1 AIX and AIX Toolbox commands differences
6.4 Administration differences
6.5 Boot process differences
6.5.1 Linux boot process
6.5.2 AIX boot process .............................................. 143
6.6 System files differences ......................................... 147
  6.6.1 File system definitions on AIX and Linux .............. 148

Appendix A. APIs ..................................................... 153
  Linux-compatible APIs and library functions .................. 154
    Linux-compatible APIs ...................................... 154
    Linux Standard Base APIs .................................. 162
  New APIs in AIX 5L 5.1 ...................................... 171

Appendix B. Differences in commands ............................. 187

Appendix C. Other Open Source Software for AIX .................. 193
  Overview ..................................................... 194
  Other sources ................................................ 194

Related publications ............................................. 195
  IBM Redbooks ................................................. 195
    Other resources ............................................ 195
      Referenced Web and FTP sites .......................... 195
  How to get IBM Redbooks .................................. 198
    IBM Redbooks collections ................................ 198

Special notices .................................................. 199

Index .......................................................... 203
Running Linux Applications on AIX
Figures

1-1 IBM’s UNIX-based operating system strategy ............................................. 9
1-2 IBM’s UNIX-based operating system evolution ............................................. 11
2-3 /opt/freeware/man tree .............................................................................. 19
2-4 /usr/opt/freeware/src tree ........................................................................ 20
2-7 SMIT installp panel (text-based) ................................................................. 27
2-8 Main SMIT installation panel (GUI interface) .............................................. 28
2-9 SMIT dialog panel ....................................................................................... 29
2-10 SMIT Command Output panel .................................................................. 30
2-11 Sample of RPM package label convention ................................................. 31
2-12 Main GnomeRPM panel .............................................................................. 34
2-13 Main KPackage panel ............................................................................... 36
6-1 AIX Toolbox for Linux Applications graphical framework .......................... 93
6-2 KDE desktop main panel ............................................................................. 97
6-3 KDE desktop and its main panel ................................................................ 97
6-4 Adding an application to the starter menu .................................................. 98
6-5 Result from adding an application to the starter menu ............................... 99
6-6 KDE File Manager main window .................................................................. 100
6-7 KOffice sample 1 ........................................................................................ 101
6-8 KOffice sample 2 ......................................................................................... 102
6-9 GNOME panel ............................................................................................ 103
6-10 GNOME desktop ....................................................................................... 104
6-11 KPackage GUI interface ........................................................................... 105
6-12 GnomeRPM (gnorpm) main window .......................................................... 106
6-13 Display package information using gnorpm ............................................. 107
6-14 gnorpm web find feature .......................................................................... 108
6-15 gnome settings for rpmfind ....................................................................... 108
6-16 CDE front panel ........................................................................................ 110
6-17 Main panel action tasks ............................................................................. 111
6-18 Subpanel example ..................................................................................... 112
6-19 Main diag menu ........................................................................................ 130
6-20 Linuxconf graphical interface ................................................................... 135
6-21 YaST interface ........................................................................................... 136
6-22 Main SMIT menu in text-based interface .................................................. 137
6-23 SMIT user account menu ......................................................................... 139
6-24 Linuxconf user account menu ................................................................. 140
6-25 YaST user account menu .......................................................................... 141
# Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>installp option summary</td>
<td>26</td>
</tr>
<tr>
<td>3-1</td>
<td>Disk space requirements for the components of the Toolbox</td>
<td>38</td>
</tr>
<tr>
<td>4-1</td>
<td>File descriptor and system table definition</td>
<td>61</td>
</tr>
<tr>
<td>4-2</td>
<td>File access mode macro value comparison</td>
<td>62</td>
</tr>
<tr>
<td>4-3</td>
<td>File open mode macros on Linux and AIX</td>
<td>63</td>
</tr>
<tr>
<td>4-4</td>
<td>File open mode macros available only in Linux using _USE_GNU</td>
<td>63</td>
</tr>
<tr>
<td>4-5</td>
<td>Linux open mode using _USE_LARGEFILE64</td>
<td>64</td>
</tr>
<tr>
<td>4-6</td>
<td>Linux open modes using _USE_POSIX199309 or _USE_UNIX98</td>
<td>64</td>
</tr>
<tr>
<td>4-7</td>
<td>Poll macro values</td>
<td>65</td>
</tr>
<tr>
<td>4-8</td>
<td>Signal values</td>
<td>66</td>
</tr>
<tr>
<td>6-1</td>
<td>Desktops and window managers</td>
<td>90</td>
</tr>
<tr>
<td>6-2</td>
<td>AIX standard shells feature comparison</td>
<td>114</td>
</tr>
<tr>
<td>6-3</td>
<td>AIX Toolbox for Linux Applications shell feature comparison</td>
<td>123</td>
</tr>
<tr>
<td>6-4</td>
<td>Login execution sequence for ksh, csh, and sh</td>
<td>124</td>
</tr>
<tr>
<td>6-5</td>
<td>Login execution sequence for bash, tcsh, and zsh</td>
<td>124</td>
</tr>
<tr>
<td>6-6</td>
<td>Commands differences examples</td>
<td>129</td>
</tr>
<tr>
<td>6-7</td>
<td>Function of -a flag in AIX and Linux</td>
<td>132</td>
</tr>
<tr>
<td>6-8</td>
<td>Linux distribution and their administration tools</td>
<td>134</td>
</tr>
<tr>
<td>6-9</td>
<td>SMIT menu examples and their corresponding fast paths</td>
<td>137</td>
</tr>
<tr>
<td>6-10</td>
<td>Basic SMIT tasks</td>
<td>138</td>
</tr>
<tr>
<td>6-11</td>
<td>Differences in configuration files between AIX and Linux</td>
<td>147</td>
</tr>
<tr>
<td>A-1</td>
<td>Different groups of APIs</td>
<td>155</td>
</tr>
<tr>
<td>A-2</td>
<td>Compatible APIs</td>
<td>155</td>
</tr>
<tr>
<td>A-3</td>
<td>APIs not implemented</td>
<td>159</td>
</tr>
<tr>
<td>A-4</td>
<td>Linux-compatible APIs introduced in AIX 5L 5.1</td>
<td>161</td>
</tr>
<tr>
<td>A-5</td>
<td>Linux-compatible APIs available on AIX but not 100% compatible</td>
<td>161</td>
</tr>
<tr>
<td>A-6</td>
<td>Different groups of LSB APIs</td>
<td>162</td>
</tr>
<tr>
<td>A-7</td>
<td>Compatible LSB APIs</td>
<td>162</td>
</tr>
<tr>
<td>A-8</td>
<td>LSB APIs not available</td>
<td>170</td>
</tr>
<tr>
<td>A-9</td>
<td>LSB APIs introduced in AIX 5L 5.1</td>
<td>170</td>
</tr>
<tr>
<td>A-10</td>
<td>LSB APIs available on AIX but not 100% compatible</td>
<td>170</td>
</tr>
</tbody>
</table>
Preface

The strengths of the AIX operating system are well known among the UNIX® software community. Its reliability and great degree of scaling makes AIX the perfect choice for hosting mission-critical applications. It is a robust and flexible operating system that meets all the requirements for the current demands of e-business environments. At the same time, Linux® is emerging and generating excitement among software developers that has not been seen in years.

With the adoption of Linux in early 2000, IBM became very interested in enabling Linux applications to run on the AIX operating system. Thus, the AIX Toolbox for Linux Applications was developed. The Toolbox provides the capability to easily recompile and port Linux applications to AIX and provides tools to work on those applications. Countless developers around the world are completely focused on developing applications for Linux systems. Now you can easily port these applications and run them directly on AIX while taking advantage of all the features and benefits that the AIX operating system offers.

This redbook will show you what you need to run Linux applications on AIX. We will help you comprehend and install the AIX Toolbox for Linux Applications, understand the procedure to follow for porting open source software, and explain the usage of Linux commands on AIX.

The team that wrote this redbook

This redbook was produced by the Blue Tuxedo Team, a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Luis Ferreira (also known as “Luix”) is a Software Engineer at IBM Corp - International Technical Support Organization, Austin Center working with Linux projects. He has 18 years of experience in UNIX-based operating systems, and holds a MSc. Degree in System Engineering from Universidade Federal do Rio de Janeiro, Brazil. Before joining the ITSO, Luis worked at Tivoli Systems as a Certified Tivoli Consultant, at IBM Brasil as a Certified I/T Specialist, and at Cobra Computadores as a kernel developer and software designer.

Janethe Co is a System Services Representative at IBM Philippines. She holds a Bachelor of Science Degree in Electronics and Communications Engineer from
De La Salle University, Manila. Her areas of expertise include AIX, Linux, Security and Firewall, Numa-Q, and Dynix/ptx.

**Dr. Jan-Rainer Lahmann** is an @server Architect in Germany. He holds a Ph.D. Degree in applied mathematics from University of Karlsruhe, Germany. He has eight years of experience in managing RS/6000 and Linux clusters. After joining IBM, he was trained in OS/390, especially in its UNIX System Services, and Linux for S/390 (when it became available in early 2000). Currently, he is a technical expert for Linux environments on both the pSeries and zSeries systems.

**Gonzalo Quesada** is an AIX Systems Specialist in GBM Costa Rica, an IBM Alliance Corporation. He has eleven years of experience in UNIX systems and four years of experience with RS/6000 SP Systems. He has worked with GBM for eight years supporting the IBM RS/6000 family of products. He is an IBM Certified Specialist for AIX System Support RS/6000 and a IBM Certified Specialist for Web Server for RS/6000. He also has done consulting work on cross-platform environments involving relational database connectivity and network management. He has written extensively on different ITSO projects, such as DRDA/Datajoiner, AIX 4.2, and Using TME 10.
The Blue Tuxedo Team is pictured here. They are, left to right, Jan, Janethe, Luis, and Gonzalo.

Acknowledgements
The team would like to express special thanks to the following people for their major contributions to this project:

IBM Austin
Mark Brown, Senior Technical Staff Member and Marc Stephenson, Senior Software Engineer

Thanks to the following people for their contributions to this project:

International Technical Support Organization, Austin Center
Scott Vetter, Richard Cutler, Lupe Brown, Gwen Monroe, Daesung Chung, Wade Wallace, Joanne Luedtke

International Technical Support Organization, Poughkeepsie Center
Fred Borchers
IBM Austin
Sharon Dobbs, Susan Pelzel, Ahmed Chibib, Dan McNichol, David Clissold, Mike Harrell, Ray Hebert, Reza Arbab

IBM Germany
Andreas Ehrhardt and Eberhard Pasch

IBM England
Nigel Griffiths

IBM Brazil
Helcio Caruso, Vicente Melo, Andre Castro, Marcus Ferreira

IBM Japan
Shigeo Murohashi

Technical University of Clausthal, Germany
Stefan Schnitter

Also, thanks to Linus Torvalds for rescuing the dream.

Special notice

This publication is intended to help software engineers and developers to utilize the AIX Toolbox for Linux Applications, establish a development environment for porting open source software from different sources, and give porting tips.

The information in this publication is not intended as the specification of any programming interfaces that are provided by IBM for AIX and AIX 5L and by any Linux documentation. See the PUBLICATIONS section of the IBM Programming Announcement for AIX 5L for more information about what publications are considered to be product documentation.
IBM Trademarks

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

- e (logo) ™
- IBM®
- AIX
- AIX 5L
- DB2
- pSeries
- RS/6000
- PowerPC
- S/390
- OS/2
- Tivoli
- Redbooks
- Redbooks Logo ™
- SP2
- Netfinity
- xSeries
- zSeries
- WebSphere
- POWER
- ServeRAID
- OS/390
- TME

Comments welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please 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:
  ibm.com/redbooks

▷ Send your comments in an Internet note to:
  redbook@us.ibm.com

▷ Mail your comments to address on Page ii
Introduction

This chapter provides a brief history of IBM’s version of the UNIX operating system (known as AIX), the status of its current version (AIX 5L), and its planned direction. It also provides a brief introduction to another very popular UNIX-based operating system called Linux, its different flavors, and why it plays a major role in the IBM AIX roadmap.

In this chapter, the following topics are described:

- AIX, including a brief history of AIX
- The current version of AIX and its features and status
- A brief history of Linux
- Linux copyrights
- GNU and the Linux kernel
- Different flavors of Linux
- Linux in IBM and future directions
1.1 AIX

The IBM AIX operating system is a modern UNIX operating system for the RS/6000 family that supports 32-bit and 64-bit applications. It provides a great degree of scaling, including uniprocessors, symmetric multiprocessors, clusters, and massively parallel systems, using a single consistent set of application and binary interfaces. AIX is highly reliable and supports high availability storage systems and various cluster configurations for near fault-tolerant operations.

AIX is designed for mission-critical, core business applications, providing an integrated environment that is stable, highly scalable, and functionality rich. It interoperates with many heterogeneous platforms and offers powerful management solutions for the enterprise.

IBM took the basic UNIX operating system and incorporated enhancements developed by other organizations and institutions, for example, BSD® UNIX (Berkeley Software Distribution), included new features and POSIX™ IEEE 1003.1 standards conformance, and added many enhancements to produce the Advanced Interactive eXecutive (AIX).

Today, AIX is the operating system that runs on IBM’s family of RISC workstations and servers, known as IBM RS/6000 or IBM @server pSeries models.

1.1.1 Current version and status

Over the past five years, the vast growth of Internet users and the explosion of e-business has produced a large demand on server computing requirements. These trends are placing unprecedented demands on IT systems and creating the need for more advanced technology and standardization. The new e-business environments demand an OS capable of handling the mission-critical functional requirements that allow businesses to focus on delivering a greater value to customers.

As the next generation of AIX, AIX 5L fulfills the required needs of e-business environments and provides even more capabilities, including enhanced support for IBM POWER platforms, support for the Intel® Itanium™ architecture, and a strong affinity with Linux.

IBM AIX 5L was designed with one simple goal in mind: to produce a single UNIX-based product line with broad industry support and to establish AIX as the leading open, industrial strength UNIX-based operating system.

AIX 5L is currently shipping and includes many new features, a description of which can be found at:
The benefits of these new features are:

- **Binary compatibility**
  - Helps assure continuing application availability across AIX Version 4 when moving to Version 5.

- **System scalability**
  - Helps preserve application investments and enables a clean transition to 64-bit applications.
  - Helps increase Web server performance by serving Web pages from the AIX network file cache.
  - Enables tuning of disk I/O and optimization of RAID storage.

- **Network performance**
  - Provides performance enhancements to the communication subsystem, such as higher throughput of the TCP/IP stack.
  - Enables multiple routes to a destination for bandwidth aggregation or high-availability configurations.
  - Facilitates fast propagation of hardware address changes.
  - Automatically provides network file system mounts, reducing the work of system administration.
  - Increases IP addressability, security, and integrity through redundant routing, dynamic rerouting, on-demand tunneling, and gateway support.

- **Security**
  - Enhanced password protection under IPv4.
  - Enhances the IP Security function in AIX for Virtual Private Networking.
  - Provides extensive base security capabilities, such as user/password management, security auditing, resource limits, and network security.

- **Systems and workload management**
  - Provides multiple AIX instances on a single POWER4 SMP system with full isolation protection.
  - Helps solve the problems of mixed workload management by providing resource availability to critical applications.

- **Base Operating System and RAS**
  - Improved random write performance on mirrored logical devices.
  - Provides new tools that simplify LVM hot spot management.
– Provides enhanced flexibility without rebooting, such as the ability to deactivate active paging spaces.
– Allows an AIX system image (mksysb) to be placed onto recordable CD-ROMs.
– Supports the /proc file system, which provides access to the state of each process and thread in the system.

► Enhanced Java™ support
– Improved performance, scalability, and stability.
– Enhances flexibility in developing and executing Java applications by allowing concurrent support of multiple Java versions installed on a system.

► Standards leadership
– Increases availability of applications, tools, and middleware from developers supporting open standards.
– Improves scalability, performance, and the assurance of application support threads.

1.2 Linux

The Linux operating system is a free UNIX-based operating system that supports full multitasking, the XWindows™ System, TCP/IP networking, and much more.

In the past few years, Linux has generated more excitement in the computer industry than any other development. Linux can basically run on a large variety of computer systems, turning them into powerful workstations that give you the power of UNIX software at your fingertips.

1.2.1 Brief history

Linux is a freely distributed operating system based on the UNIX operating system. It was originally developed by Linus Torvalds, who started work on Linux in 1991 as a student at the University of Helsinki in Finland. Torvalds was inspired by Andrew Tanenbaum’s Minix, a small UNIX-based operating system.

The initial release of Linux was distributed by means of the Internet, and generated one of the largest software development phenomena of all time.
The first official release of Linux, version 0.02, took place on October 5, 1991; at this point, Torvalds was able to run bash (the GNU Bourne Again Shell) and gcc (the GNU C compiler). Basically, Linux was intended as a hacker’s system. The situation has now changed, and the operating system provides a solid graphical environment, easy-to-install packages, and high-level applications.

Linux was initially developed for the Intel x86 architecture platform, but it is important to know that Linux now supports many other hardware platforms, such as PowerPC, S/390, SPARC®, and Alpha®.

### 1.2.2 About Linux’s copyright

The Linux kernel is written, distributed, and covered under the GNU General Public License (GPL), which means that its source code can be freely distributed and is available to the general public.

For information regarding GNU/Linux copyrights, the GNU Project, and the GNU General Public License (GPL), please refer to the following URL and Section 1.2.3, “The GNU Project and the Linux kernel” on page 5:

http://www.gnu.org/

### 1.2.3 The GNU Project and the Linux kernel

By the 1980s, operating systems were basically proprietary, which meant that you had to use the operating system provided for that specific platform.

The initiative of the Free Software Foundation and the GNU Project motivated and stimulated open development and worldwide user cooperation. The main goal of the GNU Project was to develop a UNIX-compatible operating system named GNU (GNU is Not Unix), capable of running on various hardware architectures. Calling it GNU was a way of paying tribute to UNIX-like systems while saying that GNU was something different. It was to be 100 percent free, which meant that users would be free to redistribute the whole system, and free to change and contribute to any part of it. It was decided to make it UNIX-compatible because UNIX had already been proven in terms of design and portability.

The GNU Project was founded by Richard Stallman, the founder of the Free Software Foundation, author of the GNU General Public License, and the original developer of some GNU software programs (for example, the gcc compiler and the Emacs text editor).
It took many years of hard work to write all the pieces of the GNU-based operating system, hundreds of programmers worldwide, and many hackers who used and worked very hard on the code. By 1990, most of the software pieces had been written except for the most important one: the kernel. The kernel is the core of the operating system. It is the piece of code that directly communicates and controls the interface between the user programs and the hardware devices (for example, disks, keyboard, mouse, and video). By that time, the free UNIX-based kernel developed by Linus Torvalds was combined with the GNU system, resulting in a complete operating system: the Linux-based GNU system.

Today, the combination of GNU tools and commands and the Linux kernel is widely used around the world, and its popularity grows on a daily basis.

### 1.2.4 Different flavors of Linux

As a benefit of the source code for the Linux kernel being freely distributed, different companies have developed their own “flavor” or distribution of Linux. Each of these flavors has its own feature set, such as installation and administration procedures, software packages, and configurations. Many of them are configured for a specific type of computer systems.

Some of the most popular distributions are:

- **Caldera® OpenLinux®**
  Developed by Caldera Systems, Inc.

- **Corel® Linux**
  Developed by Corel Corporation

- **Debian™ GNU/Linux**
  Developed by The Debian Project

- **Linux Mandrake®**
  Developed by MandrakeSoft, Inc.

- **Red Hat® Linux**
  Developed by Red Hat, Inc.

- **SuSE® Linux**
  Developed by SuSE, Inc.

- **TurboLinux®**
  Developed by TurboLinux, Inc.
As early as 1995, IBM Research and recognized experts in the Linux community ported Linux to the native PowerPC architecture platform and a Linux kernel (Version 2.2) for the IBM RS/6000 was developed. The initial RS/6000 support, following PowerPC Reference Platform (PReP) and Common Hardware Reference Platform (CHRP) specifications, was provided by Yellow Dog Linux™ on the IBM produced machines, such as the 7043-150, 7025-F50, and 7046-B50.

Today, Linux/PPC kernel Version 2.4 is available and is known to work on Power3 Uni and SMP machines. These include the models 170, 260, 270, and @server pSeries p640.

For more information regarding Linux on PPC, please refer to:
http://linuxppc.org/
and
http://www.rs6000.ibm.com/linux/

1.3 Linux at IBM

IBM is focusing on Linux because of the increased mind share and market share that Linux is getting, the rapid market changes, and the customer needs. Also, Linux is a stable and reliable development and deployment platform for Internet applications. Its low cost and broad platform support allow applications to be developed on commodity hardware and deployed across a wide range of systems.

Linux can be acquired at no cost as a download from the Internet, and the kernel and most of the extensions are available as source code and can be improved by anyone willing to contribute.

Linux is a very popular operating system for Web servers and dedicated networking functions, such as Web infrastructure, file-and-print serving, firewalls, directory serving, e-mail serving, and so on. Linux has also gained acceptance as an embedded OS for new Internet and other application appliances.

It is a different story in the enterprise arena; that is why AIX is IBM's strategic UNIX operating system for mission-critical, core business applications. The industrial-strength features and functions of AIX have been well proven over the years in a wide variety of server environments, from relatively small, single-processor systems to IBM's massively parallel RS/6000 Scalable POWERParallel (SP) servers. Features include 32-bit and 64-bit Application Programming Interface (API) support, state-of-the-art preemptive kernels, dynamic configuration and device attachments, a robust journaled file system,
Logical Volume Manager (LVM) software, the simplified system administration commands System Management Interface Tool (SMIT) and Web-based System Manager (WebSM), industry standards compliance, high-availability cluster multiprocessing (HACMP) software products, and more than 13,000 supported customer applications.

1.3.1 Linux applications on AIX

There is a strong affinity between Linux and AIX for applications. AIX has a long history of standards compliance and it is generally straightforward to rebuild Linux applications for AIX.

To make AIX more compatible with Linux applications, we must use two complementary methods: using the AIX Toolbox for Linux Applications, and including additional Linux-compatible APIs and commands in AIX 5L.

The current differences in terms of APIs are discussed in Chapter 4, “Source compatibility: Linux-compatible APIs on AIX” on page 55. This information should be used as a guideline when developing or porting any Linux application that will be used on AIX.

The AIX Toolbox for Linux Applications contains a collection of open source and GNU software built for AIX Version 4.3.3 and AIX 5L on IBM RS/6000 and IBM @server pSeries systems. These tools provide the basis for the development environment of choice for many Linux application developers. All the tools are packaged using the easy-to-install RPM format.

1.3.2 IBM’s UNIX-based operating system strategy

The IBM strategy for UNIX-based operating systems is built upon the great momentum that AIX is having, the establishment of AIX 5L as an enterprise class, industry leading, UNIX-based system with support for POWER/Intel/NUMA architectures, and a solid affinity with Linux.

Linux is being positioned as the strategic, high volume UNIX-based operating system. Enabling Linux across all IBM @server platforms is also an important part of our strategy. This allows porting applications to all of these platforms with little to no changes required to the source code.

In Figure 1-1 on page 9, we show:

- How AIX (on the RS/6000) is gaining tremendous market momentum as the industrial strength UNIX-based platform for mission-critical environments.
- The renewed ISV enthusiasm for AIX.
The Linux compatibility that will help drive AIX to be more open, as opposed to being thought of as IBM’s proprietary UNIX-based operating system.

- The integration of AIX and Linux in customer environments.

Figure 1-1 presents how IBM is bringing these strengths not only to its POWER architecture, but also to the Intel Itanium architecture, resulting in a single environment that customers can use on both POWER and Intel architectures. Also, it demonstrates IBM’s commitment to the UNIX philosophy and gives reassurances that IBM is producing an open industry platform.
1.4 Future trends and directions

The high level of activity on the UNIX-based systems and Linux fronts during the past few years is allowing Linux to establish itself as a mainstream UNIX player. It looks as though Linux is going to be transformed into an enterprise class operating system.

We expect the best of both worlds, AIX and Linux, to be the foundation of IBM AIX 5L: an operating system capable of working on IBM’s POWER and PowerPC architectures and Intel Itanium architecture.

IBM plans to bring new features and consistent user functionality to AIX 5L, such as:

- LPAR support (Logical Partitioning)
- Intel Itanium processor support
- Linux - AIX networking inter-operability
  - NFS
  - FTP
  - DNS
  - DHCP
  - SMB
  - PPP
  - IPv6
  - TCP (including extensions)
  - RFC1323 (Allows the maximum TCP window size to expand to 4 GB instead of 64 KB)
  - Window policy (Dynamically determine the absolute upper bound on the amount of real memory that can be used by the communication subsystem)
  - IPSec (IP Security Protocol)
  - LDAP (Lightweight Directory Access Protocol)
- Enhanced AIX Toolbox for Linux Applications
  - System Management capabilities
    - User administration
    - Print management
    - System management tools
– Bigger and more robust applications (enterprise-like)

Figure 1-2 lays out the road ahead regarding IBM’s perception of the UNIX-based operating system evolution. The direction for the IBM UNIX-based operation system evolution can be shown by the integration of the latest technology trends in operating system architectures, such as AIX, Sequent Dynix/PTX (NUMA (Non-Uniform Memory Access) architecture), and a solid Linux affinity with the AIX 5L operating system.

By incorporating this technology-sharing philosophy into a common application environment, new hardware trends, such as Internet appliances and embedded Linux for PDA (Personal Digital Assistant, also known as handheld computing), will gradually evolve and become mature.
Chapter 2. AIX Toolbox for Linux Applications

With the adoption of Linux in early 2000, IBM became very interested in enabling Linux applications to run on the AIX operating system. Thus, the AIX Toolbox for Linux Applications was developed. The Toolbox provides the capability to easily recompile and port Linux applications to AIX and provides tools to work on those applications. This chapter provides information on and an in-depth discussion of the Toolbox, its benefits, structure, and key components.

**Important:** For a complete description, list of packages, and detailed documentation, please refer to the AIX Toolbox for Linux Application Web site.

For a general description see:

For a complete and updated list of all packages, licenses, and installation instructions, see:

The Toolbox is also available on CD. At the time of writing, it was not separately orderable. The AIX Toolbox for Linux Applications CD is available with AIX 5L.
The most common ways of installing applications on both the native Linux and the classical AIX environments are discussed in this chapter. Both methods are important and will be used for the Toolbox installation. Section 2.3, “Design of the Toolbox” on page 17 will describe each installation method in greater detail.

This chapter is divided into the following sections:

- Overview
- Design of the Toolbox
- Structure of the Toolbox
- Components
- Installation methods
2.1 Overview

AIX has a long history of standards compliance, such as X/Open, UNIX98, and POSIX. Because of this history, there is a high degree of compatibility at the Application Programming Interface (API) level between AIX and other flavors of UNIX-based systems, such as Linux. AIX is a mission-critical operating system developed for scalability and stability. By porting and running Linux applications on AIX using the Toolbox, you will get the benefits of both worlds. The goal of the AIX Toolbox for Linux Applications is to be able to recompile Open Source Software (OSS), without modifications, into AIX systems. The recompiled Linux applications are treated as native AIX applications and inherit the reliability and availability of AIX.

Linux applications that are written using the standard Linux-compatible APIs can be recompiled to run on the AIX operating system using the AIX Toolbox for Linux Applications. The binaries are created using the appropriate tools and compilers from the Toolbox. All the tools in the Toolbox are packaged using the RPM Package Manager (the binary packages are called RPMs).

The main reasons for using the AIX Toolbox for Linux applications are:
- Building and packaging Linux applications for use on AIX
- Running GNOME and KDE desktops
- Running other popular software commonly found in Linux distributions
- Managing open source software using the popular RPM Package Management system
- Developing new applications for AIX using GNU and Linux application development tools

These are some of the benefits of AIX Toolbox for Linux Applications:
- Redeployment of Linux applications on AIX
- Reduction of deployment time of new systems
- Consolidation of emerging Linux applications on existing AIX systems to reduce cost of ownership
- Ability to start e-business with small Linux/Intel servers, and scale up to high performance IBM @server pSeries and RS/6000 based systems
- Allows the companies to utilize familiar Linux application development tools
- Gives companies more flexibility in choosing applications that are best for their needs
There is some Open Source Software that is currently available for AIX, but many more applications, that have already been recompiled for use with AIX, come with the Toolbox. In some cases, concurrent use of this software and the Toolbox software will produce potential conflicts (caused by the use of executable search paths and library paths). For more details on this subject, see Appendix C, “Other Open Source Software for AIX” on page 193.

The AIX Toolbox for Linux Applications contains a wide variety of software, including, but not limited to, the following:

- Application development: gcc, g++, gdb, rpm, cvs, automake, autoconf, libtool, bison, flex, and gettext
- Desktop environments: GNOME and KDE
- GNU-based utilities: gawk, m4, indent, sed, tar, diffutils, fileutils, findutils, textutils, grep, and sh-utils
- Programming languages: guile, python, tcl/tk, rep-gtk, and C and C++ compilers
- System utilities: emacs, vim, bzip2, gzip, git, ncftp, rsync, wget, lsof, less, samba, zip, unzip, and zoo
- Graphics applications: ImageMagick, transfig, xfig, xpdf, ghostscript, gv, and mpage
- Libraries: ncurses, readline, libtiff, libpng, libjpeg, slang, fnlib, db, gtk+, and Qt
- System shells: bash2, tcsh, and zsh
- Window managers: enlightenment and sawfish

### 2.2 Additional information

The AIX Toolbox for Linux Applications contains a collection of open source and GNU software built for AIX Version 4.3.3 and AIX 5L for IBM pSeries systems and IBM RS/6000 systems. These tools provide the basis of the development environment of choice for many Linux application developers.

For additional information, please refer to the official AIX Toolbox for Linux Applications Web site at:


You can also refer to the AIX Toolbox ReadMe file at:

2.3 Design of the Toolbox

The Toolbox was designed to provide the best performance possible on both AIX Version 4.3.3 as well as AIX 5L. All of the elements of the Toolbox were compiled as native AIX applications, with little or no porting of the original source code. This was done using existing tools, such as autoconf and automake, but the high affinity that AIX already has with Linux in APIs helped.

AIX 5L 5.1 has been enhanced to include more Linux-compatible APIs that were not included in AIX Version 4.3.3. This will add more interoperability between the two operating systems, resulting in a higher degree of Linux application compatibility.

The Toolbox also addresses the issue of continuous development being done on the application, either through enhancements or through fixing of bugs, by porting the GNU tool set along with other open source tools and utilities to AIX. The GNU tools are one of the components of the Toolbox and are needed to recompile Linux applications to run on AIX. These tools allow end users to work on existing applications, as well as develop new applications with a sense of familiarity.

2.3.1 Directory structure

It is recommended that you create a separate file system for the /opt/freeware directory prior to Toolbox installation. The directory will store the software packages you decide to install. The diagram in Figure 2-1 on page 18 and Figure 2-2 on page 18 show the directory structures after the Toolbox installation. In this example, a new file system for /opt/freeware has been created before Toolbox installation.

Important: The GNU tools are governed by the GPL license agreement and their source code is made available by IBM. Non-GNU tools may carry unique licenses.

The licenses associated with the various packages are available for viewing on the Toolbox CD-ROM and on the following Web site:

Figure 2-1  /opt/freeware tree

Figure 2-2  /usr/opt/freeware tree

**Note:** On AIX Version 4.3.3 systems, under certain conditions, where /opt would normally be a part of the root file system and no /opt/freeware file system has been created, /opt/freeware will be created as a symbolic link to /usr/opt/freeware in order to avoid filling the root file system.
When the Toolbox is installed on an AIX system, new directories and files are created and some library links are added. All the packages from the Toolbox will be installed under the /opt/freeware directory. This strategy will prevent any collisions between AIX files and RPMs, which may cause a system failure or software malfunction. Also, RPMs can be easily controlled with these settings.

The following is a description of the /opt/freeware tree on Figure 2-1 on page 18:

- /opt/freeware/bin
  Primary directory of essential binary commands that may be used by both the administrator and users.

- /opt/freeware/etc
  Contains symbolic links to /etc.

- /opt/freeware/info
  GNU Information system’s primary directory.

- /opt/freeware/lib
  Contains shared libraries used by the Toolbox applications. It also contains object libraries, compiler program binaries, and other libraries.

- /opt/freeware/man
  Manual pages. See Figure 2-3 for details.

- /opt/freeware/sbin
  Contains utilities used for system administration.

- /opt/freeware/src
  A link to /usr/opt/freeware/src directory. See Figure 2-4 on page 20.

- /opt/freeware/doc
  Contains miscellaneous documentation.

- /opt/freeware/include
  Contains include files for the Toolbox.
> /opt/freeware/libexec

Contains support programs and libraries for a particular set of programs that are not meant to be executed or linked directly by other applications.

> /opt/freeware/share

Contains architecture-independent files, such as timezone, terminfo information, and so on.

The following is a description of the /opt/freeware tree on Figure 2-1 on page 18:

> /usr/opt/freeware/bin

Primary directory of essential binary commands that may be used by both the administrator and users.

> /usr/opt/freeware/lib

Contains shared libraries used by the Toolbox applications. It also contains object libraries, compiler program binaries, and other libraries.

> /usr/opt/freeware/src

Contains all source codes if the source packages (SRPMs) are installed. See Figure 2-4 for details.

```
   packages
     ├── BUILD Directory for building binary RPMs
     │    └── RPMS Contains binary RPMs after they were built
     ├── SOURCES Contains source tar files and patches
     │    └── SPECS Contains all the spec files
     │         └── SRPMS Contains all source RPMs after they were built
```

*Figure 2-4 /usr/opt/freeware/src tree*

### 2.3.2 System variables

The Linux binaries and libraries installed from the Toolbox will be placed in the /opt/freeware/bin and /opt/freeware/lib directories with links being added to /usr/bin, /usr/linux/bin, /usr/lib, and /usr/linux/lib. This structure is set up in a way that avoids conflicts with AIX binaries and libraries. In effect, no modifications of or user intervention for the Toolbox applications are needed after installing them on AIX. In some cases, where the added Toolbox command has the same name as an AIX command, then no links are provided in /usr/bin, but are provided in /usr/linux/bin. To execute the Linux version of the command, you can either call it with its complete relative path, or change the PATH variable to have /usr/linux/bin in the beginning of the PATH. For example:
To access the man pages of the installed Toolbox applications, add 
/opt/freeware/man to your MANPATH variable. A MANPATH variable tells the 
current shell where to obtain information to the commands. To do this, use the 
following command:

```
# export MANPATH=$MANPATH:/opt/freeware/man
```

### 2.4 Components

Here we discuss the components of the Toolbox.

#### 2.4.1 Development utilities

Most applications that are developed for AIX use the IBM Visual Age compiler, 
while applications developed for Linux more often use the GNU compilers. The 
Toolbox uses both compilers, depending on which one is best for a particular 
application. Also, the spec files included in the source codes are made so it is 
possible to build binary packages using either compiler.

#### 2.4.2 User environment utilities and applications

As part of the AIX Toolbox for Linux applications, GNOME and KDE2 have been 
ported by using the GNU tools, APIs, and header files that are part of the 
Toolbox. (GNOME and KDE are two very popular desktop environments used in 
Linux systems.) GNOME and KDE are sets of user-friendly applications and 
desktop tools that are used in connection with a window manager for the 
XWindow system. Both are similar in concept to CDE\textsuperscript{T\tiny M}, but are fully based on 
Open Source Software. The complexity of these applications highlights the 
capability of AIX to run large, sophisticated Linux applications using the Toolbox. 
Applications that are not included in the Toolbox may be ported to AIX using the 
Toolbox. A complete guide on porting Linux applications on AIX can be found in 
Chapter 5, “Package building and porting” on page 69.
2.4.3 Binaries and sources (rpm and srpm)

A Source RPM (SRPM) does not contain compiled binaries, but instead contains the sources that a binary package can be built from. The SRPM packages in the Toolbox are marked by the file extensions src.rpm. This source package file is an archive that contains the original compressed tar file(s) with source code, patches, and spec file(s).

The binary package file contains all the files that make up the application, along with additional information that is needed to install, upgrade, and erase it. A binary RPM can be installed by using the rpm command without needing to do any recompilations.

SRPMs are important if you want to rebuild an RPM package for whatever reason. Rebuilding a SRPM file does not mean that the package has been or will be installed on the particular system. To actually use the application, you have to install the binary RPM package that was produced during the rebuilding the SRPM. Rebuilding of packages from source is discussed in Section 5.2.2, “Rebuilding a Toolbox RPM” on page 72.

2.5 Installation methods

The installation method for both Linux and AIX is very similar. Both can be done through the command line or through a user interface (SMIT for AIX, GnomeRPM for GNOME desktop, and KPackage for KDE desktop). GnomeRPM and KPackage can only be used after the installation of their corresponding desktop base and applications.

RPM Package Manager maintains a database of all installed packages and their corresponding files. It also stores information on all the packages that are installed or upgraded on the system. The database also reflects the configuration of the system on which it resides; thus, it could easily check if the RPM database has become corrupted or if the system configuration has changed.

On the other hand, AIX stores its installation information in the Object Data Manager (ODM). The ODM is a database intended for storing system information. Information is stored and maintained as objects with associated characteristics. The ODM is also used to manage Vital Product Data (VPD) of application programs for installation and update procedures.
As previously mentioned, the tools and applications that come with the Toolbox are all in RPM format except for the rpm.rte package. In order to install these packages, the RPM Package Manager should be installed first. The RPM Package Manager and a few requisite tools, such as gzip, bzip2, and some patches, are available from the Toolbox installation repository in installp format. This package should be installed using `installp` or SMIT.

While `installp` is installing rpm.rte, a program called `/usr/sbin/updtvpkg` is executed. This will load the RPM database, which contains information on all shared libraries and shells being used by AIX on the current system. This is an important aspect for the RPM Package Manager, because it always relies on shared libraries for all its requisite handling.

A simple installation process of the Toolbox is shown in Figure 2-5.

In case you have decided to install some other AIX package after the installation of the Toolbox, it will be wise to always update or refresh the RPM database (refer to Figure 2-6 on page 24), especially if AIX libraries were added. You can do this by running `/usr/sbin/updtvpkg` manually. When the RPM database is refreshed, it will again gather information on the shared libraries installed in the current system by `installp`. This will prevent installation errors for Linux applications that need these shared libraries.
In a future version of AIX 5L:

- The `lslpp` command will be capable of listing RPM packages installed on the system.
- The `gencopy` command, which is a general version of `bffcreate`, will be able to handle other install formats.
- The `geninstall` command, which is the general install wrapper, will be able to handle `installp`, RPM, and few other formats.

The next section and Section 2.5.2, “RPM Package Manager” on page 30 will help you with some basic concepts of `installp` and the RPM Package Manager.

### 2.5.1 AIX installp

The `installp` command installs and updates software in the AIX operating system. Updates that have been applied to the system can either be committed or rejected at a later time.

An AIX software product installation package is in AIX Backup File Format (.bff) and contains the files of the software product to be installed, the required installation control files, and the optional installation customization files. It contains one or more separate installable, logically-grouped units called filesets. Each fileset in a package must belong to the same product.

The fileset is the lowest installable base unit in AIX. For example, `bos.net.nfs.client.4.3.0.0` is part of the base OS network package. When a base level fileset is installed on the system, it is automatically committed. A fileset update or an update package contains modifications to an existing fileset and has a different fix ID or maintenance level. For example, `bos.net.nfs.client 4.3.0.2` and `bos.net.nfs.client 4.3.3.28` are both fileset updates for `bos.net.nfs.client 4.3.0.0`.

To determine if a fileset is installed on the system, use the command:
# lslpp -L <filesetname>

**Using the installp command**

The basic mode of operations for `installp` are:

- **Apply**
  
  ```
  # installp -a [ -N ] [ -e LogFile ] [ -V Number ] [-d Device ] [ -b ] [ -B ] [ -D ] [ -I ] [ -p ] [ -Q ] [ -q ] [ -v ] [ -X ] [ -F ] [ -g ] [ -O { [ r ] [ s ] [ u ] } ] [ -t SaveDirectory ] [ -w ] [ -z BlockSize ] { FilesetName [ Level ] ... | -f ListFile | all }
  ```

  When a fileset update is applied to the system, the update is installed. The current version of that software is saved in a special directory on the disk so that the new version can be rejected later, if desired. Once a new version of a software product has been applied to the system, that version becomes the currently active version of the software.

- **Commit**
  
  ```
  # installp -c [ -e LogFile ] [ -V Number ] [ -b ] [ -g ] [ -p ] [ -v ] [ -X ] [ -O { [ r ] [ s ] [ u ] } ] [ -w ] { FilesetName [ Level ] ... | -f ListFile | all }
  ```

  When updates are committed with the `-c` flag, the saved files from all previous versions of the software product are removed from the system, thereby saving disk space but making it impossible to return to a previous version of the software product.

- **Reject**
  
  ```
  # installp -r [ -e LogFile ] [ -V Number ] [ -b ] [ -g ] [ -p ] [ -v ] [ -X ] [ -O { [ r ] [ s ] [ u ] } ] [ -w ] { FilesetName [ Level ] ... | -f ListFile }
  ```

  When a software product update is rejected with the `-r` flag, the active version of the software product is changed to the previously installed version. Files saved for the rejected update and any updates that were applied after it are removed from the system.

- **Cleanup**
  
  ```
  # installp -C [ -b ] [ -e LogFile ]
  ```

  If an installation of any application is interrupted and leaves software in a state of either applying or committing, it is necessary to perform cleanup before any further installations will be allowed. An attempt to clean up all products is performed when the `-C` flag is used.
Deinstall

```bash
# installp -u [ -e LogFile ] [ -V Number ] [ -b ] [ -g ] [ -p ] [ -v ] [ -X ] [ -0 [ -r ] [ s ] [ u ] ] [ -w ] { FilesetName [ Level ] ... | -f ListFile }
```

When a base level is removed, the files that are part of the software product and all its updates are removed from the system. Mostly, a cleanup of system configuration and other information pertaining to the product is also done, but this is dependent on the product and may not always be complete.

Table 2-1 gives a summary of some `installp` flags.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-ac</td>
<td>Applies and commits.</td>
</tr>
<tr>
<td>-g</td>
<td>Includes requisites.</td>
</tr>
<tr>
<td>-N</td>
<td>Overrides saving of existing files.</td>
</tr>
<tr>
<td>-q</td>
<td>Quiet mode.</td>
</tr>
<tr>
<td>-w</td>
<td>Does not place a wild card at the end of fileset name.</td>
</tr>
<tr>
<td>-X</td>
<td>Attempts to expand file system type if needed.</td>
</tr>
<tr>
<td>-d</td>
<td>Input device.</td>
</tr>
<tr>
<td>-l</td>
<td>List of installable filesets.</td>
</tr>
<tr>
<td>-c</td>
<td>Commits an applied fileset.</td>
</tr>
<tr>
<td>-C</td>
<td>Cleans up after a failed installation.</td>
</tr>
<tr>
<td>-u</td>
<td>Uninstalls.</td>
</tr>
<tr>
<td>-r</td>
<td>Rejects an applied fileset.</td>
</tr>
<tr>
<td>-p</td>
<td>Preview of installation.</td>
</tr>
<tr>
<td>-e</td>
<td>Defines an installation log.</td>
</tr>
<tr>
<td>-F</td>
<td>Forces overwrite of same and newer version.</td>
</tr>
</tbody>
</table>

Using the Systems Management Interface Tool (SMIT)

Another feature of AIX that distinguishes it from other UNIX-based operating systems is the quality of its administrative tool. This tool is available both as a XWindows application and as a text based menu system. When invoked with the `smit` command, the system will start the GUI version msmit if a $DISPLAY variable is set (refer to Figure 2-8 on page 28); otherwise, it will invoke smitty, the text based version (refer to Figure 2-7 on page 27).
To get to the installation menu, simply type `smit` and choose the Software Installation and Maintenance option. This is also possible using a fastpath, that is, a shortcut method, to display a menu directly from the command line. There is a fastpath for each task/operation, such as managing the devices, security and users, applications, and more. The following shows the fastpath for installation of software:

```
# smit installp
```
The Software Installation and Maintenance menu provides information that you can use for installing and updating software, and other tasks.

The graphical interface for SMIT displays a hierarchy of menus. This was designed to simplify systems management tasks. There are several parts to the SMIT Graphical User Interface:

- **Menu panel**: Lower panel of the primary SMIT panel. The allowable functions will be displayed in the menu bar and a list of menu items appears in the menu panel (refer to Figure 2-8).
- **Path panel**: Top panel of the primary SMIT panel. It shows menus that have been traversed to get to the current menu (refer to Figure 2-8).
- **Dialog panel**: A pop-up menu panel that appears each time a task is selected in the menu panel. This is where you supply details of the task selected (refer to Figure 2-9 on page 29).
Figure 2-9  SMIT dialog panel

- Command output panel: A display associated with the dialog panel when “Do” is selected. The output generated by the command will be displayed on this panel (refer to Figure 2-10 on page 30).
2.5.2 RPM Package Manager

The RPM Package Manager is an open packaging system that can work on Linux systems and other UNIX-based systems. It is easy to use and provides many features for installing, uninstalling, upgrading, and deleting packages using the command line or a Graphical User Interface (GUI). RPM Package Manager makes the process of building a package and distributing the software easy by taking the source code of the software and packaging it into source or binary form.

With the RPM database feature, you can perform queries and verification of the installed RPM packages in your system and determine what package a certain file belongs to. You can search through the entire database for packages or just certain files to get information about the system. Identification of these packages is done using package labels. Each label contains information that uniquely identifies the package. So, even if the package file is renamed, the new file name will not confuse the RPM Package Manager, because the package label is within the context of the file. A sample of the RPM package labelling convention is shown in Figure 2-11 on page 31.
The three components in each package label are:

- **The software name**
  All RPM package labels start with the software name. This may be derived from the application name or a description of the related programs grouped together in one package.

- **The software version**
  This is an identifier that states the version of the packaged software.

- **The package release**
  This is the most specific part of the package label, which shows the number of times the package has been rebuilt with the same software version. Rebuilds are normally done due to bugs uncovered after packaging and use.

### Using RPM in command line

RPM Package Manager has five basic modes of operation. This section contains an overview of each mode. To get the full details and options, refer to the `rpm` man page.

- **Install**

  ```bash
  ```

  RPM turns the installation process into a single command. `rpm -i` installs software that has been packaged into an RPM package file. It does this by going through several steps:

  a. Performing dependency checks
  b. Checking for conflicts
  c. Performing tasks that are required before the install
  d. Deciding what to do with the configuration files
  e. Unpacking files from the package and putting them in their proper places
f. Performing tasks that are required after the install

g. Keeping track of what has been done

- Uninstall

```
# rpm {--erase -e} [--root <dir>] [--noscripts] [--allmatches]
[--notriggers] package1 ... packageN
```

The `rpm -e` command removes, or erases, one or more packages from the system. RPM performs a series of steps whenever it erases a package:

a. It checks the RPM database to make sure that no other packages depend on the package being erased.

b. It executes a pre-uninstall script.

c. It checks to see if any of the package's configuration files have been modified. If so, it saves copies of them.

d. It reviews the RPM database to find every file listed as being part of the package; if they do not belong to another package, it automatically deletes them.

e. It executes a post-uninstall script.

f. It removes all traces of the package and the files belonging to it from the RPM database.

- Upgrade

```
# rpm {--upgrade -U} [-v] [--hash -h] [--percent] [--force] [--test]
[--rcfile <file>] [--ignorearch] [--dbpath <dir>] [--prefix <dir>]
[--justdb] [--noorder] [--relocate oldpath=newpath] [--badreloc]
[--excluderel <path>] [--ignorearch] package1.rpm ... packageN.rpm
```

The `rpm -U` command performs two distinct operations that are reduced to a single command. First, it installs the desired package and automatically uninstalls any older versions of the package. RPM also performs intelligent upgrading of packages with configuration files. It will save the original configuration file if it is not forward compatible with the new configuration file in the package.
Chapter 2. AIX Toolbox for Linux Applications

Query

```bash
{package1...packageN file1...fileN}
```

The `rpm -q` command consists of two distinct parts: package selection (displays packages contained in the query) and information selection (filters/displays information based on set parameters).

Verify

```bash
# rpm {--verify -V -y} [-afpg] [---root <dir>] [---nofiles] [---noscripts]
[---nomd5] {package1.rpm...packageN.rpm}
```

The `rpm -V` command verifies an installed package. It also checks for package dependencies on other packages. A file can be verified and checked for many different attributes, such as owner, group, size, and modification time.

Using GnomeRPM

One of the most convenient package manipulation tools available is GnomeRPM. This is a graphical tool that runs (typically) under the GNOME Desktop. It is also referred to as GnoRPM. The beauty of this tool is that it allows the end user to easily work with RPM technology through a user-friendly graphical interface. GnomeRPM is "GNOME-compliant," which means that it completely integrates into the GNOME desktop environment.

**Note:** This tool will only be available subsequent to the installation of the GNOME Desktop Applications package included in the Toolbox.

Operations are performed in GnomeRPM by finding and selecting packages, and then choosing the type of operation to perform by using a push-button on the toolbar (through a menu) or by right-clicking the mouse.

Using GnomeRPM to perform package operations provides all the functionality as if using RPM from the shell prompt. However, the graphical interface often makes these operations easier to perform and offers some additional functionality, such as transparent access to packages over the Internet. To open the GnomeRPM panel, click on the start button of the GNOME Desktop Toolbar and select Programs -> System -> GnomeRPM.

The interface features a menu, a toolbar, a tree, and a display panel of currently installed packages (refer to Figure 2-12 on page 34):

- **Package panel:** Located on the left side. It allows the user to browse and select packages on the system.
Display panel: Located at the right of the package panel. It shows the contents of the folders in the panel.

Toolbar: Located above the display panel and package panel. It is a graphical display of the package tools.

Menu: Located above the toolbar. It contains text-based commands, as preferences and other settings, as well as help information.

Status bar: Located below the display panel and package panel. It shows the total number of selected packages.

The normal way to handle GnoRPM is to display the available packages, select the package(s) you want to operate on, and then select an option from the toolbar or menu to perform an operation/task. For example, you can install, upgrade or uninstall several packages with a few button clicks. Similarly, you can query and verify more than one package at a time. See “GnoRPM” on page 105 for more details on this subject.
Using KPackage

KPackage is another GUI Interface tool for the RPM Package Manager. It is designed to integrate with the KDE desktop. KPackage can also be used for Debian, BSD and Slackware® package managers. This tool (typically) runs under the KDE Desktop Environment.

**Note:** This tool will only be available subsequent to the installation of the KDE Desktop Applications package included in the Toolbox.

KPackage makes use of the KDE Drag and Drop protocol. This means that you can easily drag and drop packages onto the KPackage panel to open them or make a query. Another feature of the KPackage is the Find File dialog, where you can drag and drop a file and search for the package the file belongs to.

To start KPackage, click on the Start button on the KDE Desktop Toolbar and select **Utilities -> Package Manager**.

There are several parts to the KPackage interface (refer to Figure 2-13 on page 36):

- **Package tree:** Located on the left side of the panel. It shows all the installed, uninstalled, and updated packages with their package name, size, and version.
- **Information panel:** Located to the right of the package tree. It displays the status information on the package and all the files included on it. It also has an Install and Uninstall button to easily install and uninstall a selected package.
- **Menu:** Located above the package tree and information panel. It contains other options, settings, and preferences that can be used to accomplish a certain task, as well as provide help information.
- **Toolbar:** Located to the left of the package tree. It is a graphical display of the package tools.

See Section 6.1.4, “Package managing using KDE or GNOME” on page 104 for more information on this subject.
Figure 2-13   Main KPackage panel
Toolbox installation

This chapter describes the installation of the Toolbox in detail, to the point of defining GNOME or KDE as the standard desktop environment. In the first section, we summarize the prerequisites for the Toolbox for AIX 5L and AIX Version 4.3.3. The next section describes the step-by-step installation of the Toolbox for AIX 5L and AIX Version 4.3.3. Some hints and tips for troubleshooting are given. The final section mentions some useful URLs.

The latest information and changes to the installation procedure, as well as solutions to problems encountered during the installation, can be found at:


Please check this Web site should there be any questions or problems with the installation. You can also download the latest versions of the Toolbox and all its packages at this URL. You can also use ftp and connect to the host ftp.software.ibm.com and go to the directory /aix/freeSoftware/aixtoolbox.

A listing of the packages by date can be found at:

3.1 System requirements

The installation of the Toolbox requires AIX Version 4.3.3 or newer. You can use either AIX 5L without additional PTFs or AIX Version 4.3.3 with the fixes from APAR IY15017. You can download them from:

http://techsupport.services.ibm.com/rs6000/support

and then search for IY15017 in the APAR database. Follow the instructions for downloading and installing the fixes. You should also check for the latest available maintenance level at:

http://techsupport.services.ibm.com/support/rs6000.support/downloads

To separate the files belonging to the Toolbox from the base operating system and other installed software, it is recommended that you create a new file system (/opt/freeware) at this time. A rough guideline for disk space requirements in this file system is given in Table 3-1. As new packages are continually added into the Toolbox, the disk space requirement to install it all may grow.

Table 3-1  Disk space requirements for the components of the Toolbox

<table>
<thead>
<tr>
<th>ezinstall group</th>
<th>Required disk space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Linux Affinity Support</td>
<td>10 MB (in /usr)</td>
</tr>
<tr>
<td>Common Support Programs for GNOME and KDE</td>
<td>14 MB</td>
</tr>
<tr>
<td>GNOME Desktop Base</td>
<td>75 MB</td>
</tr>
<tr>
<td>GNOME Desktop Applications</td>
<td>75 MB</td>
</tr>
<tr>
<td>KDE Desktop Base</td>
<td>160 MB</td>
</tr>
<tr>
<td>KDE Desktop Applications</td>
<td>75 MB</td>
</tr>
<tr>
<td>Total</td>
<td>400 - 500 MB</td>
</tr>
<tr>
<td>GNUpro Toolkit (gcc compiler and so on)</td>
<td>200 MB</td>
</tr>
</tbody>
</table>

If you want to use the GNOME or KDE desktops, you need to make sure the following AIX products are installed: X11.adt.lib, X11.apps.xdm, and X11.samples.apps.clients. This can be checked with the `lslpp` command:

```
# lslpp -L X11.adt.lib X11.apps.xdm X11.samples.apps.clients
```
If one or more of them are not installed on your system, install them from the AIX system installation media.

If you want to use the gcc compiler, as described in Chapter 5, “Package building and porting” on page 69, you need to have some header files and other tools installed. It is recommended that you install the complete bos.adt and X11.adt filesets, although a subset might be enough in some cases.

We tested the procedures described below to install the Toolbox using the version available in February 2001 on both AIX Version 4.3.3 and a prerelease of AIX 5L 5.1. Only the basic operating system had previously been installed on these systems (based on RS/6000 F50 hardware, 2-way, 512 MB RAM).

3.2 Installation procedure

This section describes the step-by-step installation of the Toolbox and gives hints for troubleshooting.

3.2.1 Installing the RPM Package Manager

As described in Chapter 2, “AIX Toolbox for Linux Applications” on page 13, the core of the Toolbox is the RPM Package Manager (RPM). It is needed to install all the other available Toolbox software. Since RPM is not available before it has been installed, we have to install RPM by traditional AIX means, which includes installp, SMIT or WebSM.

The easiest way to install RPM is to change to the directory where the rpm.rte image is located (for example, /cdrom/installp/ppc) and use the following command:

```
# installp -qacXgd rpm.rte rpm.rte
```

or install it using SMIT or WebSM. The rpm.rte image can be downloaded from:


using binary download type and user ID ftp.

The installation will take some time, because it needs to gather information about the shared libraries already installed on the system (see Section 2.5, “Installation methods” on page 22 for more details).

It is not necessary to change the environment at this point. The newly installed binaries have a link to /usr/bin (as long as there are no conflicts with the AIX commands in /usr/bin; otherwise, there will be a link to /usr/linux/bin) and can be found by the shell by using the standard PATH. Section 2.3.1, "Directory
structure” on page 17, describes in more detail where libraries and header files are located. If you wish to access man pages of the Toolbox software, you should add /opt/freeware/man to the environment variable MANPATH using the following command:

```bash
# export MANPATH=$MANPATH:/opt/freeware/man
```

After having installed RPM successfully, the other software of the Toolbox can be installed using the RPM Package Manager. You can continue with downloading prebuilt RPMs from the Toolbox Web site, such as the GNOME and KDE desktops, application development tools, GNU tools or other tools, programs, and libraries.

For compiling new software from sources, you would have to install the appropriate application development tools (for example, gcc, g++, automake, autoconf, bison, and libtool). This process is described in Chapter 5, “Package building and porting” on page 69.

### 3.2.2 Preparing to install GNOME, KDE2 and other applications

To facilitate installation of the various programs included in the Toolbox, some of them are classified into installation groups, such as:

- Base Linux Affinity Support
- Common Support Programs for GNOME and KDE
- GNOME Desktop Base
- GNOME Desktop Applications
- KDE Desktop Base
- KDE Desktop Applications

For more information on this subject, see:


You can install all the packages pertaining to the group you want to install with just one command. If you do not already have the install images, you might want to create new directories to hold the installation files of each group instead of storing all the RPMs in a single directory. Enter the following three command lines, one after the other, pressing Enter after entering each of them; the system will create the directories after you press Enter for third time:

```bash
# for dir in base desktop.base gnome.base gnome.apps kde.base kde.opt;
do mkdir -p ezinstall/$dir || : ;
done
```

Now download the files belonging to the individual ezinstall groups into the just created directories. The next section about FTP tools might be able to help you with this task.
3.2.3 FTP tools

To download the images, we recommend that you install the ncftp or wget packages first. If you already have all the packages you plan to install, this step can be omitted.

wget is a command line tool that retrieves and recursively downloads files from the Web using the HTTP or FTP protocols (see http://www.wget.org/ for more information).

ncftp is a really nice replacement for the common ftp command, because it has a lot of usability enhancements (see http://www.ncftp.com/ncftp/ for more information).

Two prerequisite packages have to be installed for wget and ncftp. Change to the directory where all RPMs are located (for example, /cdrom/RPMS/ppc) and install the bash and info packages using (specify the current version of these packages, should you have newer ones) the command:

# rpm -ivh bash2-2.04-3.aix4.3.ppc.rpm info-4.0-5.aix4.3.ppc.rpm

Both packages can also be found at the following FTP sites:

wget

If you decide to use wget and do not have the corresponding RPM package yet, download it from:

using the user ID ftp. By issuing the command:

# rpm -ivh wget<version>.rpm

You can install it into the directory where the wget RPM file is located on your system. wget is now installed and you can recursively download the other packages needed.

Attention: In order to download all RPMs out of the Toolbox, 310 MB was necessary. Make sure you have enough disk space and network bandwidth. As more images are added in the Toolbox, this number is constantly growing.

For example, to download all RPMs out of the Toolbox Web site, you would issue the following command:

In case you have to use a ftp proxy, you have to set the environment variable 
ftp_proxy first by using the command:

```bash
# export ftp_proxy=http://your.proxy:port/
```

**ncftp**

You can download the ncftp RPM from:


By issuing the command:

```bash
# rpm -ivh ncftp<version>.rpm
```

you can install it into the directory where the ncftp RPM file is located on your 
system. You can start ncftp with the command:

```bash
```

You can use standard **ftp** commands like **dir** or **get** (with automatic **reget**, in 
case the connection should end unexpectedly and you need to issue the **get** 
command again). ncftp also offers enhancements like word completion (press 
the Tab key once or twice and watch what happens) and retrieval of whole 
directory trees with **get -R**. To get all the RPMs out of the Toolbox, you would 
use the command:

```bash
ncftp> get -R RPMS
```

after the initial **ncftp** command. To accomplish this task more quickly, use the 
command:

```bash
```

Example 3-1 shows a sample ncftp session.

```
Example 3-1   ncftp session
NcFTP 3.0.1 (March 27, 2000) by Mike Gleason (ncftp@ncftp.com).
Copyright (c) 1992-2000 by Mike Gleason. 
All rights reserved.

Connecting to 207.25.253.26... 
2000) ready. 
Logging in... 
Please read the file README 
it was last modified on Fri Jul 28 10:27:42 2000 - 199 days ago 
Guest login ok, access restrictions apply. 
Current remote directory is /aix/freeSoftware/aixtoolbox.
```
3.2.4 Installing the Toolbox base

We can now install the base packages of the Toolbox. These are basic utilities, such as info, gzip, bash, patch, tar, bzip2, unzip, gettext, zip, bash and so on. They can be installed from the base ezinstall directory using the command:

```bash
# rpm -ivh *
```

or from a directory containing all RPMs by using the command:

```bash
# rpm -ivh info-* bzip2* gett* gzip* patch* popt* rpm* tar* unzi* zip* bash2*
```

**Note:** Make sure you also install the info package, as this is a prerequisite for most other RPMs. Do not specify any packages as parameters on the command line that are already installed on the system (for example, bash and info).

To check what packages are installed on the system, use the following command:

```bash
# rpm -qa
```

If nothing went wrong, you can continue with the installation of the ezinstall groups you would like to use (such as GNOME and KDE) or other packages. Just in case something did not go smoothly, we will discuss the usage of the RPM Package Manager and give some hints for troubleshooting during the installation in the next section.
3.2.5 Using the RPM Package Manager

RPM is the RPM Package Manager. It allows users to take the source code for new software and package it into source and binary form so that binaries can be easily installed and tracked and the source can be easily rebuilt. It also maintains a database of all packages and their files that can be used for:

- Verifying packages
- Querying for information about files and/or packages

After using the RPM to successfully install packages, we now want to look in greater detail on how to use and troubleshoot this tool. The basic options for RPM are:

- -i: install
- -e: erase
- -q: query
- -v: verbose
- -V: verify
- --help

See Section 2.5.2, “RPM Package Manager” on page 30 for more information on this subject.

Information about RPM can be found at:
http://www.rpm.org/

The links in the documentation section there, especially the RPM-HOWTO (the section of the RedHat reference manual on using RPM and the softcopy of Maximum RPM by Ed Bailey, found at http://www.rpm.org/maximum-rpm.ps.gz or http://www.rpm.org/rpmbook/) will answer every question about RPM and its usage.

Let us look at some examples of RPM usage, what problems might arise, and how to solve them.

Searching for files in a set of RPMs

If multiple packages are to be installed with only one call of RPM, and one of the packages to be installed has an unresolved dependency, none of the packages will be installed. RPM will give some error messages indicating missing files or packages. In Example 3-2, only missing files are noted. In this case, it is harder to find out where to get the files from than it is if the error message tells you the names of missing packages, which might also occur.

Example 3-2   Installation of multiple RPM files

```
# rpm -iv bzi* gett* gzip* patch* popt* rpm* tar* unzi* zip*
```
error: failed dependencies:
/sbin/install-info is needed by gzip-1.2.4a-3
/sbin/install-info is needed by tar-1.13-2

To find the packages containing the missing files, we used the query option of RPM (rpm -q). The query option has some additional sub-options (see Section 2.5.2, “RPM Package Manager” on page 30 for more information):

- `-a` queries all currently installed packages.
- `-f <file>` will query the package owning `<file>`.
- `-p <packagefile>` queries the package `<packagefile>`.
- `-i` displays package information.
- `-l` displays the list of files that the package contains.
- `-s` displays the state of all the files in the package.
- `-d` displays a list of files marked as documentation.
- `-c` displays a list of files marked as configuration files.

To check the names of all packages installed in the system, use `rpm -qa`, as shown in Example 3-3.

Example 3-3   Checking the names of installed packages
# rpm -qa
SysProvides-5.1.0.0-1
bash2-2.04-3
bash2-doc-2.04-3

Note: The name of the package SysProvides has been changed. The new name of the package is AIX-rpm.

To find the package file that contains the missing file install-info from the installation example above, you can use the search facility at:
http://rpmfind.net/

to deduce what package might contain the missing file install-info. You can also issue the following command while in a directory that contains the RPMs you want to be searched:

# for f in *.rpm; do (rpm -qlp $f |grep install-info && echo $f) ; done

After finding the file, install the RPM that was missing and reissue the installation command that failed in the first run.
Running out of disk space

Let us assume an installation command failed because we were running out of space in the file system. This can happen because unlike `installp`, RPM cannot automatically increase the size of a file system during the install. The error message might look like the one in Example 3-4.

Example 3-4   Installation attempt

```
# rpm -ivh *
ORBit-0.5.1-2
control-center-1.2.0-2
gdbm-1.8.0-3
gdk-pixbuf-0.8.0-2
glib-1.2.8-3
gnome-core-1.2.1-2
gnome-libs-1.2.0-5
gtk+-1.2.8-2
imlib-1.9.8.1-4
libglade-0.13-2
librep-0.12.4-2
libxml-1.8.7-2
rep-gtk-0.13a-1
rep-gtk-gnome-0.13a-1
unpacking of archive failed on file
/opt/freeware/libexec/rep/rs6000-ibm-aix4.3.3.0/libgdk-pixbuf.so.0.0.0: cpio:
copy failed - There is not enough space in the file system.
rep-gtk-libglade-0.13a-1
unpacking of archive failed on file
/opt/freeware/libexec/rep/rs6000-ibm-aix4.3.3.0/libglade.so.0.0.0: cpio: copy
copy failed - There is not enough space in the file system.
sawfish-0.30.3-1
unpacking of archive failed on file /opt/freeware/bin/sawfish: cpio: copy
copy failed - There is not enough space in the file system.
sawfish-themer-0.30.3-1
unpacking of archive failed on file /opt/freeware/bin/sawfish-themer: cpio:
copy failed - There is not enough space in the file system.
```

If we enlarge the file system and try to issue the same command again, it fails, because some of the RPMs to be installed are already installed. RPM does not attempt to install any of the packages. Unfortunately, the error messages do not explicitly say that the install failed, only which files are already installed. See the Example 3-5 for more details.

Example 3-5   Reissuing installation

```
# rpm -ivh *
package ORBit-0.5.1-2 is already installed
package control-center-1.2.0-2 is already installed
package gdbm-1.8.0-3 is already installed
```
package gdk-pixbuf-0.8.0-2 is already installed
package glib-1.2.8-3 is already installed
package gnome-core-1.2.1-2 is already installed
package gnome-libs-1.2.0-5 is already installed
package gtk+-1.2.8-2 is already installed
package imlib-1.9.8.1-4 is already installed
package libglade-0.13-2 is already installed
package librep-0.12.4-2 is already installed
package libxml-1.8.7-2 is already installed
package rep-gtk-0.13a-1 is already installed

As there is no option of rpm that will allow you to easily get around this, you might think about uninstalling the recently installed packages and then reinstalling the complete set. But in order to deinstall packages, you have to distinguish between a package label and the corresponding package file. Giving the names of the package files as arguments to RPM in order to have RPM deinstall them does not work. RPM needs the package labels and not the names of the package files. Example 3-6 shows what happens if you use the names of package files (compare with Example 3-5 on page 46).

Example 3-6 Using file names to deinstall packages

```
# rpm -e *
error: package ORBit-0.5.1-2.aix4.3.ppc.rpm is not installed
error: package control-center-1.2.0-2.aix4.3.ppc.rpm is not installed
error: package gdbm-1.8.0-3.aix4.3.ppc.rpm is not installed
error: package gdk-pixbuf-0.8.0-2.aix4.3.ppc.rpm is not installed
error: package glib-1.2.8-3.aix4.3.ppc.rpm is not installed
error: package gnome-core-1.2.1-2.aix4.3.ppc.rpm is not installed
error: package gnome-libs-1.2.0-5.aix4.3.ppc.rpm is not installed
error: package gtk+-1.2.8-2.aix4.3.ppc.rpm is not installed
error: package imlib-1.9.8.1-4.aix4.3.ppc.rpm is not installed
error: package libglade-0.13-2.aix4.3.ppc.rpm is not installed
error: package librep-0.12.4-2.aix4.3.ppc.rpm is not installed
error: package libxml-1.8.7-2.aix4.3.ppc.rpm is not installed
error: package rep-gtk-0.13a-1.aix4.3.ppc.rpm is not installed
error: package rep-gtk-gnome-0.13a-1.aix4.3.ppc.rpm is not installed
error: package rep-gtk-libglade-0.13a-1.aix4.3.ppc.rpm is not installed
error: package sawfish-0.30.3-1.aix4.3.ppc.rpm is not installed
error: package sawfish-themer-0.30.3-1.aix4.3.ppc.rpm is not installed
```

To deinstall the packages, you can use the command rpm -e or (preferably) execute the destroyRPMS script provided by the Toolbox Web site at:

The destroyRPMS script, shown in the Example 3-7, removes all RPMS installed on the system and associated images. It also removes root configuration for GNOME and KDE. You should exit KDE and GNOME before running the destroyRPMS script.

Example 3-7   Removing all packages by using destroyRPMS script

# destroyRPMS
This script removes all RPMS installed on the system, removes the RPM database, and removes the rpm.rte install image. It also removes root configuration for GNOME and KDE. You should exit KDE and GNOME and run this from a root console.

Proceed to destroy all RPMs on the system? y/(n) y

Subsequently, you can redo the installation that failed because of disk space shortage. You try to install the packages one by one (automated, of course) with the command:

# ls *.rpm | xargs -n 1 rpm -ivh

But because a package might depend on a package installed later in this cycle, you have to repeat this command until all packages are reported to be already installed. See Example 3-8 for more details.

Example 3-8   Reinstalling

# ls
ghostscript-5.50-2.aix4.3.ppc.rpm
ghostscript-fonts-6.0-1.aix4.3.noarch.rpm
gnome-print-0.20-2.aix4.3.ppc.rpm
gnome-utils-1.2.0-2.aix4.3.ppc.rpm
gnumeric-0.54-2.aix4.3.ppc.rpm
# rpm -ivh *
package gnome-utils-1.2.0-2 is already installed
# ls *.rpm | xargs -n 1 rpm -ivh
error: failed dependencies:
  ghostscript-fonts is needed by ghostscript-5.50-2
ghostscript-fonts-6.0-1
gnome-print-0.20-2
package gnome-utils-1.2.0-2 is already installed
gnumeric-0.54-2
# ls *.rpm | xargs -n 1 rpm -ivh
ghostscript-5.50-2
package ghostscript-fonts-6.0-1 is already installed
gnome-print-0.20-2 is already installed
gnumeric-0.54-2 is already installed
Corrupt package files

Another likely reason for an RPM install to fail is the presence of a corrupt package file. Example 3-9 shows the initial (not very meaningful) error message that results from having a corrupt package file, and how to get more information about what caused the error by using `rpm -ivv` instead of `rpm -iv`. The important lines are the two which show differing package sizes.

Example 3-9  Installation attempt

```
# rpm -ivh kdebase
error: kdebase-2.0.1-4.aix4.3.ppc.rpm cannot be installed
# rpm -ivv kdebase
D: counting packages to install
D: found 1 packages
D: looking for packages to download
D: retrieved 0 packages
D: New Header signature
D: Signature size: 68
D: Signature pad : 4
D: sigsize         : 72
D: Header + Archive: 21639000
D: expected size   : 28631180
error: kdebase-2.0.1-4.aix4.3.ppc.rpm cannot be installed
D: found 0 source and 0 binary packages
```

Getting individual files out of a package file

In the next example, we try to isolate a single file out of an RPM package file. We do not want to install the whole package for this purpose, so we use the `rpm2cpio` command. Let us try to get a file named `sample.xinitrc` out of one of the KDE packages. First, we have to find the package that contains the file, as shown in Example 3-10.

Example 3-10  Looking for a file inside an RPM package

```
# for f in k*; do echo $f; rpm -qlp $f | grep xinitrc; done
kdeadmin-1.1.2-1.aix4.3.ppc.rpm
kdebase-1.1.2-34.aix4.3.ppc.rpm
/opt/freeware/kde/share/apps/kdm/sample/xinitrc
kdegames-1.1.2-3.aix4.3.ppc.rpm
```

We have to install the cpio package from the Toolbox because `rpm2cpio` and the standard AIX `cpio` command are incompatible. This installation is shown in Example 3-11 on page 50 together with the actual `rpm2cpio` commands to find the exact file name and then extract the file. Subsequently, the `sample.xinitrc` can be found in the subdirectory `opt/freeware/kde/share/apps/kdm/`. 
Example 3-11 Installing cpio with rpm2cpio

```
# rpm -ivh cpio-2.4.2-17.aix4.3.ppc.rpm
# rpm2cpio kdebase-1.1.2-34.aix4.3.ppc.rpm | /usr/linux/bin/cpio -ivt | grep \  
  xinitrc
-rw-r--r--  1 root     system       1022 Nov  2 11:39  
  opt/freeware/kde/share/apps/kdm/sample.xinitrc
87759 blocks
# rpm2cpio kdebase-1.1.2-34.aix4.3.ppc.rpm | /usr/linux/bin/cpio -ivd \  
  opt/freeware/kde/share/apps/kdm/sample.xinitrc
```

The last RPM option we want to look at is the verify option. The command:

```
# rpm -Va
```

allows us to verify all the installed packages, while:

```
# rpm -Vf <filename>
```

verifies the package containing the file <filename>.

### 3.2.6 Installing KDE2

In this section, we will describe how to install KDE2 and define it as the default desktop environment, including using the kdm login manager instead of the standard dtlogin. We will not discuss KDE1.

The necessary packages for installing KDE2 are located in three group directories: desktop.base, kde.base and kde.opt (the last two are also consolidated in kde.all). Make sure your system satisfies the requirements mentioned in Section 3.1, “System requirements” on page 38. Disk space and the X11 components are especially important. Make also sure you use the appropriate directories for KDE2, not KDE1.

**Note:** If you had KDE1 installed, you should delete it before attempting to install KDE2. Remove all packages belonging to KDE1, including the Qt library, and remove the .kde directory in your home directory.

**Note:** An earlier version of kdebase-2 did not include a sample.xinitrc file. This had to be retrieved, as described in Section 3.2.5, “Using the RPM Package Manager” on page 44.

To install the desktop.base group (containing important libraries for both the KDE and GNOME desktop), change to the corresponding ezinstall directory and issue the following command:
# rpm -ivh *

**Note:** The package gdbm should be included in this group.

The packages kde.base and kde.opt or kde.all have to be installed in the same way. Only a few steps have to be completed before KDE can be used on a RS/6000 running AIX 5L.

**Note:** The package kdelibs-sound should be included in kde.base and kde.all.

Copy the sample.xinitrc file as $HOME/.xinitrc to your home directory (to each users’ home directory, to be precise). Make sure the KDEDIR variable in this file points to the correct directory (probably KDEDIR=/opt/freeware/kde).

If you want to keep CDE’s dtlogin as the login manager and start KDE manually, you have to choose the Command Line Login option at the dtlogin panel. After having logged in, issue the command:

```bash
# xinit -- -T
```

and KDE2 will start.

For more information about how to use KDE and its advantages, see Section 6.1.2, “The KDE desktop” on page 96. For instructions on how to set up a convenient user environment, see Section 6.2, “Available shells” on page 112.

**Make KDE2 start at each system restart**

If you complete this test successfully and want to have KDE2 automatically started at each system restart and use kdm as your default login manager, replace the standard /usr/lib/X11/xdm/Xsession file with the one that can be found at:


Save the old one before you install the new one with the commands:

```bash
# mv /usr/lib/X11/xdm/Xsession /usr/lib/X11/xdm/Xsession.old
# cp Xsession.kde2 /usr/lib/X11/xdm/Xsession
# chmod 755 /usr/lib/X11/xdm/Xsession
# echo "'/opt/freeware/kde/bin/kdmdesktop" >> /usr/lib/X11/xdm/Xsetup_0
```

Replace the dtlogin entry dt:2:wait:/etc/rc.dt in /etc/inittab with the appropriate entry kdm:2:once:/opt/freeware/kde/bin/kdm to have kdm automatically started after system restart. (Do not try to comment this line; replace it after saving the original file.). If you decide to install GNOME as well, this setup is sufficient to
allow you to choose CDE, KDE, or GNOME from the kdm login menu as the
desktop of choice for each particular login session. Changes to the kdm login
menu can be made in the files /opt/freeware/kde/share/config/kdmrc and
$HOME/.xinitrc.

Attention: Editing /etc/inittab should be done carefully. If this file is corrupted,
the system might not reboot, and system recovery would have to be
performed. Therefore, instead of editing /etc/inittab using an editor, we
recommend that you use of the following commands:

```
# cp /etc/inittab /etc/inittab.old
# mkitab -i dt kdm:2:once:/opt/freeware/kde/bin/kdm
# rmitab dt
```

3.2.7 Installing GNOME

In this section, we will describe how to install GNOME and define it as the default
desktop using the xdm login manager or using kdm from the KDE2 package.

If you did not install KDE2, you first have to install the desktop.base ezinstall
group by using `rpm -ivh *` in the corresponding directory, as described in
Section 3.2.6, “Installing KDE2” on page 50. Remember to exclude the
gbm-1.8.0-3 package from gnome.base group before installation, if you had
already installed KDE or GNOME before.

Now you may install the gnome.base and gnome.apps groups.

Note: Please note that we did not check all possible combinations of installed
software and order of installation. You may change the order and choice of
software, but then the ezinstall groups might not contain all the required
packages.

Each user who wants to run GNOME needs to have a modified .xinitrc file in the
home directory (this assumes that you do not want to change the system wide
default /usr/lpp/X11/defaults/xinitrc). If a .xinitrc file already exists in $HOME,
save it. If it did not exist, copy the one from /usr/lpp/X11/defaults/xinitrc to
$HOME/.xinitrc (watch the `:` in the second file name). Edit the file
$HOME/.xinitrc and replace the last three lines shown in Example 3-12 with the
single line

```
exec /usr/bin/gnome-session
```

Example 3-12 Lines to be edited in .xinitrc

```
xsetroot -solid grey60
aixterm =80x25+0-0 &
```
exec mwm -multiscreen -xrm "ShowFeedback: -quit"

If you want to start GNOME manually, proceed as you did for KDE2 (see Section 3.2.6, “Installing KDE2” on page 50) by choosing the Command Line Login from the dtlogin menu and type:

# xinit -- -T

after you have logged in.

If you want to start GNOME automatically from the KDE kdm login manager, you should not have to do anything, provided you installed kdm using the instructions in Section 3.2.6, “Installing KDE2” on page 50.

For more information about how to use GNOME and its advantages, see Section 6.1.3, “The GNOME desktop” on page 102. For instructions on how to setup a convenient user environment, see Section 6.2, “Available shells” on page 112.

If you tested GNOME successfully and want to use the third option, using the (old) xdm login manager to login and start GNOME, you should first save the existing /etc/inittab file and then call:

# /usr/lib/X11/xdm/xdmconf

to change /etc/inittab, /etc/rc.tcpip, and /etc/tcpip.clean, in order to start xdm instead of dtlogin. A man page for xdmconf is available and can be viewed using the command:

# nroff -man /usr/lib/X11/xdm/xdmconf.man | more

The xdmconf utility only works if the configuration files are close to the initial configuration. It does not work if /etc/inittab has already been changed to start kdm, as described in Section 3.2.6, “Installing KDE2” on page 50. Running xdmconf with the -d option restores the previous configuration. Changes to /etc/inittab require a reboot to take effect and should be done very carefully.

### 3.3 Useful URLs

In this last section of this chapter, we present some URLs that give additional information about the Toolbox and associated software.

**AIX Toolbox for Linux applications:**

AIX Toolbox for Linux applications README file with latest information for installation and configuration:

AIX Toolbox for Linux applications licensing information:

AIX Toolbox for Linux applications download pages:

AIX Toolbox for Linux applications FTP site:

RPM Package Manager homepage:
http://www.rpm.org/

RPM Documentation Project:
http://www.rpm.org/

The Official Red Hat Linux Reference Guide, Chapter 5 about RPM:

RPM HOWTO:
http://www.rpm.org/RPM-HOWTO/index.html

Maximum RPM book, Ed Bailey, found at:
http://www.rpm.org/maximum-rpm.ps.gz

or
http://www.rpm.org/rpmbook/
Source compatibility: Linux-compatible APIs on AIX

This chapter describes the similarities and differences of Linux and AIX in terms of APIs. It is intended to be an aid to application developers porting code from Linux to AIX or writing code on Linux that is intended for deployment on AIX. The chapter also gives general guidelines for writing portable code and what APIs have been added or changed in AIX 5L to make it even more source compatible with Linux.

This chapter provides the following:

- How to write portable code
- Linux-compatible APIs and LSB (Linux Standard Base) subroutines on AIX
- File macro supported values
- Signal values
4.1 Writing portable code

We begin this chapter with a section on application development, with a special focus on writing portable code. The considerations mentioned here mainly apply to situations where code is written for Linux (or will be written) and will be ported to AIX, particularly using the AIX Toolbox for Linux Applications.

As we see in Chapter 5, “Package building and porting” on page 69, many applications can be recompiled and RPM packages rebuilt without any (or minor) changes to the source code. The following pages will give you some guidelines on how to write such portable code in C or C++.

If there are no coding guidelines defined in your coding project, the GNU coding standards (see http://www.gnu.org/prep/standards_toc.html and Section 5.4, “Using libtool to handle shared libraries” on page 82 for details) would be a good start to ensure a consistent debugging and build environment (compilers, make files, library management, and so on), which is available on many different platforms. GNU’s Autoconf (refer to http://www.gnu.org/software/autoconf/) is a major building block in this framework to achieve portability to most UNIX-based systems.

Since Linux applications will probably be written using the GNU compilers, and the same compilers are available with the Toolbox on AIX, there will not be any language-specific errors in the code. In general, this is also true for ANSI C compatible code. But there might be errors because of missing and/or incompatible APIs. Section 4.2, “Linux-compatible APIs and LSB functions on AIX” on page 57 describes which APIs exist in Linux that are either not available under AIX or incompatible with AIX. In “New APIs in AIX 5L 5.1” on page 171, you will find a discussion of APIs that have recently been added to AIX 5L to increase its compatibility with Linux.

As AIX is UNIX98 branded, all the interfaces defined in those standards are available in AIX. If an application is developed according to those standards, no missing APIs should occur. Linux, however, does not fully comply to these two standards. For a detailed listing of the differences, see Section 4.2, “Linux-compatible APIs and LSB functions on AIX” on page 57 and “New APIs in AIX 5L 5.1” on page 171. The chances of hitting a missing API do decrease significantly if the code was already ported to more than just one operating system. On the other hand, the chances of hitting a missing API also increase if you are programming “close” to the kernel, such as using specific threading features, programming “close” to specific hardware and special device drivers, or even using assembler code. Also, keep in mind that AIX uses eXtended Common Object File Format (XCOFF) while Linux uses Executable and Linking Format (ELF).
When using C++, templates might cause incompatibilities because of differing instantiations. This is especially the case when using different compilers on different platforms. Using the GNU compilers on all platforms should avoid these kind of problems.

Code intended to be portable should not depend on any specific byte ordering (some designs, like Intel architecture, are little endian, while others, like POWER architecture, are big endian) or alignment and size of scalar types (structure sizes and alignment of long might differ on different architectures). National language support (NLS) and specific characteristics in the networking layers (serialization, blocking IP, and so on) are also areas to watch out for when creating portable code.

If you are developing applications to be installed on several platforms, the packaging and delivery vehicle is also important. Now that Linux and AIX offer the same package management tool (RPM), this is the natural choice for this combination of environments.

Of course, certain prerequisites, such as both libraries and other software (like databases and middleware), have to be available for the target platform, especially if it is proprietary software and the source code not available.

### 4.2 Linux-compatible APIs and LSB functions on AIX

Libraries for use by programs consist of header files that define types, macros, declare variables, and function prototypes, and the actual library or archive that contains the definitions of the variables and functions. We must keep this in mind when porting or developing a new application for deployment on multiple platforms because if the header files or libraries used are not present on the destination platform, we can run into invalid header files, header files not found, and missing or unresolved symbols situations.

As a generic recommendation (to avoid getting into these uncomfortable situations), we must make sure that our program source files include the appropriate header files, so that the compiler has declarations of these facilities available and can correctly process references to them. Once our program has been compiled, the linker resolves these references to the actual definitions provided in the libraries.

As a matter of programming style, we must explicitly include all header files required for the libraries facilities we are using, and avoid the use of library header files that automatically include other library header files.
When building the RPM packages for the AIX Toolbox for Linux Applications, some Linux system calls and library symbols could not be found in the equivalent library on AIX. For example, the function frexp() exists in libm.a on Linux and in libc.a on AIX; the function flock() exists in libm.a on Linux and is implemented on AIX by using the fcntl() API.

One of the major goals during a porting project is to assure cross-distribution and compatibility without impeding new enhancements and improvements to the application. Application Programming Interfaces (APIs), shared libraries, and header files are at the core of many application compatibility issues that are raised during the porting phase. An issue that is simple to avoid and sometimes is not taken into consideration is the lack of strict control over versions of libraries.

Different situations can be presented to us, such as missing symbols, undefined libraries, missing header files, or in a worst-case scenario, when the application compiles and links successfully but its output results in unexpected errors when the application runs, up to the point of crashing.

Imagine that we have an API on Linux with the same name and functionality as an API on AIX, but with different structure parameters, such as size, order, pointer references, macro usage, and different macro values. In this case, we must consider not simply recompiling and linking, but carefully looking at the code and, if necessary, re-writing the code or parts of it.

As a generic recommendation to avoid unexpected situations, we must validate and test all ported applications through a validation process. This is because the code might compile error free, while the resulting program might still produce unexpected results.

Appendix A, “APIs” on page 153 provides a list of all Linux system calls and their level of compatibility to AIX and shows analogous information for the Linux Standard Base (LSB) calls. Appendix A, “APIs” on page 153 also shows, in detail, the Linux-compatible APIs added into AIX 5L Version 5.1.

### 4.3 File macro supported values

Macros allow the developer to hide non-meaningful parameter values behind descriptive macro names. They also provide greater application portability because application source code does not have to be changed if parameter values are defined differently on different systems. These values are masked by invariable macro names.
Example 4-1 provides a way of seeing the differences between buffered I/O (using the `fopen()` API to create and open a file) and non-buffered I/O (using the `touch` command to create the file and using the `open()` API with file access mode `O_WRONLY` to open the file in write only mode). Once compiled and run, the program produces three forms of output: stdout and two files, `File1.out`, and `File2.out`.

Example 4-1  Buffered I/O and non-buffered I/O

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

main(int nargc, char *argv[])
{
    char *str;
    int nFid;
    FILE *fp;

    // Buffered Output to Standard Out
    printf("printf: Writing to stdout (File2.out)\n\n");
    fprintf(stdout, "fprintf: Writing to stdout (File2.out)\n\n");
    str = "fwrite: Writing to stdout (File2.out)\n\n";
    fwrite(str, 1, strlen(str), stdout);

    // Non-Buffered Output to Standard Out
    str = "write: Writing to stdout (File1.out)\n\n";
    write(1, str, strlen(str));

    // Buffered Output to a File (File2.out)
    // fopen function creates a new stream named File2.out and
    // opens it for (w) writing only, if the file already exists,
    // it is truncated to zero length, otherwise a new file is created.
    fp=fopen("File2.out", "w");
    if (fp != NULL)
    {
        fprintf(fp, "fprintf: Writing to file\n");
        str = "fwrite: Writing to file\n";
        fwrite(str, 1, strlen(str), fp);
        fclose(fp);
    }
```

Chapter 4. Source compatibility: Linux-compatible APIs on AIX  59
// Non-Buffered Output to a File (File1.out)
// Create the file File1.out by using the operating system command
// touch, once created is opened using the open function which
// returns a file id (nFid) and the file access mode O_WRONLY
// (write only). The close function uses the file id (nFid) to
// close the open file.

system("touch File1.out");

nFid = open("File1.out", O_WRONLY);
if (nFid >= 0)
{
    str = "write: Writing to file\n";
    write(nFid, str, strlen(str));
    close(nFid);
}

**Tip:** The C source code in Example 4-1 uses "/" to introduce a comment in
the line, as in C++ style. By default, gcc compiler accepts this style, but the
AIX xlc compiler does not. For that, it is necessary to provide the -q flag with
the cpluscmt option:

```
# xlc -qcpluscmt <inputfile>
```

These macro values should be looked at carefully, because sometimes
applications could have been written with these values *hardcoded* into it; in such
cases, the application may be reviewed or changed when being ported to
another architecture.

### 4.3.1 File access modes

In an operating system, the control of file access is a fundamental service;
therefore, the operating system must ensure that users are provided appropriate
access for their use, that no two users simultaneously update the same record,
and that each task waits its turn.

The file access modes allow a file descriptor to be used for reading, writing, or
both. A file descriptor is an unsigned integer used by a process to identify an
open file. They are generally unique to each process, but they can be shared by
child processes created with a fork() subroutine or copied by the fcntl(), dup(),
and dup2() subroutines.
File descriptors are indexes to the file descriptor table in the u_block area maintained by the kernel for each process. The most common ways for processes to obtain file descriptors are through open() or creat() operations or through inheritance from a parent process. When a fork() operation occurs, the descriptor table is copied for the child process, which allows the child process equal access to the files used by the parent process.

System file and file descriptor tables
The system file and file descriptor data structures track each process’ access to a file and ensure data integrity. Table 4-1 provides a definition of the activity and contents of each of these structures.

Table 4-1 File descriptor and system table definition

<table>
<thead>
<tr>
<th>Structure</th>
<th>Activity and contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>File descriptor table</td>
<td>Translates an index number (file descriptor) in the table to an open file. File descriptor tables are created for each process and are located in the u_block area set aside for that process. Each of the entries in a file descriptor table has two fields: the flags area and the file pointer. The structure of the file descriptor table is:</td>
</tr>
<tr>
<td></td>
<td>struct ufd</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>struct file *fp;</td>
</tr>
<tr>
<td></td>
<td>int flags;</td>
</tr>
<tr>
<td></td>
<td>} u_ufd[OPEN_MAX]</td>
</tr>
<tr>
<td></td>
<td>The close-on-exec (FD_CLOEXEC bit) flag can be set in the file descriptor table using the fcntl subroutine. The dup subroutine copies one file descriptor entry into another position in the same table. The fork subroutine creates an identical copy of the entire file descriptor table for a child process.</td>
</tr>
</tbody>
</table>
When developing an application or porting one, we must keep this structure in mind, because the file access modes are defined differently in Linux and AIX in some cases. Therefore, as a generic recommendation, the macros described in Table 4-2 must be used by those APIs when file manipulation subroutines are required, such as open(), fnctl(), lseek(), dup(), or dup2(), and hard coded values have to be avoided.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Activity and contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>System file table</td>
<td>Contains entries for each open file. Two of the most important pieces of information tracked in a file table entry are the current offset referenced by all read or write operations to the file and the open mode (O_RDONLY, O_WRONLY, or O_RDWR) of the file. The open file data structure contains the current I/O offset for the file. The system treats each read/write operation as an implied seek to the current offset. Thus, if x bytes are read or written, the pointer advances x bytes. The lseek subroutine can be used to reassign the current offset to a specified location in files that are randomly accessible. Stream-type files (such as pipes and sockets) do not use the offset because the data in the file is not randomly accessible.</td>
</tr>
</tbody>
</table>

**Table 4-2  File access mode macro value comparison**

<table>
<thead>
<tr>
<th>Flag</th>
<th>Linux</th>
<th>AIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>O_ACCMODE</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>O_RDONLY</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>O_WRONLY</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>O_RDWR</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>O_CREAT</td>
<td>0x40</td>
<td>0x100</td>
</tr>
<tr>
<td>O_EXCL</td>
<td>0x80</td>
<td>0x400</td>
</tr>
<tr>
<td>O_NOCTTY</td>
<td>0x100</td>
<td>0x800</td>
</tr>
<tr>
<td>O_TRUNC</td>
<td>0x200</td>
<td>0x200</td>
</tr>
<tr>
<td>O_APPEND</td>
<td>0x400</td>
<td>0x8</td>
</tr>
<tr>
<td>O_NONBLOCK</td>
<td>0x800</td>
<td>0x4</td>
</tr>
<tr>
<td>O_NDELAY</td>
<td>0x800</td>
<td>0x8000</td>
</tr>
</tbody>
</table>
Table 4-3 describes the open or access modes that are common to both Linux and AIX.

**Table 4-3  File open mode macros on Linux and AIX**

<table>
<thead>
<tr>
<th>Flag</th>
<th>Linux</th>
<th>AIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>O_SYNC</td>
<td>0x1000</td>
<td>0x10</td>
</tr>
<tr>
<td>O_FSYNC</td>
<td>0x1000</td>
<td>undefined</td>
</tr>
<tr>
<td>O_ASYNC</td>
<td>0x2000</td>
<td>undefined</td>
</tr>
</tbody>
</table>

Table 4-4 describes the open modes that are available only in Linux with _USE_GNU defined.

**Table 4-4  File open mode macros available only in Linux using _USE_GNU**

<table>
<thead>
<tr>
<th>Open mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O_ACCMODE</td>
<td>Mask for file access only.</td>
</tr>
<tr>
<td>O_RDONLY</td>
<td>Opens for reading only.</td>
</tr>
<tr>
<td>O_WRONLY</td>
<td>Opens for writing only.</td>
</tr>
<tr>
<td>O_RDWR</td>
<td>Opens for both reading and writing.</td>
</tr>
<tr>
<td>O_CREAT</td>
<td>Creates file, if it does not exist.</td>
</tr>
<tr>
<td>O_EXCL</td>
<td>If both O_CREAT and O_EXCL are set, the open is unsuccessful if the file exists. This is guaranteed to never erase an existing file.</td>
</tr>
<tr>
<td>O_NOCTTY</td>
<td>The file is never assigned to a tty.</td>
</tr>
<tr>
<td>O_TRUNC</td>
<td>Truncates the file.</td>
</tr>
<tr>
<td>O_APPEND</td>
<td>Sets append mode.</td>
</tr>
<tr>
<td>O_NONBLOCK</td>
<td>No system calls will block on the file.</td>
</tr>
<tr>
<td>O_NDELAY</td>
<td>As O_NONBLOCK.</td>
</tr>
<tr>
<td>O_SYNC</td>
<td>Performs a synchronous write, blocked until physically updated.</td>
</tr>
<tr>
<td>O_DIRECTORY</td>
<td>If the file is not in the directory, the open fails.</td>
</tr>
<tr>
<td>O_NOFOLLOW</td>
<td>Does not follow a symbolic link in the directory.</td>
</tr>
</tbody>
</table>
Table 4-5 describes the open mode that is available in Linux with __USE_LARGEFILE64 defined and in AIX with _LARGE_FILE_API defined.

Table 4-5   Linux open mode using __USE_LARGEFILE64

<table>
<thead>
<tr>
<th>Open mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O_LARGEFILE</td>
<td>Allows large files.</td>
</tr>
</tbody>
</table>

Table 4-6 describes the open modes that are available in Linux with either __USE_POSIX199309 or __USE_UNIX98 defined and in AIX with _XOPEN_SOURCE==500.

Table 4-6   Linux open modes using __USE_POSIX199309 or __USE_UNIX98

<table>
<thead>
<tr>
<th>Open mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O_DSYNC</td>
<td>Synchronizes write option (file data only).</td>
</tr>
<tr>
<td>O_RSYNC</td>
<td>Synchronizes file attributes on read.</td>
</tr>
</tbody>
</table>

### 4.3.2 File descriptor flags for fcntl

fcntl is a UNIX libc (standard C library) subroutine that performs file control and I/O control on file descriptors. For our purposes, the fcntl() structure is identical in Linux and AIX.

The syntax is:

```c
int fcntl(FileDescriptor, Command, Argument)
```

where:

- **FileDescriptor** Specifies an open file descriptor obtained from a successful call to the open, fcntl, or pipe subroutines. File descriptors are small, positive integers used (instead of file names) to identify files.

- **Command** Specifies the operation performed by the fcntl subroutine. The fcntl subroutine can duplicate open file descriptors, set file-descriptor flags, set file descriptor locks, set process IDs, and close open file descriptors.

- **Argument** Specifies a variable whose value sets the function specified by the Command parameter. When dealing with file locks, the Argument parameter must be a pointer to the flock structure.
4.3.3 File modes

File permission names and values are identical on Linux and AIX.

4.3.4 Poll macro values

The poll API is a standard subroutine in the UNIX libc.a library, and its main function is to check the I/O status of multiple file descriptors and message queues to see if they are ready for reading (receiving) or writing (sending), or to see if they have an exception condition pending.

**Note:** The poll API applies only to character devices, pipes, message queues, and sockets, but not all character device drivers support it. Please refer to the descriptions of individual character devices for information about whether and how specific device drivers support the poll and select subroutines.

The header file poll.h defines several structures used by the poll API. One of these structures is the pollfd, which defines an array of file descriptors or file pointers.

For our purposes, the pollfd structures are identical, but the poll macro values for the event elements are different, as described in Table 4-7. These macro values are of special interest to us if we are developing or porting applications, because as mentioned before, if they are different, we must verify that they have not been hardcoded into the application.

<table>
<thead>
<tr>
<th>Event</th>
<th>Linux</th>
<th>AIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>POLLIN</td>
<td>0x001</td>
<td>0x001</td>
</tr>
<tr>
<td>POLLOUT</td>
<td>0x004</td>
<td>0x002</td>
</tr>
<tr>
<td>POLLPRI</td>
<td>0x002</td>
<td>0x004</td>
</tr>
<tr>
<td>POLLRWNORM</td>
<td>0x100</td>
<td>0x002</td>
</tr>
<tr>
<td>POLLRDNORM</td>
<td>0x040</td>
<td>0x010</td>
</tr>
<tr>
<td>POLLRDBAND</td>
<td>0x080</td>
<td>0x020</td>
</tr>
<tr>
<td>POLLWRBAND</td>
<td>0x200</td>
<td>0x040</td>
</tr>
<tr>
<td>POLLMMSG</td>
<td>0x400</td>
<td>0x080</td>
</tr>
</tbody>
</table>
4.4 Signal values

A signal is a software interrupt delivered to a process. The operating system uses signals to report exceptional situations to an executing program. Some signals report errors, such as references to invalid memory addresses, while others report asynchronous events, such as disconnection of a phone line. For example, if you anticipate an event that will cause signals, you can define a handler function and tell the operating system to run it when that particular type of signal arrives.

A signal can also report the occurrence of an exceptional event. The following events are some events that can cause or raise a signal:

▶ A program error, such as dividing by zero or issuing an address outside the valid range.
▶ A user request to interrupt or terminate the program. Most environments are set up to let a user suspend the program by typing Ctrl-Z, or terminate it with Ctrl-C. Whatever key sequence is used, the operating system sends the proper signal to interrupt the process.
▶ The termination of a child process.
▶ Expiration of a timer or alarm.
▶ A call to kill or raise by the same process.
▶ A call to kill from another process. Signals are a limited but useful form of interprocess communication.
▶ An attempt to perform an I/O operation that cannot be done. An example is reading from a pipe that has no writer.

Some signal values are different between Linux and AIX, as shown in Table 4-8. (No code was found in the Linux kernel to implement the SIGSTKFLT signal.)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Linux</th>
<th>AIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGHUP</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SIGINT</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>SIGQUIT</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>SIGILL</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>SIGTRAP</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>SIGABRT</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>SIGIOT</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Signal</td>
<td>Linux</td>
<td>AIX</td>
</tr>
<tr>
<td>-------------</td>
<td>-------</td>
<td>-----</td>
</tr>
<tr>
<td>SIGBUS</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>SIGFPE</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>SIGKILL</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>SIGUSR1</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>SIGSEGV</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>SIGUSR2</td>
<td>12</td>
<td>31</td>
</tr>
<tr>
<td>SIGIPE</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>SIGALRM</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>SIGTERM</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>SIGSTKFLT</td>
<td>16</td>
<td>undefined</td>
</tr>
<tr>
<td>SIGCLD</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>SIGCHLD</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>SIGCONT</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>SIGSTOP</td>
<td>19</td>
<td>17</td>
</tr>
<tr>
<td>SIGTSTP</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>SIGTTIN</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>SIGTTOU</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>SIGURG</td>
<td>23</td>
<td>16</td>
</tr>
<tr>
<td>SIGXCPU</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>SIGXFSZ</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>SIGVTALRM</td>
<td>26</td>
<td>34</td>
</tr>
<tr>
<td>SIGPROF</td>
<td>27</td>
<td>32</td>
</tr>
<tr>
<td>SIGWINCH</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>SIGPOLL</td>
<td>29</td>
<td>23</td>
</tr>
<tr>
<td>SIGIO</td>
<td>29</td>
<td>23</td>
</tr>
<tr>
<td>SGPWR</td>
<td>30</td>
<td>29</td>
</tr>
<tr>
<td>SIGSYS</td>
<td>31</td>
<td>12</td>
</tr>
<tr>
<td>Signal</td>
<td>Linux</td>
<td>AIX</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>_NSIG</td>
<td>64</td>
<td>undefined</td>
</tr>
</tbody>
</table>
Chapter 5. Package building and porting

In this chapter, we describe the basic application development environment that is available with the Toolbox. The first section discusses the requirements and procedures that are necessary to install the GNUPro Toolkit, which contains the gcc and g++ compilers, as well as other utilities and the gdb debugger. The second section shows how to recompile Toolbox applications from source and how to make changes to packages or produce your own packages (both RPMs and SRPMs). The third section briefly describes how to compile and install open source software that is not packaged in RPM format. Then there is a discussion of shared libraries and the GNU utility to handle them, libtool. libtool is one of the key components of the Toolbox and is used to simplify the building of applications that use shared libraries. In the last section of this chapter, we give some more examples of porting applications and building packages using the Toolbox. This chapter is closely related to Chapter 4, “Source compatibility: Linux-compatible APIs on AIX” on page 55 and Appendix C, “Other Open Source Software for AIX” on page 193.
5.1 Compiler installation and requirements

The GNUPro development environment images are now available as RPM installable packages. They contain the gcc compiler, gcc C++ compiler, gdb debugger and associated utilities, such as ar, nm, and readelf.

We recommend that you install the complete filesets bos.adt and X11.adt prior to installing the compiler suite, although a subset might be enough in some cases. These filesets provide header files, libraries, and some other tools needed for the development environment. A total of 20 MB of disk space in /usr will be needed for both filesets.

The RPM packages can be found in the Toolbox Web site in the same directory as the other RPMs:


For an installation of all images of the GNUPro Toolkit, we need 80 MB of disk space. The installation is performed by using the `rpm -ivh` command as shown in Example 5-1.

Example 5-1 Installing the GNUPro development package

```bash
# rpm -ivh gcc-2.9.aix43.010216-1.aix5.1.ppc.rpm
# rpm -ivh g++-2.9.aix43.010216-1.aix5.1.ppc.rpm
# rpm -ivh binutils-2.9.aix43.010216-1.aix5.1.ppc.rpm
# rpm -ivh gdb-2.9.aix43.010216-1.aix5.1.ppc.rpm
```

All four packages get installed under the /opt/freeware/GNUPro directory. Also, links to the executables may be created in /usr/bin and /usr/linux/bin. The link in /usr/bin is created if the executables do not conflict with the AIX gcc compiler. Before executing the commands, make sure the /usr/linux/bin and /usr/bin directories are set in the PATH variable.

**Note:** The gcc package is a prerequisite of the g++ package.

Installation of the GNUPro sources

Optionally, you can also install the GNUPro sources. They can be found in the Toolbox Web site in the same directory as the other sources:

5.2 Rebuilding Toolbox packages

The first and easiest step to use the compiler kit might be to simply rebuild a package that is already part of the Toolbox from its Source RPM (SRPM or .src.rpm). Let us take a look at the `rpm` command that will do the job for us.

5.2.1 Building packages with rpm

In Chapter 3, “Toolbox installation” on page 37, we use the `rpm` command provided by the rpm.rte package to install binary packages into the system. But RPM can do more than that. It has additional options that allow you to install source packages, or SPRMs, compile them, and produce new RPMs and SRPMs. Detailed descriptions of the capabilities and usage of RPM can be found in the references mentioned in Section 3.3, “Useful URLs” on page 53, especially Maximum RPM by Ed Bailey, found at:

http://www.rpm.org/rpmbook/

An RPM’s build process is controlled by a file called spec, which is part of every SRPM. This file has several sections for the various stages in the build process. The most important stages are:

- **preamble**: Information about the package and its history; intended to be read by human beings.
- **%prep**: To set up a clean build environment, expand archives, and so on.
- **%build**: The actual build step for the software (make is typically executed here).
- **%install**: Virtual installation of the software within the built environment (make install will be placed here).
- **%files**: A list of all files belonging to the package.

Here is a short summary of the options used during the build process. The basic syntax is:

```
# rpm -b<stage> options file1.spec
```

where `<stage>` can be

- **p**: Executes only the %prep section of the spec file.
- **c**: Executes %prep, %build.
- **i**: Executes %prep, %build, and %install.
- **b**: Executes %prep, %build, and %install, and builds the binary package.
a  Executes %prep, %build, and %install, and builds the binary and source packages.

Additionally, the following options might be useful:

--short-circuit  Forces build to start at a particular stage (-bc, -bi only).
--vv  Displays debug information.

After this introduction to the way rpm -b works, we can turn to an example. This example will include a deeper look at the contents of a typical spec file.

5.2.2 Rebuilding a Toolbox RPM

Using libjpeg as an example, we will demonstrate how to rebuild packages of the Toolbox from source code (SRPMs, to be precise). We will analyze the important sections in the respective spec files and produce our first .rpm and .src.rpm files.

First, install the libjpeg SRPM from the Toolbox by issuing the following command from the SRPM directory (for example, /cdrom/SRPMS/libjpeg):

```
# rpm -iv libjpeg-6b-2.src.rpm
```

This will install the sources, patches, and the spec file into the subdirectories of /opt/freeware/src/packages, as shown in Example 5-2.

Example 5-2  /opt/freeware/src/packages directory

```
# pwd
/opt/freeware/src/packages
# ls SOURCES SPECS
SOURCES:
jpeg-6b-aixltconf.patch  jpegsrc.v6b.tar.gz
SPECS:
libjpeg.spec
```

The fastest way to build the libjpeg RPM and SRPM files would be to issue the command `rpm -ba libjpeg.spec` from the /opt/freeware/src/packages/SPECS directory. But because we want to see the process step-by-step, we start by just going through the %prep stage of the spec file, as shown in Example 5-3.

Example 5-3  Lines from the spec file

```
%prep
%setup -q -n jpeg-6b
%patch0 -p1 -b .aixlt
```
The %prep section is pretty small because of the possible use of macros (%setup and %patch) here. The macros are explained in detail at the beginning of this section. For our purposes, you can look at the output in Example 5-4, which shows the expansion of the macros. First, the sources are unpacked into the BUILD subdirectory of /opt/freeware/src/packages/ and then a patch is applied. We will explain this patch in greater detail in Section 5.4, “Using libtool to handle shared libraries” on page 82.

Example 5-4  Expansion of the macros

```
# rpm -bp libjpeg.spec
Executing(%prep): /bin/sh -e /var/opt/freeware/tmp/rpm-tmp.13945
  + umask 022
  + cd /opt/freeware/src/packages/BUILD
  + cd /opt/freeware/src/packages/BUILD
  + rm -rf jpeg-6b
  + tar -xf -
  + /bin/gzip -dc /opt/freeware/src/packages/SOURCES/jpegsrc.v6b.tar.gz
  + STATUS=0
  + [ 0 -ne 0 ]
  + cd jpeg-6b
  + /bin/id -u
  + [ 0 = 0 ]
  + /bin/chown -Rhf root .
  + /bin/id -u
  + [ 0 = 0 ]
  + /bin/chgrp -Rhf system .
  + /bin/chmod -Rf a+rX,g-w,o-w .
  + echo Patch #0 (jpeg-6b-aixltconf.patch):
  Patch #0 (jpeg-6b-aixltconf.patch):
    + patch -p1 -b --suffix .aixlt -s
  + 0< /opt/freeware/src/packages/SOURCES/jpeg-6b-aixltconf.patch
  + exit 0
```

As no error was returned, we can now go one step further and let RPM also execute the %build stage of the libjpeg.spec file, which is shown in Example 5-5. The command, together with an edited and annotated summary of its output, is shown in Example 5-6 on page 74. Comments are set in square brackets and italics. For better readability of the output, some lines of the original output were deleted. This is indicated by “...”.

Example 5-5  %build stage in the .spec file

```
%build
%configure

  --enable-shared --enable-static --prefix=$RPM_BUILD_ROOT%{_prefix} \
  --exec_prefix=$RPM_BUILD_ROOT%{_prefix}
make
%ifnarch armv4l
```
In this section, some changes were made to the spec file. Compare it to the one that you will find, for example, in RedHat Linux for Intel Itanium architectures. The values for --prefix and --exec-prefix were added here. \%(_prefix) is a macro that is defined in the macros file, which is part of the basic rpm command package. It can be found in the directory /usr/opt/freeware/lib/rpm/ with other configuration files for RPM, such as rpmsrc. It is advisable to use macros from the macros file in the spec file wherever possible. This will facilitate architecture independent spec files and ease porting.

Example 5-6   Building stage

    # rpm -bc libjpeg.spec
    Executing(%prep): /bin/sh -e /var/opt/freeware/tmp/rpm-tmp.3514
      [...%prep section output is identical to Example 5-4 on page 73...]
    + exit 0
    Executing(%build): /bin/sh -e /var/opt/freeware/tmp/rpm-tmp.5331
      + umask 022
      + cd /opt/freeware/src/packages/BUILD
      + cd jpeg-6b
      + CFLAGS=-O2 -fsigned-char
      + export CFLAGS
      + CXXFLAGS=-O2 -fsigned-char
      + export CXXFLAGS
      + FFLAGS=-O2 -fsigned-char
      + export FFLAGS
      [...configure is called with the mentioned additional options]
      + [ -f configure.in ]
      + ./configure ppc-ibm-aix4.3 --prefix=/opt/freeware --exec-prefix=/opt/freeware
        --bindir=/opt/freeware/bin --sbindir=/opt/freeware/sbin
        --sysconfdir=/opt/freeware/etc --datadir=/opt/freeware/share
        --includedir=/opt/freeware/include --libdir=/opt/freeware/lib
        --libexecdir=/opt/freeware/libexec --localstatedir=/opt/freeware/var
        --sharedstatedir=/opt/freeware/com --mandir=/opt/freeware/man
        --infodir=/opt/freeware/info --enable-shared --enable-static
        --prefix=/var/tmp/libjpeg-root/opt/freeware
        --exec_prefix=/var/tmp/libjpeg-root/opt/freeware
        checking for gcc... gcc
        checking whether the C compiler (gcc -O2 -fsigned-char ) works... yes
        [...the usual configure tests are being done here...]
        checking libjpeg version number... 62
        creating ./config.status
        creating Makefile
        creating jconfig.h
[now make is called; libtool is used as an interface to call the gcc compiler;]

+ make

./libtool --mode=compile gcc -O2 -fsigned-char -I. -c ./jcapimin.c

gcc -O2 -fsigned-char -I. -c -DPIC ./jcapimin.c

ln -s jcapimin.o jcapimin.lo

./libtool --mode=compile gcc -O2 -fsigned-char -I. -c ./jcapistd.c

gcc -O2 -fsigned-char -I. -c -DPIC ./jcapistd.c

ln -s jcapistd.o jcapistd.lo

[...skipping some lines of compiler calls...]

[libtool is used to link the first library, libjpeg.so.62.0.0]

./libtool --mode=link gcc -o libjpeg.la jcapimin.lo jcapistd.lo jctrans.lo

cparam.lo jdatadst.lo jcinit.lo jcmaster.lo jcmarker.lo jcmaintct.lo

cprectc.lo jcoedctc.lo jccolor.lo jcsample.lo jchuff.lo jchpuff.lo jcdctmgr.lo

cdctfst.lo jfdctflit.lo jfdctint.lo jdpamin.lo jdpaintd.lo jdtrans.lo

datasc.lo jdmaster.lo jdinutlo.jdmarker.lo jdhufflo.jdphuff.lo jdmaintct.lo

dcoedctc.lo jdpctcct.lo jddctmgr.lo jdtcflit.lo jdtcflitt.lo jdtcint.tlo

dctredc.lo jdsamplelo.jdcolorlo.jquantlo.jquant2lo.jdmerge.lo jcomapi.lo

dultilo.jerrorlo.jmemmgrlo.jmemnobs.lo

-rpath /var/tmp/libjpeg-root/opt/freeware/lib -version-info 62

mkdir .libs

gcc -shared jcapimin.o jcapistd.o jctransc.o jparamc.o jdatadstc.o jcinitc.o

cmasterc.o jcmarkerc.o jcmaintctc.o jckprectc.o jccoeftc.o jccolorc.o jcsamplec.o

chuffc.o jchpuffc.o jcdctmgrc.o jdfctflitc.o jdfctintc.o jdpamin.c

dpaintd.c.jdtrans.c.jdatasc.c.jdmaster.c.jdinutlo.jdmarker.c.jdhuffc.c.

dphuffc.o.jdmaintct.c.jdcoefctc.o.jdpctcctc.o.jddctmgrc.o.jdctcflitc.o.

dctcflittc.d.jdtcintc.tlo.jdscapel.o.jdcolorlo.jquantlo.jquant2lo.jdmerge.c.

comapi.c.jultilo.jerrorlo.jmemmgrlo.jmemnobs.c.o.-lc.-Wl,-bnoentry -O

.libs/libjpeg.so.62.0.0

[...skipping more compiler and linker calls...]

[make test to check for correctness of the previous step]

+ make test

+ LD_LIBRARY_PATH=/opt/freeware/src/packages/BUILD/jpeg-6b

rn -f testout *

./djpeg -dct int -ppm -outfile testout.ppm . /testorig.jpg

./djpeg -dct int -bmp -colors 256 -outfile testout.bmp . /testorig.jpg

./cjpeg -dct int -outfile testout.jpg . /testimg.ppm

./djpeg -dct int -ppm -outfile testoutpp.ppm . /testprog.jpg

./cjpeg -dct int -progressive -opt -outfile testoutpp.jpg . /testimg.ppm

./jpegtran -outfile testouttt.jpg . /testimg.jpg

cmp . /testimg.ppm testout.ppm

cmp . /testimg.bmp testout.bmp

cmp . /testimg.jpg testout.jpg

rm . /testimg.ppm testout.ppm
cmp . /testimg.jpg testout.jpg

+ exit 0
In the next example, we execute the `%install` section of the spec file (see Example 5-7). The (edited and annotated) output, beyond what was previously shown, can be found in Example 5-8 on page 77. The variable RPM_BUILD_ROOT is set by the instruction Buildroot: /var/tmp/libjpeg-root in the preamble of the spec file. The name of the files (with patches) to be applied are specified there as well (with the instruction Patch0: jpeg-6b-aixltconf.patch in this case).

After the creation of the directories, the just compiled files belonging to libjpeg are installed under the RPM_BUILD_ROOT directory by using a `make install` command. Finally, some links are created in order to make the binaries available in standard locations, as described in Section 2.3.1, “Directory structure” on page 17. The section of the spec file for the creation of these links is not the present standard Linux/Intel spec file for libjpeg.

**Note:** Because the software will be temporarily installed in RPM_BUILD_ROOT, which is located in the /var file system, /var should have enough free space. We recommend 150 MB, if larger packages are to be built.

Example 5-7   %install section of the .spec file

```plaintext
%install
rm -rf $RPM_BUILD_ROOT
%ifos linux
mkdir -p $RPM_BUILD_ROOT/usr/{lib,include,bin,man/man1}
%else
for i in lib include bin man/man1
  do
  mkdir -p $RPM_BUILD_ROOT%{_prefix}/$i
  done
%endif
make prefix=$RPM_BUILD_ROOT%{_prefix} install
/usr/bin/strip $RPM_BUILD_ROOT%{_prefix}/bin/* || : 

(cd $RPM_BUILD_ROOT
  cd %(_prefix)/lib
  for lib in lib*.so.%{LIBVER}
    do
      shortform="echo $lib | sed "s/\.%{LIBVER}$///``
      [ ! -f $shortform ] && ln -sf $lib $shortform
    done
  cd -

  for dir in bin lib include
    do
      mkdir -p usr/$dir
      cd usr/$dir
```
Example 5-8 Installing stage

```bash
# rpm -bi libjpeg.spec
 Executing(%install): /bin/sh -e /var/opt/freeware/tmp/rpm-tmp.17073
 + umask 022
 + cd /opt/freeware/src/packages/BUILD
 + cd jpeg-6b
 + rm -rf /var/tmp/libjpeg-root
 + mkdir -p /var/tmp/libjpeg-root/opt/freeware/lib
 + mkdir -p /var/tmp/libjpeg-root/opt/freeware/include
 + mkdir -p /var/tmp/libjpeg-root/opt/freeware/bin
 + mkdir -p /var/tmp/libjpeg-root/opt/freeware/man/man1
 + make prefix=/var/tmp/libjpeg-root/opt/freeware install
 /usr/bin/installbsd -c -m 644 jconfig.h
 /var/tmp/libjpeg-root/opt/freeware/include/jconfig.h
 [...installing more header files...]
 ./libtool --mode=install /usr/bin/installbsd -c libjpeg.la
 /var/tmp/libjpeg-root/opt/freeware/lib/libjpeg.la
 /usr/bin/installbsd -c .libs/libjpeg.so.62.0.0
 /var/tmp/libjpeg-root/opt/freeware/lib/libjpeg.so.62.0.0
 rm -f /var/tmp/libjpeg-root/opt/freeware/lib/libjpeg.so.62
 /var/tmp/libjpeg-root/opt/freeware/lib/libjpeg.a
 (cd /var/tmp/libjpeg-root/opt/freeware/lib && ln -s libjpeg.so.62.0.0
  libjpeg.so.62)
 (cd /var/tmp/libjpeg-root/opt/freeware/lib && ln -s libjpeg.so.62.0.0
  libjpeg.a)
 /usr/bin/installbsd -c libjpeg.la
 /var/tmp/libjpeg-root/opt/freeware/lib/libjpeg.la
------------------------------------------------------------------------------
Libraries have been installed in:
 /var/tmp/libjpeg-root/opt/freeware/lib

To link against installed libraries in a given directory, LIBDIR, you must use the `-LLIBDIR' flag during linking.

You will also need to do one of the following:
- add LIBDIR to the `LIBPATH' environment variable during execution
- use the `-Wl,-bnolibpath -Wl,-blibpath:LIBDIR:/usr/local/lib:/usr/lib:/lib' linker flag

See any operating system documentation about shared libraries for more information, such as the 1d(1) and 1d.so(8) manual pages.
------------------------------------------------------------------------------
./libtool --mode=install /usr/bin/installbsd -c cjpeg
/var/tmp/libjpeg-root/opt/freeware/bin/cjpeg
/usr/bin/installbsd -c .libs/cjpeg /var/tmp/libjpeg-root/opt/freeware/bin/cjpeg
 [...] installing more files and manpages [...] + /usr/bin/strip /var/tmp/libjpeg-root/opt/freeware/bin/cjpeg
/var/tmp/libjpeg-root/opt/freeware/bin/djpeg
/var/tmp/libjpeg-root/opt/freeware/bin/jpegtran
/var/tmp/libjpeg-root/opt/freeware/bin/rdjpgcom
/var/tmp/libjpeg-root/opt/freeware/bin/wrjpgcom
 [...] adding link ln -sf libjpeg.so.62.0.0 libjpeg.so... [adding links to standard locations for binaries, libraries and headers]
+ cd -
/var/tmp/libjpeg-root
+ mkdir -p usr/bin
+ cd usr/bin
+ ln -sf ../../opt/freeware/bin/cjpeg ../../opt/freeware/bin/djpeg
    ../../opt/freeware/bin/jpegtran ../../opt/freeware/bin/rdjpgcom
    ../../opt/freeware/bin/wrjpgcom.
+ cd -
/var/tmp/libjpeg-root
+ mkdir -p usr/lib
+ cd usr/lib
+ ln -sf ../../opt/freeware/lib/libjpeg.a ../../opt/freeware/lib/libjpeg.la
    ../../opt/freeware/lib/libjpeg.so.62 ../../opt/freeware/lib/libjpeg.so.62.0.0.
+ cd -
/var/tmp/libjpeg-root
+ mkdir -p usr/include
+ cd usr/include
+ ln -sf ../../opt/freeware/include/jconfig.h
    ../../opt/freeware/include/jerror.h  ../../opt/freeware/include/jmorecfg.h
    ../../opt/freeware/include/jpeglib.h.
+ cd -
/var/tmp/libjpeg-root
+ exit 0
Processing files: libjpeg-6b-2
                [processing the %doc section, but an error occurs!]
Executing(%doc): /bin/sh -e /var/opt/freeware/tmp/rpm-tmp.205 2
+ umask 022
+ cd /opt/freeware/src/packages/BUILD
+ cd jpeg-6b
+ DOCDIR=/var/tmp/libjpeg-root/opt/freeware/doc/libjpeg-6b
+ export DOCDIR
+ rm -rf /var/tmp/libjpeg-root/opt/freeware/doc/libjpeg-6b
+ /usr/linux/bin/mkdir -p /var/tmp/libjpeg-root/opt/freeware/doc/libjpeg-6b
/var/opt/freeware/tmp/rpm-tmp.2052[25]: /usr/linux/bin/mkdir: not found.
Bad exit status from /var/opt/freeware/tmp/rpm-tmp.2052 (%doc)
File not found: /var/tmp/libjpeg-root/opt/freeware/doc/libjpeg-6b
                [...] skipping the rest of the output...
During processing of the `%doc` section, an error occurred. It was caused by the missing binary `/usr/linux/bin/mkdir`. In this case, you need to install the package `fileutils`, because the AIX provided `mkdir` binary was not found (and uses a slightly different syntax).

**Tip:** We recommend that you install the following packages for a basic build environment to avoid the related problems just mentioned:

- `autoconf`, `automake`, `bison`, `fileutils`, `findutils`, `flex`, `gawk`, `gettext`, `grep`, `libtool`, `m4`, `make`, and `texinfo`.

This list is not complete and does not avoid all “requisite missing” errors. See the sections about development utilities on the “listing by functional group” page of the AIX Toolbox Web site:


for a more complete listing of available tools.

Sometimes errors caused by the absence of open source versions of already installed AIX binaries are hard to find because no meaningful error message is generated, or the build process continues for some time after the incompatibility occurred.

We recommend installing all packages from the Toolbox and naming them with appropriate descriptions, such as “The GNU version of...” in case of problems that occur during package (re)builds.

After installing the `fileutils` package, the `rpm -bi` command returns no error. This means we can now create a RPM file which could be installed on other machines and a new SRPM file for source distribution. As we did not make any changes to the source in this example, this would not make much sense. However, it would be possible to make changes to the source code in the `/opt/freeware/src/packages/BUILD` directory or to the spec file after the initial install of the `.src.rpm` file. For example, the source for the application could be replaced by a newer version. In this case, we would get a changed RPM and SRPM file. See Section 5.5, “Examples” on page 84.

While changing the source code, it might be handy to use the `--short-circuit` option of `rpm -b`. This allows the build process to start at a specified stage in the spec file (`%build` or `%install`) instead of always starting at the very beginning.

We will not show the output of `rpm -bb`, which would run through the whole spec file again and produce a binary RPM file.
Example 5-9 shows the final creation of both binary and source RPMs with `rpm -ba`. These RPMs can then be found in the directories `/opt/freeware/src/packages/RPMS` and `/opt/freeware/src/packages/SRPMS`.

### Example 5-9  Binary and source RPMs creation

```
# rpm -ba libjpeg.spec
[...skipping output already seen in the previous examples...]
Finding  Provides: (using /opt/freeware/lib/rpm/find-provides)...
Finding  Requires: (using /opt/freeware/lib/rpm/find.requires)...
Requires: libjpeg
Wrote: /opt/freeware/src/packages/SRPMS/libjpeg-6b-2.src.rpm
Wrote: /opt/freeware/src/packages/RPMS/ppc/libjpeg-6b-2.aix4.3.ppc.rpm
Wrote: /opt/freeware/src/packages/RPMS/ppc/libjpeg-devel-6b-2.aix4.3.ppc.rpm
Executing(%clean): /bin/sh -e /var/opt/freeware/tmp/rpm-tmp.2847
+ umask 022
+ cd /opt/freeware/src/packages/BUILD
+ cd jpeg-6b
+ rm -rf /var/tmp/libjpeg-root
+ exit 0
```

### 5.3 Compiling open source software

In this section, we describe how to compile and install open source software without using the RPM utility. Basically, by using the utilities provided by the Toolbox, this can be done “as usual” for those packages. For example, we take the fvwm2 window manager, which is used (among others) in Section 6.1, “Desktop and graphical applications” on page 90.

Download the sources from:

http://fvwm.org/

or

http://xwinman.org/

and unpack them while in the directory, for example, `/opt/freeware/src/`, using the command:

```
# cd /opt/freeware/src; tar -xzvf fvwm-2.2.4.tar.gz
```
Change to the newly created fvwm-2.2.4 directory and follow the instructions in the INSTALL and README files. During the final **make install**, the software will be installed in subdirectories (like bin, lib, or man) of the directory given as the --prefix option to **configure**. Remember to set the environment to be able to execute the binaries and find the executables later on. Example 5-10 briefly shows the compilation and installation process.

Example 5-10  Compilation and installation process

```
# ./configure --prefix=/opt/freeware
        [...skipping some output...]

Configuration:

    FVWM Version:             2.2.4
    Build extra modules?      no
    Have ReadLine support?    no
    Have RPlay support?       no
    Have XPM support?         no: Xpm library or header not found!

# make 2>&1 | tee make.log
        [...skipping some output...]
# make install 2>&1 | tee makeinstall.log
        [...skipping some output...]
```

If you want to do a step-by-step installation of software, but do not have the sources as a tar or compressed tar file but instead as a SRPM package from any Linux distribution, you can extract the sources by one of the following two methods:

- Execute only the %prep section of the SRPM by using `rpm -bp` and retrieve the sources from `/opt/freeware/src/packages/SOURCES`.
- Extract the source archives out of the SRPM by using `rpm2cpio`, as described in Section 3.2.5, “Using the RPM Package Manager” on page 44.

The above described installation procedures are generic for applications developed according to the GNU coding standards, as described at:

http://www.gnu.org/prep/standards_toc.html

In general, developing applications according to these standards will ensure easy portability to various UNIX-based platforms, including Linux. We will learn a bit more about this subject in the following section.

Please see Appendix C, “Other Open Source Software for AIX” on page 193 for a more detailed discussion of other open source software packages, and comments on interoperability with the Toolbox.
5.4 Using libtool to handle shared libraries

The libtool command is a GNU software package which helps develop and maintain shared libraries. It simplifies the use of shared libraries by hiding the complexity. The tool is fully integrated with the GNU autoconf and automake utilities.

For a detail information about libtool, please refer to the following Web site:
http://www.gnu.org/software/libtool/

GNU libtool encapsulates the platform-specific dependencies and the user interface in a single script. The libtool interface helps to hide the idiosyncrasies from the programmers. Many of the open source applications we are looking at make use of libtool.

See the following Web site for more information:
http://www.gnu.org/prep/standards_toc.html

Attention: At the time of writing, the Toolbox contained a patched version of libtool 1.3.5. IBM developers worked with the maintainers of libtool to get these patches inserted into the mainstream code of libtool. This helps ensure an environment for shared libraries on AIX, which is as close as possible to the rest of the UNIX software community. In mid-April 2001, all images were rebuilt with this new libtool version, which is able to produce shared libraries for all versions of AIX that the Toolbox is running on (this applies to both the POWER and Intel Itanium architectures). Therefore, all early users of the Toolbox (prior to mid-April 2001) will unfortunately need to update their older versions via reinstall, so that the packages maintain compatibility with any new packages that they may build or install from this day forward. It is a short term inconvenience to the early users, but will provide long-term stability and compatibility.

Overview of libtool usage

This section will focus on the basic design and use of libtool. For a more detailed description, see the libtool documentation:
http://www.gnu.org/software/libtool/

In order to use libtool, the following files are needed in the source tree of the software to be compiled:

- **configguess** Attempts to guess a canonical system name (such as powerpc-ibm-aix4.3.3.0)
- **config.sub** Validation script for a canonical system name
ltconfig  Generates a libtool script for a given system
ltmain.sh  A generic script implementing basic libtool functionality

These files should not be included in the source tree of the application. Instead, the libtoolize program should be used, which is part of the libtool package itself. In this case (during a software build), the libtool package has to be installed on the system prior to running libtoolize. After copying the needed files with libtoolize to the source tree, the actual libtool script can be generated with the ltconfig command.

In the libjpeg example, a ltconfig script is included in the source tree. This embedded ltconfig script does not contain the patches needed for libtool under AIX, so a patch has to be applied to the ltconfig file (see Example 5-3 on page 72 and Example 5-4 on page 73). Because it is quite common for programs to embed their own versions of ltconfig and ltmain.sh, this patch has to be applied to many packages you might want to install. Fortunately, this patch is quite generic and can be used unchanged (most of the time) for other source code packages. Be aware that this patch might have to be replaced by a newer version, as soon as the final changes to libtool have been made by the libtool maintainers.

Based on information it generates or gathers, ltconfig generates the system specific libtool script. This process is somewhat similar to running ./configure to generate a make file.

The resulting libtool script can then be used as an interface to generate appropriate compiler, linker, debugger, and installer calls. Here are some examples of how libtool transforms generic calls into system specific syntax for handling shared libraries (the first line shown is the call of libtool; the following lines show the resulting commands which libtool executes in turn):

- **Compiler calls**

  # libtool gcc -g -O -c foo.c
gcc -g -O -c -fPIC -DPIC foo.c
mv -f foo.o foo.lo
gcc -g -O -c foo.c >/dev/null 2>&1

- **Linker call for libraries**

  # libtool gcc -g -O -o libhello.la foo.lo hello.lo \
  -rpath /usr/local/lib -lm mkdir .libs
ld -Bshareable -o .libs/libhello.so.0.0 foo.lo hello.lo -lm
ar cru .libs/libhello.a foo.o hello.o
ranlib .libs/libhello.a
creating libhello.la
Running Linux Applications on AIX

- **Linker call for executables**
  
  ```
  # libtool gcc -g -O -o test.o /usr/local/lib/libhello.la
  gcc -g -O -o .libs/test.o -Wl,--rpath -Wl,/usr/local/lib
  /usr/local/lib/libhello.a -lm
  
  creating test
  ```
  
  Note that even library dependencies on libm are resolved automatically.

- **Debugging executables**
  
  ```
  # libtool gdb hell
  ```
  
  In some cases, the debugger has to be called by libtool and not directly.

- **Installing libraries**
  
  ```
  # libtool install -c libhello.la /usr/local/lib/libhello.la
  install -c .libs/libhello.so.0.0 /usr/local/lib/libhello.so.0.0
  install -c libhello.la /usr/local/lib/libhello.la
  install -c .libs/libhello.a /usr/local/lib/libhello.a
  ranlib /usr/local/lib/libhello.a
  ```
  
  Additionally, there is a --finish mode that might have to be called after this step.

  For a more detailed explanation and more examples, see the libtool manual at:
  

5.5 Examples

In this section, we want to discuss the changes necessary to a spec file in general and especially when updating Toolbox packages to a newer release level. As an example, we use wget (see Section 3.2.3, “FTP tools” on page 41 for more details on wget).

5.5.1 Rebuilding and updating the wget package

As described earlier, we first install the wget SRPM from the Toolbox with the command:

```
# rpm -iv /cdrom/SRPMS/wget/wget-1.5.3-1.src.rpm
```

Now change to the directory `/opt/freeware/src/packages/SPECS` and rebuild the original wget package with the command:

```
# rpm -ba wget.spec
```

This should generate the following RPMs:

- wget-1.5.3-1.src.rpm in `/opt/freeware/src/packages/SRPMS/`
Let us now take a closer look at the spec file included in this SRPM and compare it to the spec file included in RedHat Linux 6.2. Example 5-11 shows the annotated output of a `diff` command. A "<" in the first column indicates lines included in the Toolbox spec file, while a ">" indicates lines from the spec file included in the RedHat Linux 6.2 distribution.

Example 5-11   Output of the diff command

```bash
# diff wget.spec wget.specRH
5c5  [the release number of the spec file is changed to 1 in the Toolbox]
< Release: 1
---
> Release: 6
8a9,10  [RedHat applies two patches, while plain sources are used in the Toolbox]
> Patch0: wget-1.5.0-man.patch
> Patch1: wget-1.5.3-symlink.patch
26a29,30
> %patch0 -p1 -b .man
> %patch1 -p1 -b .symlink
29c33  [configure is called with the correct prefix for the Toolbox]
< ./configure --prefix=%{_prefix} --sysconfdir=/etc
---
> #./configure --prefix=/usr --sysconfdir=/etc
30a35
> %configure --sysconfdir=/etc
35,37c40,42  [the correct prefix also has to be added for the following commands; the AIX provided strip command is used by specifying the complete path]
< make install prefix=$RPM_BUILD_ROOT%{_prefix} sysconfdir=$RPM_BUILD_ROOT/etc
< gzip $RPM_BUILD_ROOT%{_prefix}/info/*
< /usr/bin/strip $RPM_BUILD_ROOT%{_prefix}/bin/* || :
---
> make install prefix=$RPM_BUILD_ROOT/usr sysconfdir=$RPM_BUILD_ROOT/etc
> gzip $RPM_BUILD_ROOT/usr/info/*
> strip $RPM_BUILD_ROOT/usr/bin/* || :
39,45d43  [a link to the wget binary is placed in /usr/bin]
< (cd $RPM_BUILD_ROOT
<   mkdir -p usr/bin
<   cd usr/bin
<   ln -sf ../..%{_prefix}/bin/*.
<   cd -
< )
<
47c45  [some hard coded paths have to be adapted to include the correct prefix]
< /sbin/install-info %{_prefix}/info/wget.info.gz %{_prefix}/info/dir
```
> /sbin/install-info /usr/info/wget.info.gz /usr/info/dir
51c49
< /sbin/install-info --delete %{_prefix}/info/wget.info.gz
%{_prefix}/info/dir
---
> /sbin/install-info --delete /usr/info/wget.info.gz /usr/info/dir
61c59 [and finally the file list is changed slightly]
< %{_prefix}/bin/wget
---
> /usr/man/man1/wget.*
63,64c61,62
< %{_prefix}/info/*
< %{_prefix}/share/locale/*/LC_MESSAGES/*
---
> /usr/info/*
> /usr/share/locale/*/LC_MESSAGES/*

We see two categories of changes in this example:

First, some hardcoded paths have to be changed to include the correct prefix
/opt/freeware. This is done by using the %{_prefix} macro, which is defined in the
/usr/opt/freeware/lib/rpm/macros file. Thus, the new spec file is, in a certain
sense, more general than the other one, and could be used on other systems.

The other change is to place links to binaries (and to libraries as well, in certain
cases) in the standard locations (/usr/bin, /usr/linux/bin or /usr/lib, /usr/linux/lib).

Now, we want to update the source code of wget to a newer level. Instead of the
Version 1.5.3 that is currently provided by the Toolbox, we want to use Version
1.6, which that is available from the GNU FTP servers, for example:
ftp://prep.ai.mit.edu/pub/gnu/wget/

It is a simple process to get this new version. Just download the new software
archive wget-1.6.tar.gz to /opt/freeware/src/packages/SOURCES and change
the line:

%define version 1.5.3

in the wget.spec file to:

%define version 1.6

and rebuild by issuing the command:

# rpm -ba wget.spec

This should generate the following RPMs:
Chapter 5. Package building and porting

- wget-1.6-1.src.rpm in /opt/freeware/src/packages/SRPMs/
- wget-1.6-1.aix4.3.ppc.rpm in /opt/freeware/src/packages/RPMS/ppc/

The new version can now be installed with one of these commands:

# rpm -iv /opt/freeware/src/packages/RPMS/ppc/wget-1.6-1.aix4.3.ppc.rpm
# rpm -Uv /opt/freeware/src/packages/RPMS/ppc/wget-1.6-1.aix4.3.ppc.rpm
# rpm -Fv /opt/freeware/src/packages/RPMS/ppc/wget-1.6-1.aix4.3.ppc.rpm

Using the option -iv for rpm will not work if a former version of wget is already installed on the system.
User and administration differences

In this chapter, we provide a general overview of the user and administration environment and its differences between Linux and AIX, such as:

- Desktop and graphical applications
- Available shells, their features, and startup files
- User commands differences
- Administration commands differences
- Boot process differences
- System files differences
6.1 Desktop and graphical applications

Here we provide an overview of the XWindow System, the AIX Toolbox for Linux Applications graphical framework, and the different graphical desktop options available when installing the AIX Toolbox for Linux Applications, such as KDE and GNOME.

Also, as described in Section 3.2.6, “Installing KDE2” on page 50 and in Section 3.2.7, “Installing GNOME” on page 52, we can set KDE2 or GNOME as the default graphical desktop on AIX instead of CDE, which is the default graphical desktop on AIX, which will also be discussed in this chapter.

6.1.1 The XWindow System

The XWindow System (sometimes referred to as “X” or “XWindows”) is an open, cross-platform, client/server system for managing a graphical user interface in a distributed network. It is the standard graphics interface for UNIX-based operating systems. When using X, the user can have multiple terminal windows in the panel at once, and each window can contain a different login session.

One of the great advantages of using the XWindows System is that its functionality is achieved through the cooperation of different components, rather than everything being packed into one single large collective.

We will focus on the specifics of the KDE and GNOME desktops environments provided by the AIX Toolbox for Linux Applications, their functionality, and how they interact with the AIX graphical environment, which is based on Motif 2.1 (a window manager) and X11R6 (X server, libraries, and clients).

Window managers

A very important part of the XWindows System, regardless if the desktop being used is KDE, GNOME, or CDE (the AIX default desktop), is the window manager. The window manager provides us with the look and feel of the X interface. This program is in charge of the placement of windows and the user interface, and is used for resizing, iconifying, moving, and changing the appearance of the window frames. Table 6-1 describes the window managers used by the different available desktops.

<table>
<thead>
<tr>
<th>Desktop</th>
<th>Window manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDE / KDE2</td>
<td>Kwm, Kwin, Enlightenment</td>
</tr>
<tr>
<td>GNOME</td>
<td>sawfish, Enlightenment</td>
</tr>
</tbody>
</table>
Let us now look at the various window managers:

- **Kwm/Kwin**
  
  kwm/kwin is the window manager of choice for the KDE desktop and is part of the kdebase package. It offers:
  
  - Complete integration with KDE.
  - Complete keyboard control and configuration.
  - The ability to be reconfigured at runtime without restarting.
  - A session management and working session management proxy for legacy applications. This proxy is able to restore applications to their previous state, including window properties (such as maximized, preferences, iconified, and so on) and the virtual desktop that the GUI was running.

- **Enlightenment**
  
  Also known as E, this is a window manager for X. Its design goal is to be as configurable as possible in look and feel. Enlightenment is also provided as an alternative window manager in the AIX Toolbox for Linux Applications. We can enable Enlightenment, once installed, by changing a line on the startkde script, located in /opt/freeware/kde/bin. This script is shown in Example 6-1.

**Example 6-1  How to change /opt/freeware/kde/bin/startkde**

Look for the following line:

```
ksmserver --restore
```

and replace it with

```
ksmserver --restore --windowmanager enlightenment
```

Some of the features the Enlightenment window manager offers are:

- Fully configurable window borders
- A graphical pager that takes miniature snapshots of your panel
- Theme support
- Translucent moving windows
- Virtual desktops
- KDE hint support
- GNOME hint support
- Tooltips

**Sawfish**

Previously known as sawmill, this program is a highly configurable window manager for X11. It uses an Emacs Lisp-like scripting language. The user interface policy is controlled through the Lisp language.

User-configuration is possible either by writing Lisp code in a personal .sawfishrc file, or through the integrated customization system.

**dtwm**

The dtwm window manager is based upon the Open Software Foundation (OSF/Motif) window manager (mwm). It facilitates the control of elements of windows, such as placement, size, and icon display. The dtwm is an integral part of the CDE desktop; it communicates and facilitates access to other components in the environment, such as the Session and Style Manager. In addition to this functionality, dtwm provides work space management capabilities. work spaces allow us to group together related windows, and each work space is independent of the other work spaces.

**AIX Toolbox graphical framework**

The AIX Toolbox for Linux Applications provides us with a wide range of tools that were ported from Linux, such as graphical desktop environments (KDE and GNOME), GNU-based utilities (gawk, sed, and tar), system shells (bash, tcsh, and zsh), window managers (enlightenment and sawfish) and administrative tools (kadmin), which, in future releases of the Toolbox, will be more robust and will support the intrinsics of AIX, such as VPD (Vital Product Data) and ODM (Object Data Management).

The graphical desktops available in the AIX Toolbox for Linux Applications are composed of different elements that provide a specific graphical development framework, depending upon the desktop you decide to use.

Figure 6-1 on page 93 show us the interaction of the graphical library layers being used in regards to each of the desktops, along with the interaction of the libraries and the application layer. For our specific case, this application layer is the desktops provided in AIX and in the AIX Toolbox for Linux Applications.
Desktop development libraries

These libraries provide the developer a software development framework that enables cross-platform developing and porting between heterogeneous graphical environments. By using these libraries, developers can create a single code base for different platforms.

In order to develop a graphical client application, a platform needs to provide the appropriate libraries that allow the application to communicate with an X server locally or across a network:

- **XLib**
  - Provides the necessary functions that can be called by an application to perform task such as:
    - Create, move, scale, stack, and delete windows.
    - Draw lines, rectangles, arcs, and polygons.
    - Employ fonts, color maps, graphic images, and cursors.

In theory, the XLib is the only library required to run an X application, and it provides the common base to create an X client application. It sounds easy, but many lines of code are required to produce a simple application by using XLib alone. This is why a set of high level pre-programmed functions known as the X toolkit, or Xt, is provided on most X client platforms. For example, one single Xt function could translate into several XLib calls. Xt functions are also called intrinsics.
Qt

Qt is a cross-platform C++ application framework developed by Trolltech AS. It is implemented as a class library and provides an API for applications developers. This means that an application written with Qt on one platform can run on another platform by recompiling and linking it with the Qt library for that specific platform. Qt is widely used on Linux and is the basis of the KDE desktop environment. It offers a wide range of functions that focus on GUIs (Graphical User Interface) and basically replace the Motif and Xt toolkit.

For more information regarding Qt, please refer to:

GLib

GLib is a library for the C language which contains portability and utility functions. The functionality provided by GLib can be divided into four main categories: portability, convenience functions, generic data structures, and the GLib main loop.

– Portability

GLib provides portable equivalents for a number of functions that are available in some, but not all, C libraries. For example, the functions functionsg_strcasecmp() and g_memmove() are portable equivalents for strcasecmp() and memmove(). On platforms where the standard functionality exists, the GLib functions will just wrap these functions.

– Convenience functions

GLib also provides a number of unique functions to make using C more convenient. For example, it provides functions to break strings into words, to do computations with dates, and to log warning messages and error messages in a flexible fashion.

– Generic data structures

GLib provides unique generic data structures, such as linked lists, hash tables, balanced trees, and variable-length arrays, and it allows programmers to take advantage of sophisticated data structures and improve the efficiency of their programs without having to reimplement the data structures from scratch. For example, the GHashTable type allows a programmer to create a hash table for arbitrary objects by simply providing two functions: a function to compute hash values for the objects in the table, and a function to compare two values.
Chapter 6. User and administration differences

- GLib main loop
  This is a generic and extensible implementation of an event loop. Standard event sources that GLib provides include timers, IO callbacks, and idle functions, but it is also possible to add completely new types of event sources into the GLib main loop.
  GDK uses this functionality to add an event source for X events. By not tying the main loop directly into the Toolkit, as is frequently done, GLib allows both graphical and non-graphical event-driven programs.

- GDK
  The GDK library provides a layer of abstraction that lays between GTK+ widgets and applications and the underlying window system. Instead of making calls directly to the XWindow System, applications call GDK when they need to draw to the panel or handle events.

- GTK
  GTK is a container based toolkit, which means that most widgets (elements of a graphical interface that display or provide information, such as icons, buttons, selection boxes, windows, and so on) serve as containers that hold other widgets. An example of this situation is a button, which is a container that will most likely contain a label widget.
  The GTK and GLib libraries provide the foundation for the user interface of GNOME. The GTK user interface toolkit was originally developed as part of the GIMP (GNU Image Manipulation Program) project, and has become widely used because of its attractive appearance, flexible and convenient programming interface, and unrestricted licensing under the GNU LGPL.

- GNOME libraries
  The GNOME libraries are divided into three basic parts:
  - libgnome
    A utility library very similar to the GLib, which provides services such as configuration loading and saving, application launching, mime-type identification, and metadata storage.
  - libgnomeui
    The user interface part; contains application framework (GnomeApp), the canvas (GnomeCanvas), and other useful, specialized widgets.
  - libgnorba
    Used when integrating GNOME/GTK applications with CORBA (Common Object Request Broker Architecture). It provides the GNOME name server for CORBA and integration of ORBit (a CORBA 2.2-compliant Object Requester Broker) and GNOME/GTK.
For more information regarding GDK, GTK+ and GNOME please refer to:

http://www.gtk.org/

and

http://www.gnome.org/

### 6.1.2 The KDE desktop

KDE is an open source graphical desktop environment for the UNIX operating system and is built on top of the X11 environment. It contains a compound application development framework, which means it provides a large collection of graphical user interface (GUI) applications and an office application suite called Koffice.

The KDE distribution includes modules such as:

- **KDE-Libs**
  - Various run-time libraries, such as kdecore, kdeui, and khtwm
- **KDE-Core**
  - KPanel, Kfm, Kcontrol, Konqueror, Kdisplay, Kwm, Organizer, and KDEHelp
- **KDE-Graphics**
  - Kpaint, Kdvi, KGhostview, and Kfax
- **KDE-Utilities**
  - Kedit, Kcal, and Knotes
- **KDE-Games**
  - Kasteroids, Konquest, Tron, Smiletris, and SnakeRace
- **KDE-Network**
  - KPpackage, KUser, and KDE System Guard (Task Manager and a Performance Monitor)
- **KDE-Network**
  - kmail, Windows Shares (SMB client), Korn (KDE mail checker), and KNode (news reader)

The KDE desktop consists of three main areas:

- A main panel at the bottom of the panel is used to start applications and switch between desktops. A large K icon (on the left side of the panel) displays a menu of applications to start when clicked, as shown in Figure 6-2 on page 97.
A taskbar at the bottom-center of the panel, used to switch between and manage currently running applications. Click on an application on the taskbar to switch to the application, as shown in Figure 6-2.

The desktop itself, on which frequently used files and folders may be placed. KDE provides multiple desktops, each of which has its own or shared windows. Click on the numbered buttons on the panel to switch between desktops, as shown in Figure 6-3.

Figure 6-2 KDE desktop main panel

Figure 6-3 KDE desktop and its main panel
KDE applications
As shown in Figure 6-3 on page 97, we use the large letter K to launch the application menu. KDE provides a set of applications to customize the behavior, functionality, look, and feel of the desktop.

For example, we can customize a frequently used application by adding it to the application starter menu simply by selecting the menu Application starter->Panel Menu->Add->Application->System->File Manager (Super User Mode). Figure 6-4 shows the sequence in a graphical manner. Figure 6-5 on page 99 shows the result: the File Manager is finally added to the main panel and can be quickly and directly accessed from here.
We can now launch the application File Manager from the starter menu by clicking the added button. The result is shown in Figure 6-6 on page 100.

Because we have used the application KDE File Manager throughout our example, it is important to notice one of the great features the KDE File Manager application provides, which is the capability of accessing a URL (Uniform Resource Locator) from the Internet directly, and, by drag and drop, copy the remote file or complete directory to a local destination. This type of technology is used throughout the KDE desktop and is called network transparency. It allows KDE applications to drag and drop an icon from the Kfm/browser to an editor or folder.
KDE provides an suite of office applications (KOffice) which includes:

- **KWord**: A word processor program
- **KSpread**: A spreadsheet program
- **KPresenter**: A presentations program
- **KChart**: An application to draw charts and diagrams
- **KIllustrator**: A vector drawing application
- **KFormula**: An editor for mathematical formulas
- **KImage**: An image viewing application

Figure 6-7 on page 101 and Figure 6-8 on page 102 show sample snapshots from some of the KOoffice applications. These samples were taken using Ksnapshot (a KDE utility used to capture images) and processed using KImage (used to convert the snapshot to a compatible image format).
6.1.3 The GNOME desktop

GNOME (GNU Network Object Model Environment) was conceived as the answer to UNIX’s lack of user friendliness. It was originally designed by programmers for programmers, and the primary GNOME interface was the command line.

GNOME, an effort by the GNU Project to address these problems, is a graphical user interface and a set of computer desktop applications. It is a free and easy-to-use desktop environment for the user, as well as a powerful application framework for the software developer. GNOME is highly configurable, enabling you to set your desktop the way you want it to look and feel. It gives you such flexibility that you can make the graphical interface look like Microsoft® Windows or Mac® OS.

The user interface part of GNOME is built on top of the X foundation and consists of the following groups of applications:

- GNOME desktop system
  
  A set of tools that provides a powerful desktop interface to users, plus various utility applications for day-to-day work.
GNOME application framework libraries

A set of libraries that ensures that GNOME applications look and behave properly.

GNOME productivity applications

Various productivity applications that are part of the GNOME Project and are distributed as part of the GNOME system.

GNOME desktop

The GNOME desktop provides us with the functionality of any modern operating system desktop. We can drag files, programs, and directory folders to the desktop; we can also drag those items back into GNOME-compliant applications, allowing you to quickly access any items you select.

One of the rich functionalities of the GNOME desktop is that it can work with basically any window manager, but the desktop's core functionality and best usage comes through when using a GNOME-compliant window manager, such as Enlightenment, fvwm2, IceWM, or WindowMaker.

 GNOME includes a panel (for starting applications and displaying status) (see Figure 6-9 for more details), a desktop (where data and applications can be placed), a set of standard desktop tools and applications (see Figure 6-10 on page 104 for more details), and a set of conventions that make it easy for applications to cooperate and be consistent with each other.

The GNOME footprint, shown in Figure 6-9, allows you to launch all of GNOME's wonderful features, such as applications, configuration tools, command line prompt, and logout and lock screen commands.
6.1.4 Package managing using KDE or GNOME

The AIX Toolbox for Linux Applications includes two very common GUIs for the RPM Package Manager: KPackage and GnoRPM.

KPackage
KPackage is a GUI interface for the RPM, Debian, Slackware, and BSD package managers, and is part of KDE. As a result, it is fully integrated with the KDE file manager. The KPackage GUI interface can be seen in Figure 6-11 on page 105.

KPackage makes use of the KDE drag and drop protocol. This means that you can drag and drop packages onto KPackage to open them. Dropping a file onto the Find File dialog will find the package that contains the file.

When KPackage is started, it displays two panels with the package tree on the left. This tree shows installed packages and, optionally, new and updated packages as well.

The tabs on the left panel are used to display installed packages, updated packages, available packages, or all packages.
The package tree shows the package name, package size, the version, and (in the case of an available package which would update an installed package) the version of the already installed package.

The right panel has tabs for displaying two different types of information about selected packages: the properties tab, which displays information on the selected package, and the file list tab, which shows the files in the package and, for installed packages, shows the state of the files.

![KPackage GUI interface](image)

**GnoRPM**

GnomeRPM (or gnorpm) is a graphical user interface for the RPM Package Manager that allows us to locate, through rpmfind (a program that will find RPM files for you), and download packages with all their dependencies and install them on our system.
Figure 6-12 displays the main package window. It has a tree of the different package groups on the left and a list of packages in the selected group on the right. The package list on the right can be configured to display as icons or as a list. From this window, you can manipulate the packages that are currently installed in the system.

After selecting some packages, you can uninstall, query, or verify them. Any of these operations can be performed either from the menu items or from the toolbar buttons.

We can bring up windows for other parts of gnorpm from the main window, such as:

**Install window**  
Used to install new packages on the system.

**Find window**  
Used to search and query package information for files installed on the machine, as shown in Figure 6-13 on page 107.

**Web find window**  
Allows you to find and download a package with its dependencies off the Internet, as shown in Figure 6-14 on page 108.
Preferences window  Allows you to change the settings for the gnorpm utility, as shown in Figure 6-15 on page 108.
Figure 6-14  gnorpm web find feature

Figure 6-15  gnome settings for rpmfind

URL used by gnorpm to obtain packages from the internet.
6.1.5 CDE desktop

AIX Version 4 introduced the Common Desktop Environment (CDE). The Common Desktop Environment is an integrated graphical user interface for open systems desktop computing, combining X Window System, OSF/Motif®, and new Common Desktop Environment technologies. CDE is designed to work across a large range of client/server platforms, support small workgroups to large enterprises, and support simple text and data, as well as advanced collaborative multimedia applications.

CDE allows system administrators to gain a higher degree of control over the desktop computing environment that has often been lost in the move from centralized to client-server or distributed computing. CDE gives end users access to the power and flexibility of today’s networked desktop systems.

Many of the familiar AIX tools, such as the System Management Interface Tool (SMIT), Visual Systems Management (VSM), and InfoExplorer can be launched directly from the desktop. Other products, including third-party applications, can be installed into the desktop’s application manager folder, where they will appear as icons.

The look and feel of the CDE desktop is composed of a series of components called panels.

The front panel is a special desktop window that contains a set of controls for doing common tasks. The front panel moves with you as you switch work spaces. Figure 6-16 on page 110 shows the front panel of the desktop; along the top of the front panel is a row of arrow buttons, which open the subpanels.
The front panel is divided in two key elements:

- **Main Panel**

  The main panel is the horizontal window at the bottom of the display. It contains a number of frequently used controls, including the work space switch, which contains buttons for changing to other work spaces.

  In Figure 6-17 on page 111, we provide a brief description of the main panel action tasks.
Chapter 6. User and administration differences

Figure 6-17 Main panel action tasks

- **Clock**: Displays the current time.
- **Calendar**: Provides facilities for the user to manage appointments.
- **File Manager**: Provides facilities to graphically navigate file systems.
- **Text Editor**: Provides a simple, easy-to-use editor.
- **Mailer**: Enables the user to send, receive, and manage e-mail.
- **Work space Switch**: Enables the user to switch work spaces.
- **Print Manager**: Allows the user to view and manage print jobs.
- **Style Manager**: Allows the user to customize the desktop.
- **Applications Manager**: Enables the user to manage applications.
- **Help Manager**: Provides a help guide for the desktop.
- **Trash Can**: Allows the user to place files without deleting them.

- **Subpanels**

  If a control in the Main Panel has an arrow button on top of it, then that control has a subpanel. Below these arrows are icons that allow us to execute different desktop administrative tasks. A subpanel example is provided in Figure 6-18 on page 112.
6.2 Available shells

A shell is a UNIX term for the interactive UNIX user interface within the operating system. The shell is the layer of programming that understands and executes the commands a user enters. It is basically a command interpreter.

AIX, by default, provides the following shells:

- **bsh**
  
  The Bourne shell is an interactive command interpreter and command programming language. It can be run as a login shell or as a subshell under the login shell. Only the `login` command can call the Bourne shell as a login shell. It does this by using a special form of the `bsh` command name: `-bsh`.

- **csh**
  
  The C shell is an interactive command interpreter and a command programming language. It uses a syntax that is similar to the C programming language. The `csh` command starts the C shell.
The Korn shell is an interactive command interpreter and command programming language. It conforms to the Portable Operating System Interface for Computer Environments (POSIX), an international standard for operating systems. The Korn shell (also known as the POSIX shell) offers many of the same features as the Bourne and C shells, such as I/O redirection capabilities, variable substitution, and file name substitution. It also includes several additional command and programming language features:

- **Arithmetic evaluation**
  Performs integer arithmetic using the built-in UNIX `let` command (used to assign values to data variables and to perform arithmetic operations or other calculations on data in columns or constants), using any base from 2 to 36.

- **Command history**
  The Korn shell stores a file that records all of the commands you enter.

- **Coprocess facility editing**
  Enables you to run programs in the background and exchange information with these background processes.

The POSIX shell, similar to ksh.

The restricted shell is used to set up login names and execution environments whose capabilities must be more controlled than those of the regular Bourne shell.

The `Rsh` or `bsh -r` command opens the restricted shell. The behavior of these commands is identical to those of the `bsh` command, except that the following actions are not allowed:

- Changing the directory (with the `cd` command)
- Setting the value of PATH or SHELL variables
- Specifying path or command names containing a `/` (slash)
- Redirecting output

This is the default shell, `/usr/bin/sh` (or `/bin/sh`), and is linked to ksh in AIX.
The **tsh** (trusted shell) command is a command interpreter that provides greater security than the Korn shell (the standard login shell). Generally, a user calls the tsh shell by using the secure attention key (SAK) sequence, which is Ctrl-X followed by Ctrl-R, after a login. The tsh shell also can be invoked by defining it as the login shell in the `/etc/passwd` file.

The trusted shell differs from the Korn shell in the following ways:

- The function and alias definitions are not supported. Alias definitions are only supported in the `/etc/tsh_profile` file.
- The IFS and PATH environment variables cannot be redefined.
- Only trusted programs can be run from the tsh shell.
- The history mechanism is not supported.
- The only profile used is the `/etc/tsh_profile` file.
- The trusted shell has the following built-in commands:
  - **logout** exits the login session and terminates all processes.
  - **shell** re-initializes the user's login session. The effect is the same as logging in to the system.
  - **su** resets the effective ID to the user's identity on the system and executes another trusted shell.

Table 6-2 provides a comparison of standard AIX shell environments.

**Table 6-2** AIX standard shells feature comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>bsh</th>
<th>csh</th>
<th>ksh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compatible with bsh</td>
<td>n/a</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Job control</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Command history</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Command line editing</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Aliases</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>noclobber (protecting files from editing)</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>ignoreeof (ignore control-D)</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
The AIX Toolbox for Linux Applications introduces new shells on AIX. These new shell environments are frequently used and are very common to the Linux community.

These shell environments are:

- bash

  The bash shell is an sh-compatible command language interpreter that executes commands read from the standard input or from a file. bash also incorporates useful features from the Korn and C shells (`ksh` and `csh`).

  The name is an acronym for the ‘Bourne-Again SHell’, a pun on Steve Bourne, the author of the direct ancestor of the current UNIX shell `/bin/sh`, which appeared in the Seventh Edition Bell Labs Research version of UNIX.

  The bash shell is intended to be an implementation that conforms to the IEEE POSIX Shell and Tools specification (IEEE Working Group 1003.2). It offers functional improvements over sh for both interactive and programming use.

  bash is quite portable, and currently runs on nearly every version of UNIX and a few other operating systems. Some independent ports exist for MS-DOS®, OS/2, Windows®, and Windows NT®.

  The bash shell provides:

  - Bourne shell style:
    - Looping constructs
    - Conditional constructs
  
  - C-shell style features:
    - Job control
    - History expansion
    - Protected redirection
    - C shell variables
    - Tilde expansion

  - Korn shell style features:
    - Korn shell constructs
    - Korn shell builtins
    - Korn variables
    - Alias builtins

<table>
<thead>
<tr>
<th>Feature</th>
<th>bsh</th>
<th>csh</th>
<th>ksh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logout file</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Some unique bash builtin commands are:

- **bind**: Binds a key sequence to a readline function, or to a macro.
  
  Syntax:
  ```
  bind [-m keymap] [-lvd] [-q name]
  bind [-m keymap] -f filename
  bind [-m keymap] keyseq:function-name
  ```

- **builtin**: Runs a shell builtin. This is useful when you wish to rename a shell builtin to be a function, but need the functionality of the builtin within the function itself.
  
  Syntax:
  ```
  builtin [shell-builtin [args]]
  ```

- **command**: Runs `<command>` with `<arg>` ignoring shell functions. If you have a shell function called `ls`, and you wish to call the command `ls`, you can say `command ls`.
  
  Syntax:
  ```
  command [-pVv] command [args ...]
  ```

- **declare**: Declares variables and/or gives them attributes.
  
  Syntax:
  ```
  declare [-frxi] [name=value]
  ```

- **enable**: Enables and disables builtin shell commands.
  
  Syntax:
  ```
  enable [-n] [-a] [name ...]
  ```

- **help**: Displays helpful information about builtin commands.
  
  Syntax:
  ```
  help [pattern]
  ```

  An example of this `help` command is shown in Example 6-2 on page 117.
Example 6-2  bash help command usage

bash2-2.04$ help alias
alias: alias [-p] [name[=value] ...]
   `alias' with no arguments or with the -p option prints the list
   of aliases in the form alias NAME=VALUE on standard output.
   Otherwise, an alias is defined for each NAME whose VALUE is given.
   A trailing space in VALUE causes the next word to be checked for
   alias substitution when the alias is expanded. Alias returns
   true unless a NAME is given for which no alias has been defined.

bash2-2.04$

- local: For each argument, creates a local variable called <name>, and
gives it a <value>. local can only be used within a function; it makes the
variable <name> have a visible scope restricted to that function and its
children.

Syntax:
local name[=value]

- type: For each <name>, indicate how it would be interpreted if used as
a command name.

Syntax:
type [-all] [-type | -path] [name ...]

For a complete reference of the bash shell, please refer to:

or

- tcsh

tcsh is an enhanced but completely compatible version of the Berkeley UNIX
C shell (csh). It is a command language interpreter usable both as an
interactive login shell and a shell script command processor. It includes a
command line editor, programmable word completion, spelling correction, a
history mechanism, a job control, and a C-like syntax.

Key features of the tcsh shell:

- Spelling correction

   The shell can correct the spelling of file names, commands and variable
   names, as well as completing and listing them.
Individual words can have their spellings corrected with the `spell-word` editor command (usually bound to Ctrl-s and Ctrl-S) and the entire input buffer can be corrected with `spell-line` (usually bound to Ctrl-$). To learn how your keys are set up, run the command `bindkey -b` (please look at Example 6-3 for more information). The correct shell variable can be set to `cmd` to correct the command name or to `all` to correct the entire line each time return is typed, and autocorrect can be set to correct the word to be completed before each completion attempt. Example 6-4 on page 119 shows how to set the spelling function and the output result.

Example 6-3  Summarize output from command `bindkey -b`

<table>
<thead>
<tr>
<th>Standard key bindings</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>^[B</code>               -&gt; backward-word</td>
</tr>
<tr>
<td><code>^[C</code>               -&gt; capitalize-word</td>
</tr>
<tr>
<td><code>^[D</code>               -&gt; delete-word</td>
</tr>
<tr>
<td><code>^[F</code>               -&gt; forward-word</td>
</tr>
<tr>
<td><code>^[H</code>               -&gt; run-help</td>
</tr>
<tr>
<td><code>^[I</code>               -&gt; downcase-word</td>
</tr>
<tr>
<td><code>^[N</code>               -&gt; history-search-forward</td>
</tr>
<tr>
<td><code>^[P</code>               -&gt; history-search-backward</td>
</tr>
<tr>
<td><code>^[R</code>               -&gt; toggle-literal-history</td>
</tr>
<tr>
<td><code>^[S</code>               -&gt; spell-word</td>
</tr>
<tr>
<td><code>^[U</code>               -&gt; upcase-word</td>
</tr>
<tr>
<td><code>^[W</code>               -&gt; copy-region-as-kill</td>
</tr>
<tr>
<td><code>^[L</code>               -&gt; insert-last-word</td>
</tr>
<tr>
<td><code>^[b</code>               -&gt; backward-word</td>
</tr>
<tr>
<td><code>^[c</code>               -&gt; capitalize-word</td>
</tr>
<tr>
<td><code>^[d</code>               -&gt; delete-word</td>
</tr>
<tr>
<td><code>^[f</code>               -&gt; forward-word</td>
</tr>
<tr>
<td><code>^[h</code>               -&gt; run-help</td>
</tr>
<tr>
<td><code>^[i</code>               -&gt; downcase-word</td>
</tr>
<tr>
<td><code>^[n</code>               -&gt; history-search-forward</td>
</tr>
<tr>
<td><code>^[p</code>               -&gt; history-search-backward</td>
</tr>
<tr>
<td><code>^[r</code>               -&gt; toggle-literal-history</td>
</tr>
<tr>
<td><code>^[s</code>               -&gt; spell-word</td>
</tr>
<tr>
<td><code>^[u</code>               -&gt; upcase-word</td>
</tr>
<tr>
<td><code>^[w</code>               -&gt; copy-region-as-kill</td>
</tr>
<tr>
<td><code>^[?</code>               -&gt; backward-delete-word</td>
</tr>
<tr>
<td><code>^[X^[X</code>            -&gt; exchange-point-and-mark</td>
</tr>
<tr>
<td><code>^[X**</code>             -&gt; expand-glob</td>
</tr>
<tr>
<td><code>^[X$</code>              -&gt; expand-variables</td>
</tr>
<tr>
<td><code>^[XG</code>              -&gt; list-glob</td>
</tr>
<tr>
<td><code>^[Xg</code>              -&gt; list-glob</td>
</tr>
<tr>
<td><code>^[Xn</code>              -&gt; normalize-path</td>
</tr>
<tr>
<td><code>^[XN</code>              -&gt; normalize-path</td>
</tr>
<tr>
<td><code>^[X?</code>              -&gt; normalize-command</td>
</tr>
<tr>
<td><code>^[X^[I</code>            -&gt; complete-word-raw</td>
</tr>
<tr>
<td><code>^[X^[D</code>            -&gt; list-choices-raw</td>
</tr>
</tbody>
</table>
Arrow key bindings
- down -> down-history
- up -> up-history
- left -> backward-char
- right -> forward-char

Example 6-4  Use of tcsh spelling correction capability

> set correct=cmd
> ls -lt /etc/a*

CORRECT>ls -lt /etc/a* (y|n|e|a)? yes

-rw-r--r-- 1 root  system  20480 Feb 21 14:51 /etc/aliases.db
-rw-r--r-- 1 root  system  0 Feb 02 12:59 /etc/aliases.dir
-rw-r--r-- 1 root  system  1024 Feb 02 12:59 /etc/aliases.pag
-rw-r--r-- 1 root  system  1329 Feb 02 12:58 /etc/aliases

Completion and listing

The shell is often able to complete words when given a unique abbreviation. Type part of a word (for example ls /usr/lost) and hit the Tab key to run the complete-word editor command. The shell completes the file name /usr/lost to /usr/lost+found/, replacing the incomplete word with the complete word in the input buffer. (Note the terminal /; completion adds a / to the end of completed directories and a space to the end of other completed words to speed typing and provide a visual indicator of successful completion. The addsuffix shell variable can be unset to prevent this.) If no match is found (perhaps /usr/lost+found does not exist), the terminal bell rings. If the word is already complete (perhaps there is a /usr/lost on your system, or perhaps you were thinking too far ahead and typed the whole thing) a / or space is added to the end if it is not already there.

Completion works anywhere in the line, not just at the end; completed text pushes the rest of the line to the right. Completion in the middle of a word often results in leftover characters to the right of the cursor, which need to be deleted.

This command line completion is shown in Example 6-5.

Example 6-5  Use of tab and CTRL-D complete a command line

> ls -lt /usr/l (Type CTRL-D instead of ENTER key to finish sentence).

> ls -lt /usr/l

> ls -lt MYDIR/SUBDIR/ (Type /MYDIR/SU+TAB key, the sentence will be finished).
Command line editing

Command-line input can be edited using key sequences much like those used in GNU Emacs or vi. The editor is active only when the edit shell variable is set, which it is by default in interactive shells. The bindkey builtin can display and change key bindings. Emacs-style key bindings are used by default (unless the shell was compiled otherwise; see the version shell variable), but bindkey can change the key bindings to vi-style bindings all at once. Refer to Example 6-9 on page 125 for additional information on how to personalize your environment.

For a complete reference of the tcsh shell, please refer to:

zsh

The zsh is a UNIX command interpreter (shell) usable as an interactive login shell and as a shell script command processor. Of the standard shells, zsh most closely resembles ksh, but includes many enhancements. Zsh has command line editing, builtin spelling correction, programmable command completion, shell functions, and a history mechanism.

Some of the key features of the zsh shell are:

- Command line editing:
  - Programmable completion, which incorporates the ability to use the power of zsh globbing
  - Multi-line commands editable as a single buffer
  - Variable editing
  - Command buffer stack
  - Inline expansion of variables and history commands

- Globbing, which is a very powerful feature (see Example 6-6 on page 121, Example 6-7 on page 122, and Example 6-8 on page 122). It includes:
Recursive globbing
- File attribute qualifiers
- Full alternation and negation of patterns
  - Handling of multiple redirections (simpler than `tee`)
  - Path expansion
  - Spelling correction

Example 6-6  Using zsh with globbing

```
luix-2% ls /tmp/FVWM98/fvwm98-2.0.43b.orig/fvwm  (Standard way)
Makefile        colormaps.c     functions.c     menus.o         read.c
Makefile.in     colormaps.o     functions.o     misc.c          read.o
add_window.c    colors.c        fvwm.c          misc.h          resize.c
add_window.o    colors.o        fvwm.h          misc.o          resize.o
alpha_header.h  complex.c       fvwm.man        module.c        screen.h
bindings.c      complex.o       fvwm.o          module.h        style.c
bindings.o      decorations.c   fvwm95          module.o        style.o
borders.c       decorations.o   fvwm95.man      move.c          sun_headers.h
borders.o       events.c        icons.c         move.o          virtual.c
builtins.c      events.o        icons.o         parse.h         virtual.o
builtins.o      focus.c         menus.c         placement.c      windows.c
buttons.h       focus.o         menus.h         placement.o      windows.o
luix-2%
```

```
lunix-2% setopt extendedglob (Enabling globbing)
lunix-2% ls -l /tmp/FVWM98/fvwm98-2.0.43b.orig/fvwm/^*.o  (Negates all .o files)
```

```
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/Makefile
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/Makefile.in
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/add_window.c
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/alpha_header.h
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/bindings.c
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/borders.c
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/builtins.c
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/buttons.h
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/colormaps.c
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/colors.c
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/complex.c
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/decorations.c
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/events.c
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/focus.c
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/functions.c
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/fvwm.c
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/fvwm.h
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/fvwm.man
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/fvwm95
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/fvwm95.man
/tmph/FVWM98/fvwm98-2.0.43b.org/fvwm/icons.c
```
Example 6-7  Using zsh with grouping
luix-2% ls -c /tmp/FVWM98/fvwm98-2.0.43b.orig/fvwm/(style|module).*
/tmp/FVWM98/fvwm98-2.0.43b.orig/fvwm/module.c
/tmp/FVWM98/fvwm98-2.0.43b.orig/fvwm/module.h
/tmp/FVWM98/fvwm98-2.0.43b.orig/fvwm/module.o
luix-2%

Example 6-8  Using zsh to find setuid files
luix-2% ls -1 /bin/s*(s)
-r-sr-xr-x 1 root system  8810 Jul 20 1999 /bin/script
-r-sr-xr-x 1 root system  6358 Jun 16 2000 /bin/setclock
-r-sr-xr-x 1 root security 32048 Aug 31 10:09 /bin/setgroups
-r-sr-xr-x 1 root security 19108 Aug 05 1999 /bin/setsenv
-r-sr-xr-x 1 root security 5190 Aug 05 1999 /bin/shell
-r-sr-xr-x 1 root security 17974 Aug 05 1999 /bin/su
-r-sr-xr-x 1 root security 79120 Aug 29 2000 /bin/sysck
luix-2%
For a complete reference for the zsh shell, please refer to:
http://sunsite.dk/zsh

or
http://www.zsh.org

Table 6-3 provides a comparison of the new AIX shell environments.

Table 6-3   AIX Toolbox for Linux Applications shell feature comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>bash</th>
<th>tcsh</th>
<th>zsh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command history</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Command alias</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Shell scripts</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Filename completion</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Command line editing</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Job control</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

6.2.1 Overview of shell startup files

When we login, the shell defines the user environment after reading the shell startup files. During the login process, the general characteristics of the user environment are defined by the values given to the environment variables; this environment is kept until the user logs off the system.

Login execution sequence

Regardless of what shell we are running, the /etc/environment and /etc/security/environ files are always read. Table 6-4 on page 124 and Table 6-5 on page 124 display the order in which the login execution sequence takes place.

The /etc/environment file sets up the user environment, such as the minimal search path, time zone, and language. This file is not a shell script type file and the only data format that it accepts is Name=<value>. We must understand that this file is read by all processes started by the init process and that it affects all login shells.

The /etc/security/environ file is an ASCII file that contains stanzas with the environment attributes for each individual user. Each stanza is identified by a user name and accepts the format Attribute=<value>. 

---

Chapter 6. User and administration differences  123
The user stanza in the /etc/security/environ file can have the following attributes:

- **usrenv**
  Defines environment variables (separated by commas) to be placed in the user environment at login time.

- **sysenv**
  Defines environment variables to be placed in the user protected state environment at login time. These variables are protected from access by unprivileged programs.

**Table 6-4 Login execution sequence for ksh, csh, and sh**

<table>
<thead>
<tr>
<th>Korn shell</th>
<th>C shell</th>
<th>Bourne shell</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/environment</td>
<td>/etc/environment</td>
<td>/etc/environment</td>
</tr>
<tr>
<td>/etc/security/environ</td>
<td>/etc/security/environ</td>
<td>/etc/security/environ</td>
</tr>
<tr>
<td>/etc/profile</td>
<td>/etc/csh.cshrc</td>
<td>/etc/profile</td>
</tr>
<tr>
<td>$HOME/.profile</td>
<td>/etc/csh.login</td>
<td>$HOME/.profile</td>
</tr>
<tr>
<td>$HOME/.kshrc</td>
<td>$HOME/.cshrc</td>
<td>$HOME/.login</td>
</tr>
</tbody>
</table>

**Table 6-5 Login execution sequence for bash, tcsh, and zsh**

<table>
<thead>
<tr>
<th>Bash shell</th>
<th>Tcsh shell</th>
<th>Z shell</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/environment</td>
<td>/etc/environment</td>
<td>/etc/environment</td>
</tr>
<tr>
<td>/etc/security/environ</td>
<td>/etc/security/environ</td>
<td>/etc/security/environ</td>
</tr>
<tr>
<td>/etc/profile</td>
<td>/etc/csh.cshrc</td>
<td>/etc/zshenv</td>
</tr>
<tr>
<td>$HOME/.bash_profile</td>
<td>/etc/csh.login</td>
<td>$HOME/.zshenv</td>
</tr>
<tr>
<td>$HOME/.bash_login</td>
<td>$HOME/.tcshrc</td>
<td>/etc/zprofile</td>
</tr>
<tr>
<td>$HOME/.profile</td>
<td>($HOME/.cshrc)</td>
<td>$HOME/.zprofile</td>
</tr>
<tr>
<td>($HOME/.bashrc)</td>
<td>$HOME/.history</td>
<td>/etc/zshrc</td>
</tr>
<tr>
<td></td>
<td>$HOME/.login</td>
<td>$HOME/.zshrc</td>
</tr>
<tr>
<td></td>
<td>$HOME/.cshdirs</td>
<td>/etc/zlogin</td>
</tr>
<tr>
<td></td>
<td>$HOME/zlogin</td>
<td>$HOME/zlogin</td>
</tr>
</tbody>
</table>
Command line editing in ksh

For Linux users that prefer the usage of the cursor keys for editing the command line (emacs-style), Example 6-9 and Example 6-10 show a .profile and a .kshrc that accomplish the same behavior on AIX and in our Toolbox environment.

Example 6-9 .profile and emacs style key binding

Add the following lines to your .profile in your home directory:

```bash
if [ -f $HOME/.kshrc -a -r $HOME/.kshrc ]; then
    ENV=$HOME/.kshrc                # set ENV if there is an rc file
    export ENV
    . $ENV
fi

alias -x __A=`echo "\020"` # up arrow = ^p = back a command
alias -x __B=`echo "\016"` # down arrow = ^n = down a command
alias -x __C=`echo "\006"` # right arrow = ^ = forward a character
alias -x __D=`echo "\002"` # left arrow = ^b = back a character
alias -x __H=`echo "\001"` # home = ^a = start of line
set -o emacs                            # emacs in-line editing mode
```

Example 6-10 .kshrc

Add the following lines to your .kshrc file in your home directory:

```bash
alias -x __A=`echo "\020"` # up arrow = ^p = back a command
alias -x __B=`echo "\016"` # down arrow = ^n = down a command
alias -x __C=`echo "\006"` # right arrow = ^ = forward a character
alias -x __D=`echo "\002"` # left arrow = ^b = back a character
alias -x __H=`echo "\001"` # home = ^a = start of line
set -o emacs                            # emacs in-line editing mode
```

Sample shell startup files

Example 6-11 on page 126 and Example 6-12 on page 126 show some examples for the tcsh and zsh shell startup files, such as .tcshrc and .zshrc.
Example 6-11 .tcshrc

```bash
setenv PATH
/usr/linux/bin:/opt/freeware/bin:/usr/local/bin:/usr/bin:/etc:/usr/sbin:
/etc:/usr/local/bin:/usr/bin/X11:/bin:/usr/sbin:/usr/ucb:${HOME}/bin:
/usr/bin/X11:/sbin:/opt/freeware/kde/bin:/opt/freeware/enlightenment/bin:
/opt/freeware/lib/xscreensaver:

setenv GCC_EXEC_PREFIX
/opt/cygnsus/aix43-000718/H-powerpc-ibm-aix4.3.3.0/lib/gcc-lib/
setenv PATH ${PATH}/opt/cygnsus/aix43-000718/H-powerpc-ibm-aix4.3.3.0/bin

setenv MOZILLA_HOME /opt/netscape
setenv MANPATH /opt/freeware/man:/opt/cygnsus/aix43-000718/man

# list of other places to look
set manpath = ( /usr/local/man /usr/local/X11R6/man /usr/local/X/man \
               /usr/local/gnu/man /usr/local/lang/man /usr/lang/man)

# only include if it exists
foreach mandir ( ${manpath} )
  if ( -d ${mandir} ) then
    setenv MANPATH ${MANPATH}:${mandir}
  endif
end
```

Example 6-12 .zshrc

```bash
#
# Generic .zshrc used in our AIX Toolbox for Linux Applications
# Environment

# Use hard limits, except for a smaller stack and no core dumps
unlimit
limit stack 8192
limit core 0
limit -s

umask 022

# Set up aliases

alias mv='nocorrect mv'       # no spelling correction on mv
alias cp='nocorrect cp'       # no spelling correction on cp
alias mkdir='nocorrect mkdir' # no spelling correction on mkdir
alias j=jobs
alias pu=pushd
alias po=popd
alias d='dirs -v'
alias h=history
alias grep=egrep
```
alias ll='ls -l'
alias la='ls -a'

# List only directories and symbolic links that point to directories
alias lsd='ls -ld *(-/DN)' # List only file beginning with "."
alias lsa='ls -ld .*'

# Shell functions
setenv() { export $1=$2 }  # csh compatibility

# Autoload all shell functions from all directories in $fpath that have the executable bit on
# (the executable bit is not necessary, but gives you an easy way to stop the autoloading of a particular shell function).
for dirname in $fpath
do
  autoload $dirname/*(.x:t)
done

# Global aliases -- These do not have to be at the beginning of the command line.
alias -g M='|more'
alias -g H='|head'
alias -g T='|tail'

manpath=(/usr/man /opt/freeware/man:/opt/cygnus/aix43-000718/man)
export MANPATH

# Filename suffixes to ignore during completion
fignore=(.o .c~ .old .pro)

# Set prompts
PROMPT='%m%# '  # default prompt
RPROMPT=' %~'  # prompt for right side of screen

# Some MAIL environment variables
export MAIL=/var/spool/mail/$USERNAME
MAILCHECK=300
HISTSIZE=200
DIRSTACKSIZE=20

# Set/unset shell options
setopt notify globdots correct pushdtohome cdablevars autolist
setopt correctall autocd recexact longlistjobs
Running Linux Applications on AIX

```bash
setopt autoresume histignoredup pushdsilent noclobber
setopt autopushd pushdminus extendedglob rcquotes mailwarning
unsetopt bgnice autoparamslash

# Setup some basic programmable completions. To see more examples
# of these, check Misc/compctl-examples in the zsh distribution.
compctl -g '*(-/) cd pushd
compctl -g '*(/)' rmdir dircmp
compctl -j -P % -x 's[[]] p[1]' -k signals -- kill
compctl -j -P % fg bg wait jobs disown
compctl -A shift
compctl -caF type whence which
compctl -F unfunction
compctl -a unalias
compctl -v unset typeset declare vared readonly export integer
compctl -e dicompctl -d enable

# Some nice key bindings
#bindkey '^X^Z' universal-argument ' ' magic-space
#bindkey '^X^A' vi-find-prev-char-skip
#bindkey '^Z' accept-and-hold
#bindkey -s '\M-/' \\
#bindkey -s '\M-=' |
# bindkey -v            # vi key bindings
bindkey -e            # emacs key bindings
bindkey ' ' magic-space  # also do history expansion on spacesable
```

### 6.3 Commands and syntax differences

In this section, we want to point out some similarities and differences in the commands and their usage on AIX compared to a native Linux environment. Appendix B, “Differences in commands” on page 187 lists the commands that differ in syntax. Detailed descriptions of their syntax are included in the AIX Toolbox for Linux Applications CD and the Web site:

In some cases, the differences are bigger than just a changed syntax, and Linux commands have to be replaced by completely other AIX commands. Table 6-6 gives examples of some Linux commands and their AIX equivalent.

<table>
<thead>
<tr>
<th>Linux command</th>
<th>AIX command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dmesg</td>
<td>errpt</td>
<td>Displays the system control messages and errors from the kernel buffer.</td>
</tr>
<tr>
<td>free</td>
<td>lsp -a</td>
<td>Displays characteristics of swap or paging space.</td>
</tr>
<tr>
<td>useradd</td>
<td>mkuser</td>
<td>Creates a new user account.</td>
</tr>
<tr>
<td>pvdisplay</td>
<td>lspv</td>
<td>Displays the physical volumes.</td>
</tr>
<tr>
<td>vgscan</td>
<td>lsvg</td>
<td>Displays the volume groups.</td>
</tr>
</tbody>
</table>

In general, commands that come from other sources than AIX should not be used for system administration. This is because administration commands manipulate system files, and Linux has different relative paths for some system files and different parameters compared to AIX. Hence, it is important to be aware of these differences before doing system administration tasks. To avoid further conflicts, it is recommended to use AIX commands for these type of tasks. See Section 6.6, “System files differences” on page 147 for more details.

Also, AIX has some commands that are unique to it and are substantial for AIX system administrations, such as the System Activity Report (sar) and running diagnostics (diag). The sar command collects, reports and saves system activity information. This is significantly useful for auditing and accounting purposes on your system. The diag command is another helpful tool in the AIX system. This tool performs hardware problem determination and gives a summary and recommendations about the cause of the problem based upon the error reports. The diag command has a menu driven interface (refer to Figure 6-19 on page 130) inspired by the SMIT utility. Here, you can do diagnostic routines, service aids, and resource selection without the hassle of command lines.
These are just some of the essential commands from the AIX operating system that function as service aids that enable you to more easily administer your system.

### 6.3.1 AIX and AIX Toolbox commands differences

The AIX Toolbox contains many Linux commands, and most of the commands come from GNU packages. These commands can be categorized into two types:

- **New commands added to the system**

  Commands that do not have any conflicts with AIX commands are placed in `/opt/freeware/bin` with links to `/usr/bin`. Example 6-13 shows examples for this case.

#### Example 6-13  Non-conflicting commands

```
/usr/bin > ls -1
lrwxrwxrwx 1 root system  30 Feb 8 10:14 aclocal ->
./../opt/freeware/bin/aclocal
lrwxrwxrwx 1 root system  31 Feb 8 10:12 autoconf ->
./../opt/freeware/bin/autoconf
lrwxrwxrwx 1 root system  33 Feb 8 10:12 autoheader ->
./../opt/freeware/bin/autoheader
```
Same commands on Linux and AIX, but with different syntax

Toolbox commands that already exist in AIX are also placed in /opt/freeware/bin but linked to /usr/linux/bin to avoid conflicts. To use these commands rather than the AIX versions, you can execute them with their complete relative path. Example 6-14 shows some commands that are placed in /usr/linux/bin with their corresponding links.

Example 6-14  Conflicting commands

```bash
/usr/linux/bin > ls -l
```

```bash
total 4
lrwxrwxrwx 1 root system 30 Feb 8 10:14 awk ->
../.../opt/freeware/bin/gawk
lrwxrwxrwx 1 root system 32 Feb 8 11:20 captoinfo ->
../../opt/freeware/bin/captoinfo
lrwxrwxrwx 1 root system 31 Feb 8 10:46 chgrp ->
../../opt/freeware/bin/chgrp
```

These Linux commands are very similar to the corresponding AIX commands except for some differences in syntax and attributes. To give some examples, here are some commonly-used commands and their differences:

- **who**
  
  This command identifies the users that are currently logged in.

  - AIX syntax
    
    ```bash
    who [ -a | -b | -d | -i | -l | -m | -p | -q | -r | -s | -t | -w | -A | -H | -T ] [File]
    ```

  - Linux syntax
    
    ```bash
    who [option] ... [File | Arg1 | Arg2 ]
    ```

  The -l option is present in both implementations, but has different behaviors. On AIX, -l will list any login process, while on Linux it will try to resolve all host names via DNS.

- **uname**

  Displays information about the current operating system.

  - AIX syntax
    
    ```bash
    uname [ -a | -x | -SName ] [ -1 ] [ -m ] [ -M ] [ -n ] [ -p ] [ -r ] [ -s ] [-TName ] [ -u ] [ -v ]
    ```
Linux syntax

```
uname [OPTION]...
```

This command is featured in both (same flags) but produces different behaviors, as shown in Table 6-7.

<table>
<thead>
<tr>
<th>Flag</th>
<th>In AIX</th>
<th>In Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a</td>
<td>Displays all information as specified with the -m, -n, -r, -s, and -v flags Cannot be used with the -x or -SName flag. If the -x flag is specified with the -a flag, the -x flag overrides it.</td>
<td>-a, --all Prints all information</td>
</tr>
</tbody>
</table>

For a more detailed explanation of the syntax differences, you may want to compare the online manual pages for AIX and Linux. The man pages can give you extensive online help for any command on your system. To view the man pages for the Linux commands, you can change the MANPATH variable to have /opt/freeware/man first. You can also view the quick reference manual through the command line. For AIX commands, use the -h option (refer to Example 6-15).

Example 6-15  tar -h option

```
/ > tar -h
Usage: tar -{c|r|t|u|x} [-BdFhilmopsvw]
[-Number] [-fFile]
[-bBlocks] [-S [Feet] [Feet @Density] [Blocksb]]
```

For Linux commands, use the --help option (refer to Example 6-16).

Example 6-16  tar usage in Linux

```
/ > /usr/linux/bin/tar --help
GNU `tar' saves many files together into a single tape or disk archive, and can restore individual files from the archive.

Usage: /usr/linux/bin/tar [OPTION]... [FILE]...

If a long option shows an argument as mandatory, then it is mandatory for the equivalent short option also. Similarly for optional arguments.

Main operation mode:
- t, --list list the contents of an archive
- x, --extract, --get extract files from an archive
- c, --create create a new archive
- d, --diff, --compare find differences between archive and file system
- r, --append append files to the end of an archive
```
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-u, --update</td>
<td>only append files newer than copy in archive</td>
</tr>
<tr>
<td>-A, --catenate</td>
<td>append tar files to an archive</td>
</tr>
<tr>
<td>--concatenate</td>
<td>same as -A</td>
</tr>
<tr>
<td>--delete</td>
<td>delete from the archive (not on mag tapes!)</td>
</tr>
</tbody>
</table>

To get the current list of all AIX commands that have a different syntax than on Linux, see Appendix B, “Differences in commands” on page 187. A detailed documentation of the commands differences is also included with the AIX Toolbox for Linux Applications CD and Web site; you may refer to them for an updated list.

**Changing the command defaults**

Although AIX commands are quite easy to comprehend, some Linux administrators and users might prefer using the Linux binaries for the sake of familiarity. Here are some options that you can use to make a Linux command the default command. Note that the following options can cause errors in some AIX utilities, such as the System Management Interface Tool (to know more about SMIT, refer to Section 6.4, “Administration differences” on page 134).

- For individual commands, you can add a command alias containing the command and its full path to the shell startup files /etc/profile or $HOME/.profile (if you are using the Korn shell). In Example 6-17, an alias for the `ls` command has been added to user janeethe’s profile to execute the Linux version for that command.

**Example 6-17  Adding an alias for the ls command**

```bash
/home/janethe > vi .profile
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11:/sbin:.
export PATH

if [ -s "$MAIL" ] # This is at Shell startup. In normal
    then echo "$MAILMSG" # operation, the Shell checks
fi # periodically.

alias ls=/usr/linux/bin/ls
```

For the changes to take effect immediately without re-logging, execute the command:

```
# . $HOME/.profile
```

- To execute the Linux version of all commands by default instead of the original AIX version, change the PATH variable and set /usr/linux/bin first. See Example 6-18 on page 134 for details.
Example 6-18  Changing the PATH variable

```
/usr/bin > export PATH=/usr/linux/bin:$PATH
```

To make the changes permanent, edit the /etc/profile file and set the PATH to what is shown in Example 6-18.

### 6.4 Administration differences

UNIX-based operating systems use different kinds of tools for system administration. Each operating system contains several tools that combine many administrative tasks. These tools have a user-friendly interface, either a text mode and/or a graphical user interface. Each of these system administration utilities may have different schemes, features, and interfaces, but all of them are used for the same purpose and functionality: to administer the running system.

All these tools do not prevent you from using UNIX commands, such as `passwd` and `adduser` directly or from editing the system configuration files manually. But since the distribution-supplied tools check dependencies with other affected system files, they make administrative work substantially easier and safer. Table 6-8 presents a summary of some well-known administration tools.

**Table 6-8  Linux distribution and their administration tools**

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Administration tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>RedHat Linux</td>
<td>Linuxconf</td>
</tr>
<tr>
<td>SuSE Linux</td>
<td>YaST (text mode), YaST2 (GUI)</td>
</tr>
<tr>
<td>AIX</td>
<td>SMIT</td>
</tr>
</tbody>
</table>

RedHat Linux developed Linuxconf as its administration utility. Linuxconf is a user interface utility that allows you to do configuration tasks; it is also an activator. It has text, web, and graphical interfaces that can be used, depending on the current environment. Linuxconf is based mostly on configuration files and can run different system administration tasks, such as configuring TCP/IP, defining file systems, activate daemons, and more. An example of the Linuxconf graphical interface can be seen in Figure 6-20 on page 135.
SuSE Linux, however, has designed and integrated “Yet another Setup Tool” or YaST (YaST2 for the GUI interface) as its system and device configuration tool. Besides system administration, it can also run tasks such as hard-disk partitioning, Linux operating system updates, and package management. Refer to Figure 6-21 on page 136 for more details.
The counterpart for both Linuxconf and YaST in the AIX operating system is the System Management Interface Tool or SMIT. SMIT is an interactive and extensible screen-oriented command interface. It prompts users for the information needed to construct command strings and presents appropriate predefined selections or run time defaults (when available). This helps users from many different backgrounds to avoid extra work or errors, such as remembering complex command syntax, parameter values, system command spelling, or custom shell path names. SMIT does everything from configuring interfaces to partitioning disks, to setting up Internet services and backing up your system.

**Important:** While it might be possible to recompile and run other Linux system administration utilities, at this time only SMIT should be used for system administration. Using other utilities will create conflicts because of different system files. To know more about the system files differences, refer to Section 6.6, “System files differences” on page 147.

The SMIT facility runs in one of two interfaces: text based or graphical interface. The first panel displayed, after you enter the `smit` command, is the main menu, which is shown in Figure 6-22 on page 137. In the SMIT interface, main menu selections lead to submenus, which narrows down the scope of choices. To skip the main menu and directly access a submenu or dialog, you can use the `smit` command with a fast path parameter.
Fast path is a unique feature of the SMIT utility. It is a shortcut method to access a certain menu directly. At any menu in SMIT, you can show the fast path parameter by pressing F8 or by choosing Show->Fast Path (if you are using the graphical interface).

To access a fast path, use the following syntax:

```
# smit <fast path name>
```

The IBM Certification Guide: AIX 4.3 System Administration, SG24-5129 provides a quick reference to and more information on all the SMIT fast path commands. Table 6-9 lists the SMIT Security and Users menu and its submenus, followed by their associated fast path.

Table 6-9  SMIT menu examples and their corresponding fast paths

<table>
<thead>
<tr>
<th>Menu Name</th>
<th>Fast Path name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security and Users</td>
<td>security</td>
</tr>
<tr>
<td>Users</td>
<td>users</td>
</tr>
<tr>
<td>Add a User</td>
<td>mkuser</td>
</tr>
<tr>
<td>Change a User's Password</td>
<td>passwd</td>
</tr>
<tr>
<td>Change/Show Characteristics of a User</td>
<td>chuser</td>
</tr>
<tr>
<td>Lock/Unlock a User's Account</td>
<td>lockuser</td>
</tr>
</tbody>
</table>
Linux administrators may quickly find that there are differences in configuring and administering the current AIX operating system compared to a Linux system because of syntax and utility differences. Since Linux and AIX systems use different system files, it is highly recommended that AIX commands and utilities should be used for system administration (this will be further explained in Section 6.6.1, “File system definitions on AIX and Linux” on page 148). The system administrator should understand the concepts behind the tasks performed in the AIX operating system, as well as understand the tools provided for system management.

Table 6-10 lists a summary of some basic SMIT tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>smitty (text-based)</th>
<th>smit (GUI interface)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter SMIT</td>
<td>Type the command smitty</td>
<td>Type the command smit</td>
</tr>
<tr>
<td>Exit SMIT</td>
<td>F10</td>
<td>F10 or Exit SMIT option from the Exit menu</td>
</tr>
<tr>
<td>Show Command</td>
<td>F6</td>
<td>F6 or Click Command option from Show menu</td>
</tr>
<tr>
<td>Show Fastpath</td>
<td>F8</td>
<td>F8 or Click FastPath option from Show menu</td>
</tr>
</tbody>
</table>

Each dialog in SMIT builds and executes a version of a standard command. Also, each task has a Help option that gives detailed information about the menu or the dialog. Let us take the example of user administration.

One of the basic tasks of a system administrator is to administer user accounts. For example, to add a user, you can select Main Menu -> Security and Users -> Users -> Add a User in SMIT or you can enter the following command:

```
# smit mkuser
```
The mkuser is an example of a fast path parameter. It takes you directly to the menu Add a User without going through the other submenus. But either way will allow you to go to the panel for adding a user, as you can see in Figure 6-23. The SMIT User Account Menu is almost the same as Linuxconf (shown in Figure 6-24 on page 140) and YaST (shown in Figure 6-25 on page 141). After adding all the needed parameters and value, press Enter to execute the task. A dialog panel will show the results.

Figure 6-23   SMIT user account menu
Figure 6-24  Linuxconf user account menu
6.5 Boot process differences

In this section, we will describe the differences in the boot process of a Linux operating system compared to AIX. The purpose is to explain the corresponding AIX procedure to Linux system administrators. In general, during the boot process, the system tests hardware, loads and runs the operating system, and configures devices. We will start our description after the kernel is already loaded in the memory, so we will not look into any details of boot records on hard drives or the way the kernel gets loaded.

For completeness sake, we first describe the boot process on a native Linux system; this will make it easier to understand the way it works on AIX.

6.5.1 Linux boot process

On a typical Intel-based system, the following processes occur in this order:
1. The BIOS is run.
2. The hardware configuration is checked.
3. A boot loader (like LILO, the Linux LOader) is executed.
4. The Linux kernel is booted.
5. The kernel takes over control.

The kernel searches for the init executable in several locations (/sbin is a very common location) and executes it. init becomes the first process on the system and is the “father” or “grandfather” of all subsequent processes. init runs the /etc/rc.d/rc.sysinit script, which does the basic system initialization, such as setting up an initial environment, starting swapping, checking file systems, and so on. /etc/rc.d/rc.sysinit reads several other files for information, for example, /etc/sysconfig/network or /etc/sysconfig/clock. Then init processes the /etc/inittab file, which describes which services are to be started in each runlevel and starts the default runlevel. /etc/rc.d/rc and /sbin/update are executed whenever a runlevel starts. The rc script starts all necessary background processes and executes certain scripts in the directory associated with the runlevel, /etc/rc.d/rc<x>.d, where <x> is a number from 1 to 6. All start scripts in that directory, whose names start with an S, are executed.

If the runlevel of a running system is changed with the init <x> command, all kill scripts, whose name starts with a K, in /etc/rc.d/rc<y>.d with <y> as the previous runlevel, are executed before the start scripts of the new runlevel are executed.

After this, /etc/inittab forks a getty process for each virtual console. The default runlevel is set with an entry like id:3:initdefault: in /etc/inittab. Individual additions to be executed at boot time can be placed in /etc/rc.d/rc.local, which will be executed after the other initializations are completed.

The scripts in /etc/rc.d/rc<x>.d are links to /etc/rc.d/init.d. All start scripts in /etc/rc.d/rc<x>.d are called by rc with the parameter start, all stop scripts with the parameter stop. This allows for both the start and the stop script in /etc/rc.d/rc<x>.d for a certain service, for example, httpd, to be linked to the same script (in /etc/rc.d/init.d, /etc/rc.d/init.d/httpd in the example). You can also call the scripts directly from the command line by using:

```
# /etc/rc.d/init.d/httpd start
```

The meaning of the seven runlevels is:

- 0: Halt
- 1: Single-user mode
- 2: Multi-user mode, without networking
- 3: Full multi-user mode
- 4: Not used
- 5: Full multi-user mode with an X-based login panel
- 6: Reboot
6.5.2 AIX boot process

The AIX boot process is documented in the *AIX System Management Guide: Operating Systems and Devices*, found at:

http://9.53.35.177/techlib/manuals/adoclib/aixbman/baseadmn/undersys.htm

It is divided into three phases:

- Read Only Storage (ROS) Kernel Init phase
- Base Device Configuration phase
- System Boot phase

Phase One, the kernel initialization phase, ends with the start of the init process.

Then, in Phase Two, the base device configuration begins, as shown in Figure 6-26 on page 144. The init process starts the rc.boot script. Phase One of the rc.boot script performs the base device configuration, and it includes the following steps:

1. The boot script calls the restbase program to build the customized Object Database Manager (ODM) database in the RAM file system from the compressed customized data.
2. The boot script starts the configuration manager, which accesses Phase One configuration rules to configure the base devices.
3. The configuration manager starts the sys, bus, disk, SCSI, and the Logical Volume Manager (LVM) and rootvg volume group (RVG) configuration methods.
4. The configuration methods load the device drivers, create special files, and update the customized data in the ODM database.
Figure 6-26   AIX boot process - Phase One
Phase Three of the boot process is the system boot phase. As shown in Figure 6-27 on page 146, the following steps are executed:

1. The init process starts Phase Two execution of the rc.boot script. Phase Two of rc.boot includes the following steps:
   a. Call the ipl_varyon program to vary on the rootvg volume group (RVG).
   b. Mount the hard disk file systems onto the RAM file system.
   c. Run `swapon` to start paging.
   d. Copy the customized data from the ODM database in the RAM file system to the ODM database in the hard disk file system.
   e. Unmount temporary mounts of hard disk file systems and then perform permanent mounts of root, /usr, and /var.
   f. Exit the rc.boot script.

2. After Phase Two of rc.boot, the boot process switches from the RAM file system to the hard disk root file system. The init process executes the processes defined by records in the `/etc/inittab` file. One of the instructions in the `/etc/inittab` file executes Phase Three of the rc.boot script, which includes the following steps:
   a. Mount the `/tmp` hard disk file system.
   b. Start the configuration manager (Phase Two) to configure all remaining devices.
   c. Use the `savebase` command to save the customized data to the boot logical volume.
   d. Exit the rc.boot script.
Figure 6-27  AIX boot process - Phase Two
On AIX, the /etc/inittab file should generally not be edited manually. This is because errors made during editing might result in a system that does not boot anymore; system recovery would have to be performed. Instead, there are some commands, such as chitab, mkitab, and rmitab, that modify the /etc/inittab file. Also, certain operations in SMIT can result in an (intended) change of /etc/inittab.

The syntax of entries in /etc/inittab is Identifier:RunLevel:Action:Command. The following conventions hold for RunLevel:

- **0-1**: Reserved for the future use by the operating system.
- **2**: Contains all of the terminal processes and daemons that are run in the multiuser environment. In the multiuser environment, the /etc/inittab file is set up so that the init command creates a process for each terminal on the system. The console device driver is also set to run at all run levels, so the system can be operated with only the console active.
- **3-9**: Can be defined according to the user's preferences.

The runlevel of a running AIX system can be changed using the init or telinit command. In general, runlevels are rarely used on AIX. This might change in the future, as AIX 5L also provides /etc/rc.d/rc<x>.d directories, which hold start or kill scripts that are processed just as they are on Linux systems.

### 6.6 System files differences

This section shows that in spite of the fact that AIX and Linux are both UNIX-based operating systems, there are some differences in the way they structure systems resources and configuration files. Table 6-11 points out some of these configuration files and their differences.

<table>
<thead>
<tr>
<th>Description</th>
<th>AIX</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>File system definitions</td>
<td>/etc/filesystems</td>
<td>/etc/fstab</td>
</tr>
<tr>
<td>Encrypted passwords</td>
<td>/etc/security/passwd</td>
<td>/etc/shadow</td>
</tr>
<tr>
<td>Default su log</td>
<td>/var/adm/sulog</td>
<td>/var/log/messages</td>
</tr>
</tbody>
</table>

In order to show some these differences in greater detail, we will use the file system definition file as an example in the next section.
6.6.1 File system definitions on AIX and Linux

This section explains how definitions for file systems on AIX compare to definitions on a typical Linux system.

Linux file system definitions
On Linux systems, the /etc/fstab file contains descriptive information about the various file systems. It is recommended that this file not be edited manually, but be edited using administration tools like YaST or linuxconf. Each file system is described on a separate line; fields on each line are separated by tabs or spaces. The order of records in fstab is important because fsck, mount, and umount sequentially iterate through fstab.

The syntax of the entries in /etc/fstab is:

```
fs_spec fs_file fs_vfstype fs_mntops fs_freq fs_passno
```

where

- `fs_spec` describes the block special device or remote file system to be mounted (like /dev/cdrom, /dev/sdb7, or my.host.net:/directory).
- `fs_file` describes the mount point for the file system (like /, /usr, or /var).
- `fs_vfstype` describes the type of the file system (like ext2, msdos, or nfs; see /proc/filesystems on a Linux system for a list of file system types supported by the installed kernel).
- `fs_mntops` describes the mount options for the file system (like noauto or user; see also the mount command).
- `fs_freq` is used for the dump command.
- `fs_passno` is used by the command fsck to determine the order in which file system checks are done at reboot time.

Example 6-19 on page 149 shows a sample /etc/fstab file.
Example 6-19  Sample /etc/fstab file

/dev/hda1  /boot  ext2  defaults 1 2
/dev/hda2  swap  swap  defaults 0 2
/dev/hda3  /  ext2  defaults 1 1
/dev/hda4  /local  ext2  defaults 1 2
proc  /proc  proc  defaults 0 0
usbdevfs  /proc/bus/usb  usbdevfs  defaults 0 0
devpts  /dev/pts  devpts  defaults 0 0
/dev/cdrom  /cdrom  auto  ro,noauto,user,exec 0 0
/dev/fd0  /floppy  auto  noauto,user 0 0

AIX file system definitions

All information about the file systems is centralized in the /etc/filesystems file on AIX. This file should not be edited manually, but only with the appropriate administration commands and tools, such as mkfs or SMIT. The /etc/filesystems file is organized into stanza names, which are file system names and contents that are attribute-value pairs specifying characteristics of the file system. The file systems file serves two purposes:

- It documents the layout characteristics of the file systems.
- It frees the person who sets up the file system from having to enter and remember items such as the device where the file system resides, because this information is defined in the file.

Each stanza names the directory where the file system is normally mounted. The file system attributes specify all the parameters of the file system. The attributes currently used are:

- **account** Used by the dodisk command to determine the file systems that are processed by the accounting system. This value can be either True or False.

- **boot** Used by the mkfs command to initialize the boot block of a new file system. This specifies the name of the load module to be placed into the first block of the file system.

- **check** Used by the fsck command to determine the default file systems to be checked. The True value enables checking while the False value disables checking. If a number, rather than the True value, is specified, the file system is checked in the specified pass of checking. Multiple pass checking, described in the fsck command, permits file systems on different drives to be checked in parallel.

- **dev** Identifies, for local mounts, either the block special file where the file system resides or the file or directory to be mounted. System management utilities use this attribute to map file system names to
the corresponding device names. For remote mounts, it identifies the file or directory to be mounted.

**mount**  
Used by the `mount` command to determine whether this file system should be mounted by default. The possible values of the mount attribute are:

- **automatic**  
  Automatically mounts a file system when the system is started. For example, in the sample file, the root file system line is the mount=automatic attribute. This means that the root file system mounts automatically when the system is started. The True value is not used, so `mount all` does not try to mount it, and `umount all` does not try to unmount it. The False value is also not used, because certain utilities, such as the `ncheck` command, normally avoid file systems with a value of mount=False.

- **False**  
  This file system is not mounted by default.

- **readonly**  
  This file system is mounted as read-only.

- **True**  
  This file system is mounted by the `mount all` command. It is unmounted by the `umount all` command. The `mount all` command is issued during system initialization to automatically mount all such file systems.

- **nodename**  
  Used by the `mount` command to determine which node contains the remote file system. If this attribute is not present, the mount is a local mount. The value of the nodename attribute should be a valid node nickname. This value can be overridden with the `mount -n` command.

- **size**  
  Used by the `mkfs` command for reference and to build the file system. The value is the number of 512-byte blocks in the file system.

- **type**  
  Used to group related mounts. When the `mount -t String` command is issued, all of the currently unmounted file systems with a type attribute equal to the String parameter are mounted.

- **vfs**  
  Specifies the type of mount. For example, `vfs=nfs` specifies the virtual file system being mounted is an NFS file system. All types of virtual file systems are described in the `/etc/vfs` file.

- **vol**  
  Used by the `mkfs` command when initializing the label on a new file system. The value is a volume or pack label using a maximum of six characters.
The LVName must be the full path name of the file system logging logical volume name to which log data is written as this file system is modified. This is only valid for journaled file systems.

Example 6-20 shows a sample /etc/filesystems file.

```plaintext
Example 6-20  Sample /etc/filesystems file

* File system information
*
default:
  vol      = "AIX"
  mount    = false
  check    = false
/
  dev      = /dev/hd4
  vol      = "root"
  mount    = automatic
  check    = true
  log      = /dev/hd8
/home:
  dev      = /dev/hd1
  vol      = "u"
  mount    = true
  check    = true
  log      = /dev/hd8
/home/joe/1:
  dev      = /home/joe/1
  nodename = vance
  vfs      = nfs
/usr:
  dev      = /dev/hd2
  vol      = "usr"
  mount    = true
  check    = true
  log      = /dev/hd8
/tmp:
  dev      = /dev/hd3
  vol      = "tmp"
  mount    = true
  check    = true
  log      = /dev/hd8
```
This appendix provides us with a list of Linux C runtime APIs, Linux Standard Base (LSB) library functions, and changes available in AIX 5L 5.1 that were placed into libc.a to provide additional Linux functionality and compatibility.
Linux-compatible APIs and library functions

Listed below are Linux-compatible APIs and LSB library functions as a reference for programmers, especially when porting Linux applications to AIX. For more details and up-to-date information, see the documentation for the AIX Toolbox for Linux Applications at:


This section is based on:
- AIX Version 4.3.3 (unless otherwise stated)
- Linux kernel Version 2.2.14
- GNU libc Version 2.1.4
- Open Group Single UNIX Specification Version 2 (SUSv2)
- Linux Standard Base specification 0.2pre

A good source of information regarding the Linux Standard Base (LSB) can be found at:

http://www.linuxbase.org/

Linux-compatible APIs

The number of UNIX-based operating systems has grown over the years, and the system calls and their parameters are not unique. One of the goals in writing UNIX programs is to make them as portable as possible across all UNIX-based operating systems. Obviously, this situation is not possible. However, most of the original UNIX system calls have not changed, so if you try to use these calls, you should be all right.

The list of APIs described here was obtained from the system call table (syscalls.h) in the i386 port of Linux, and includes APIs and library functions that do not exist on all Linux ports, and some of them are not listed in the Linux Standard Base (LSB). We are providing them in this document as a quick reference. Table A-1 on page 155 describes the different groups of APIs regarding compatibility and implementation.

For Linux system calls information, please refer to the following Web sites:

http://howto.tucows.com/LDP/LDP/lpg/node1.html

Regarding AIX system calls, please refer to Kernel Extensions and Device Support Programming Concepts, found at:

**Table A-1** Different groups of APIs

<table>
<thead>
<tr>
<th>Group of APIs</th>
<th>Listed in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux-compatible APIs compatible with AIX</td>
<td>Table A-2</td>
</tr>
<tr>
<td>Linux-compatible APIs not available on AIX</td>
<td>Table A-3 on page 159</td>
</tr>
<tr>
<td>Linux-compatible APIs introduced in AIX 5L 5.1</td>
<td>Table A-4 on page 161</td>
</tr>
<tr>
<td>Linux-compatible APIs available on AIX but not 100% compatible</td>
<td>Table A-5 on page 161</td>
</tr>
</tbody>
</table>

**Table A-2** Compatible APIs

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access</td>
<td>Checks user's permissions for a file.</td>
</tr>
<tr>
<td>acct</td>
<td>Switches process accounting on or off.</td>
</tr>
<tr>
<td>alarm</td>
<td>Sets an alarm clock for delivery of a signal.</td>
</tr>
<tr>
<td>brk, sbrk</td>
<td>Changes data segment size.</td>
</tr>
<tr>
<td>chdir</td>
<td>Changes working directory.</td>
</tr>
<tr>
<td>chmod, fchmod</td>
<td>Changes permissions of a file.</td>
</tr>
<tr>
<td>chown, fchown, lchown</td>
<td>Changes ownership of a file.</td>
</tr>
<tr>
<td>chroot</td>
<td>Changes root directory.</td>
</tr>
<tr>
<td>close</td>
<td>Closes a file descriptor.</td>
</tr>
<tr>
<td>create</td>
<td>Creates a file or device.</td>
</tr>
<tr>
<td>dup, dup2</td>
<td>Duplicates a file descriptor.</td>
</tr>
<tr>
<td>execve</td>
<td>Executes program.</td>
</tr>
<tr>
<td>exit</td>
<td>Causes normal program termination.</td>
</tr>
<tr>
<td>fchdir</td>
<td>Changes the working directory.</td>
</tr>
<tr>
<td>fcntl</td>
<td>Manipulates file descriptor.</td>
</tr>
<tr>
<td>fdatasync</td>
<td>Synchronizes a file's in-core data with that on disk.</td>
</tr>
<tr>
<td>API</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>flock</td>
<td>Applies or removes an advisory lock on an open file.</td>
</tr>
<tr>
<td>fork</td>
<td>Creates a child process.</td>
</tr>
<tr>
<td>fstat, stat, lstat</td>
<td>Gets file status.</td>
</tr>
<tr>
<td>fstatfs, statfs</td>
<td>Gets file system statistics.</td>
</tr>
<tr>
<td>fsync</td>
<td>Synchronizes a file complete in-core state with that on disk.</td>
</tr>
<tr>
<td>ftruncate, truncate</td>
<td>Truncates a file to a specified length.</td>
</tr>
<tr>
<td>getcwd, get_current_dir_name, getwd</td>
<td>Gets current working directory.</td>
</tr>
<tr>
<td>getegid</td>
<td>Gets group identity.</td>
</tr>
<tr>
<td>geteuid</td>
<td>Gets user identity.</td>
</tr>
<tr>
<td>getgid</td>
<td>Gets group identity.</td>
</tr>
<tr>
<td>getgroups, setgroups</td>
<td>Gets/sets list of supplementary group IDs.</td>
</tr>
<tr>
<td>getitimer, settimer</td>
<td>Gets or sets the value of an interval timer.</td>
</tr>
<tr>
<td>getpgid, setpgid, setgrp, getpgrp</td>
<td>Sets/gets process group.</td>
</tr>
<tr>
<td>getpid, getppid</td>
<td>Gets process identification.</td>
</tr>
<tr>
<td>getpriority, setpriority</td>
<td>Gets/sets program scheduling priority.</td>
</tr>
<tr>
<td>getrlimit, getrusage, setrlimit</td>
<td>Gets/sets resource limits and usage.</td>
</tr>
<tr>
<td>getsid</td>
<td>Gets session ID.</td>
</tr>
<tr>
<td>gettimeofday, settimeofday</td>
<td>Gets/sets time.</td>
</tr>
<tr>
<td>getuid</td>
<td>Gets user identity.</td>
</tr>
<tr>
<td>init_module</td>
<td>Initializes a loadable module entry.</td>
</tr>
<tr>
<td>kill</td>
<td>Sends signal to a process.</td>
</tr>
<tr>
<td>link</td>
<td>Makes a new name for a file.</td>
</tr>
<tr>
<td>lseek</td>
<td>Repositions read/write file offset.</td>
</tr>
<tr>
<td>mkdir</td>
<td>Creates a directory.</td>
</tr>
<tr>
<td>mknod</td>
<td>Creates a directory or special or ordinary file.</td>
</tr>
<tr>
<td>API</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>mmap, munmap</td>
<td>Maps or unmaps files or devices into memory.</td>
</tr>
<tr>
<td>mprotect</td>
<td>Controls allowable accesses to a region of memory.</td>
</tr>
<tr>
<td>msync</td>
<td>Synchronizes a file with a memory map.</td>
</tr>
<tr>
<td>nice</td>
<td>Changes process priority.</td>
</tr>
<tr>
<td>open</td>
<td>Opens a file or device.</td>
</tr>
<tr>
<td>pause</td>
<td>Waits for signal.</td>
</tr>
<tr>
<td>pipe</td>
<td>Creates pipe.</td>
</tr>
<tr>
<td>poll</td>
<td>Waits for some event on a file descriptor.</td>
</tr>
<tr>
<td>pread, pwrite</td>
<td>Reads from or writes to a file descriptor at a given offset.</td>
</tr>
<tr>
<td>read</td>
<td>Reads from a file descriptor.</td>
</tr>
<tr>
<td>readdir</td>
<td>Reads entry from directory handle.</td>
</tr>
<tr>
<td>readlink</td>
<td>Reads the value of a symbolic link.</td>
</tr>
<tr>
<td>rename</td>
<td>Changes the name or location of a file.</td>
</tr>
<tr>
<td>rmdir</td>
<td>Deletes a directory.</td>
</tr>
<tr>
<td>sched_setparam, sched_getparam</td>
<td>Sets and gets scheduling parameters.</td>
</tr>
<tr>
<td>sched_yield</td>
<td>Yields the processor.</td>
</tr>
<tr>
<td>select</td>
<td>Synchronous I/O multiplexing.</td>
</tr>
<tr>
<td>setdomainname, getdomainname</td>
<td>Gets/sets the domain name.</td>
</tr>
<tr>
<td>setgid</td>
<td>Sets group identity.</td>
</tr>
<tr>
<td>setregid, setegid</td>
<td>Sets real and/or effective group ID.</td>
</tr>
<tr>
<td>setreuid, seteuid</td>
<td>Sets real and/or effective user ID.</td>
</tr>
<tr>
<td>setsid</td>
<td>Creates a session and sets the process group ID.</td>
</tr>
<tr>
<td>setuid</td>
<td>Sets user identity.</td>
</tr>
<tr>
<td>API</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sgetmask</td>
<td>Signal handling function. Not supported. Superseded by sigprocmask(). No man pages available.</td>
</tr>
<tr>
<td>sigaction, sigprocmask, sigpending, sgsuspend</td>
<td>POSIX signal handling function.</td>
</tr>
<tr>
<td>sigaltstack</td>
<td>Defines and examines the state of an alternate stack for signal handlers.</td>
</tr>
<tr>
<td>signal</td>
<td>ANSI C signal handling function.</td>
</tr>
<tr>
<td>ssetmask</td>
<td>Signal handling function. Not supported. Superseded by sigprocmask(). No man pages available.</td>
</tr>
<tr>
<td>stime</td>
<td>Sets time.</td>
</tr>
<tr>
<td>swapon, swapoff</td>
<td>Starts/stops swapping to file/device.</td>
</tr>
<tr>
<td>symlink</td>
<td>Makes a new name for a file.</td>
</tr>
<tr>
<td>time</td>
<td>Gets time in seconds.</td>
</tr>
<tr>
<td>times</td>
<td>Gets process times.</td>
</tr>
<tr>
<td>umask</td>
<td>Sets file creation mask.</td>
</tr>
<tr>
<td>umount</td>
<td>umount a file system.</td>
</tr>
<tr>
<td>uname</td>
<td>Gets name and information about current kernel.</td>
</tr>
<tr>
<td>unlink</td>
<td>Deletes a name and possibly the file it refers to.</td>
</tr>
<tr>
<td>utoke, utimes</td>
<td>Changes access and/or modification times of an inode.</td>
</tr>
<tr>
<td>vfork</td>
<td>Creates a child process and block parent.</td>
</tr>
<tr>
<td>waitpid</td>
<td>Waits for process termination.</td>
</tr>
<tr>
<td>write</td>
<td>Writes to a file descriptor.</td>
</tr>
<tr>
<td>writev</td>
<td>readv, writev - read or write data into multiple buffers.</td>
</tr>
<tr>
<td>API</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>adjtimex</td>
<td>Tunes kernel clock.</td>
</tr>
<tr>
<td>bdflush</td>
<td>Starts, flushes, or tunes buffer-dirty-flush daemon.</td>
</tr>
<tr>
<td>capget, capset</td>
<td>Sets gets process capabilities.</td>
</tr>
<tr>
<td>clone</td>
<td>Creates a child process.</td>
</tr>
<tr>
<td>create_module</td>
<td>Creates a loadable module entry.</td>
</tr>
<tr>
<td>delete_module</td>
<td>Deletes a loadable module entry.</td>
</tr>
<tr>
<td>get_kernel_syms</td>
<td>Retrieves exported kernel and module symbols.</td>
</tr>
<tr>
<td>getresgid, setresgid</td>
<td>Sets or gets real, effective, and saved group ID.</td>
</tr>
<tr>
<td>getresuid, setresuid</td>
<td>Sets real, effective and saved user or group ID.</td>
</tr>
<tr>
<td>idle</td>
<td>Makes process 0 idle.</td>
</tr>
<tr>
<td>ioperm</td>
<td>Sets port input/output permissions.</td>
</tr>
<tr>
<td>iopl</td>
<td>Changes I/O privilege level.</td>
</tr>
<tr>
<td>ipc</td>
<td>System V IPC system calls.</td>
</tr>
<tr>
<td>llseek</td>
<td>Repositions read/write file offset.</td>
</tr>
<tr>
<td>mlock</td>
<td>Disables paging for some parts of memory.</td>
</tr>
<tr>
<td>mlockall</td>
<td>Disables paging for calling process.</td>
</tr>
<tr>
<td>modify_ldt</td>
<td>Gets or sets ldt.</td>
</tr>
<tr>
<td>mremap</td>
<td>Remaps a virtual memory address.</td>
</tr>
<tr>
<td>munlock</td>
<td>Reenables paging for some parts of memory.</td>
</tr>
<tr>
<td>munlockall</td>
<td>Reenables paging for calling process.</td>
</tr>
<tr>
<td>nfsservctld</td>
<td>Syscall interface to kernel nfs daemon.</td>
</tr>
<tr>
<td>personality</td>
<td>Sets the process execution domain.</td>
</tr>
<tr>
<td>prctl</td>
<td>Operations on a process.</td>
</tr>
<tr>
<td>API</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>query_module</td>
<td>Queries the kernel for various bits pertaining to modules.</td>
</tr>
<tr>
<td>sched_get_priority_max, sched_get_priority_min</td>
<td>Gets static priority.</td>
</tr>
<tr>
<td>sched_setscheduler, sched_getscheduler</td>
<td>Sets and gets scheduling.</td>
</tr>
<tr>
<td>sched_rr_get_interval</td>
<td>Gets the SCHED_RR interval for the named process.</td>
</tr>
<tr>
<td>sendfile</td>
<td>Transfers data between file descriptors.</td>
</tr>
<tr>
<td>setfsgid</td>
<td>Sets group identity used for file system checks.</td>
</tr>
<tr>
<td>setsuid</td>
<td>Sets user identity used for file system checks.</td>
</tr>
<tr>
<td>sigreturn</td>
<td>Returns from signal handler and cleanup stack frame.</td>
</tr>
<tr>
<td>socketcall</td>
<td>Socket system calls.</td>
</tr>
<tr>
<td>sysctl</td>
<td>Reads/writes system parameters.</td>
</tr>
<tr>
<td>sysinfo</td>
<td>Returns information on overall system statistics.</td>
</tr>
<tr>
<td>uselib</td>
<td>Selects shared library.</td>
</tr>
<tr>
<td>vhangup</td>
<td>Virtually hangup the current tty.</td>
</tr>
<tr>
<td>vm86old, vm86</td>
<td>Enter virtual 8086 mode.</td>
</tr>
</tbody>
</table>
### Table A-4  Linux-compatible APIs introduced in AIX 5L 5.1

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nanosleep</td>
<td>Pauses execution for a specified time.</td>
</tr>
<tr>
<td>ptrace</td>
<td>Processes trace.</td>
</tr>
<tr>
<td>quotactl</td>
<td>Manipulates disk quotas.</td>
</tr>
<tr>
<td>reboot</td>
<td>Reboots or enables/disables Ctrl-Alt-Del.</td>
</tr>
<tr>
<td>sysfs</td>
<td>Gets file system type information.</td>
</tr>
<tr>
<td>wait3, wait4</td>
<td>Wait for process termination, BSD style.</td>
</tr>
</tbody>
</table>

### Table A-5  Linux-compatible APIs available on AIX but not 100% compatible

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ioctl</td>
<td>Control device.</td>
</tr>
<tr>
<td>mount</td>
<td>Mounts file systems.</td>
</tr>
<tr>
<td>readv</td>
<td>Reads data into multiple buffers.</td>
</tr>
<tr>
<td>sync</td>
<td>Commits buffer cache to disk.</td>
</tr>
<tr>
<td>syslog</td>
<td>Writes messages to the system logger.</td>
</tr>
</tbody>
</table>
Linux Standard Base APIs

One of goals of the Linux Standard Base is to develop and promote standards that will increase compatibility among Linux distributions and enable software applications to run on any compliant Linux system. The next tables contain the rest of the library functions from the Linux Standard Base (LSB) not mentioned in the previous tables. Table A-6 describes the different groups of APIs regarding compatibility and implementation.

**Table A-6  Different groups of LSB APIs**

<table>
<thead>
<tr>
<th>Group of APIs</th>
<th>Listed at Table:</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSB APIs compatible with AIX</td>
<td>Table A-7</td>
</tr>
<tr>
<td>LSB APIs not available on AIX</td>
<td>Table A-8 on page 170</td>
</tr>
<tr>
<td>LSB APIs introduced in AIX 5L 5.1</td>
<td>Table A-9 on page 170</td>
</tr>
<tr>
<td>LSB APIs available on AIX but not 100% compatible</td>
<td>Table A-10 on page 170</td>
</tr>
</tbody>
</table>

**Table A-7  Compatible LSB APIs**

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abort</td>
<td>Causes abnormal program termination.</td>
</tr>
<tr>
<td>abs</td>
<td>Computes the absolute value of an integer.</td>
</tr>
<tr>
<td>accept</td>
<td>Accepts a connection on a socket.</td>
</tr>
<tr>
<td>asctime</td>
<td>Transforms binary date and time to ASCII.</td>
</tr>
<tr>
<td>atexit</td>
<td>Registers a function to be called at normal program termination.</td>
</tr>
<tr>
<td>atof</td>
<td>Converts a string to a double.</td>
</tr>
<tr>
<td>atoi</td>
<td>Converts a string to an integer.</td>
</tr>
<tr>
<td>atol</td>
<td>Converts a string to a long integer.</td>
</tr>
<tr>
<td>bcmp</td>
<td>Compares byte strings.</td>
</tr>
<tr>
<td>bcopy</td>
<td>Copies byte strings.</td>
</tr>
<tr>
<td>bind</td>
<td>Binds a name to a socket.</td>
</tr>
<tr>
<td>bsearch</td>
<td>Binary search of a sorted array.</td>
</tr>
<tr>
<td>bzero</td>
<td>Writes zeros to a byte string.</td>
</tr>
<tr>
<td>API</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>calloc</td>
<td>Allocates and frees dynamic memory.</td>
</tr>
<tr>
<td>catopen, catclose, catgets</td>
<td>Message catalog operations.</td>
</tr>
<tr>
<td>cfgetispeed</td>
<td>Gets terminal input speed.</td>
</tr>
<tr>
<td>cfgetospeed</td>
<td>Gets terminal output speed.</td>
</tr>
<tr>
<td>cfsetispeed</td>
<td>Sets terminal input speed.</td>
</tr>
<tr>
<td>cfsetospeed</td>
<td>Sets terminal output speed.</td>
</tr>
<tr>
<td>clearerr, feof, ferror, fileno</td>
<td>Checks and resets stream status.</td>
</tr>
<tr>
<td>clock</td>
<td>Determine processor time.</td>
</tr>
<tr>
<td>closedir</td>
<td>Closes a directory.</td>
</tr>
<tr>
<td>connect</td>
<td>Initiates a connection on a socket.</td>
</tr>
<tr>
<td>ctermid</td>
<td>Gets the controlling terminal name.</td>
</tr>
<tr>
<td>ctime</td>
<td>Transforms binary date and time to ASCII.</td>
</tr>
<tr>
<td>difftime</td>
<td>Calculates time difference.</td>
</tr>
<tr>
<td>div</td>
<td>Computes the quotient and remainder of integer division.</td>
</tr>
<tr>
<td>drand48</td>
<td>Generates uniformly distributed pseudo-random numbers.</td>
</tr>
<tr>
<td>ecvt</td>
<td>Converts a floating-point number to a string.</td>
</tr>
<tr>
<td>endgrent, setgrent, getgrent</td>
<td>Gets group file entry.</td>
</tr>
<tr>
<td>endpwent, getpwent, setpwent-getpwuid</td>
<td>Gets password file entry.</td>
</tr>
<tr>
<td>erand48</td>
<td>Generates uniformly distributed pseudo-random numbers.</td>
</tr>
<tr>
<td>execl</td>
<td>Executes a file.</td>
</tr>
<tr>
<td>execle</td>
<td>Executes a file.</td>
</tr>
<tr>
<td>execlp</td>
<td>Executes a file.</td>
</tr>
<tr>
<td>execv</td>
<td>Executes a file.</td>
</tr>
<tr>
<td>execvp</td>
<td>Executes a file.</td>
</tr>
<tr>
<td>API</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
</tr>
<tr>
<td>fclose</td>
<td>Closes a stream.</td>
</tr>
<tr>
<td>fcvt</td>
<td>Converts a floating-point number to a string.</td>
</tr>
<tr>
<td>fopen, fdopen, freopen</td>
<td>Streams open functions.</td>
</tr>
<tr>
<td>fflush</td>
<td>Flushes a stream.</td>
</tr>
<tr>
<td>ffs</td>
<td>Finds first bit set in a word.</td>
</tr>
<tr>
<td>fgetc, fgets, getc, getchar, gets, ungetc</td>
<td>Input of characters and strings.</td>
</tr>
<tr>
<td>fgetpos, fseek, fsetpos, ftell, rewind</td>
<td>Repositions a stream.</td>
</tr>
<tr>
<td>fprintf, printf, sprintf, snprintf, vprintf, vsnprintf</td>
<td>Formatted output conversion.</td>
</tr>
<tr>
<td>fputc, fputs, putc, printf, puts</td>
<td>Output of characters and strings.</td>
</tr>
<tr>
<td>fread, fwrite</td>
<td>Binary stream input/output.</td>
</tr>
<tr>
<td>free</td>
<td>Frees dynamic memory.</td>
</tr>
<tr>
<td>fscanf, scanf, sscanf</td>
<td>Inputs format conversion.</td>
</tr>
<tr>
<td>getdents</td>
<td>Gets directory entries.</td>
</tr>
<tr>
<td>getenv</td>
<td>Gets an environment variable.</td>
</tr>
<tr>
<td>gethostid, sethostid</td>
<td>Gets or sets the unique identifier of the current host.</td>
</tr>
<tr>
<td>gethostname</td>
<td>Gets host name.</td>
</tr>
<tr>
<td>getmsg</td>
<td>Gets the next message off a stream.</td>
</tr>
<tr>
<td>getpeername</td>
<td>Gets name of connected peer.</td>
</tr>
<tr>
<td>getsockname</td>
<td>Gets socket name.</td>
</tr>
<tr>
<td>getsockopt, setsockopt</td>
<td>Gets and sets options on sockets.</td>
</tr>
<tr>
<td>getw, putw</td>
<td>Input and output of words (ints).</td>
</tr>
<tr>
<td>gmtime</td>
<td>Transforms binary date and time to ASCII.</td>
</tr>
<tr>
<td>index, rindex</td>
<td>Locates character in string.</td>
</tr>
<tr>
<td>initstate</td>
<td>Random number generator.</td>
</tr>
</tbody>
</table>
### APIs

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isalnum, isalpha, isascii, isblank, iscntrl, isdigit, isgraph</td>
<td>Character classification routines.</td>
</tr>
<tr>
<td>islower, isprint, ispunct, isspace, isupper, isxdigit</td>
<td>Character classification routines.</td>
</tr>
<tr>
<td>iswalnum</td>
<td>Tests for alphanumeric wide character.</td>
</tr>
<tr>
<td>iswalpha</td>
<td>Tests for alphanumeric wide character.</td>
</tr>
<tr>
<td>jrand48</td>
<td>Generates uniformly distributed pseudo-random numbers.</td>
</tr>
<tr>
<td>killpg</td>
<td>Sends signal to a process group.</td>
</tr>
<tr>
<td>labs</td>
<td>Computes the absolute value of a long integer.</td>
</tr>
<tr>
<td>ldiv</td>
<td>Computes the quotient and remainder of long integer division.</td>
</tr>
<tr>
<td>listen</td>
<td>Listens for connections on a socket.</td>
</tr>
<tr>
<td>localtime</td>
<td>Transforms binary date and time to ASCII.</td>
</tr>
<tr>
<td>lrand48</td>
<td>Generates uniformly distributed pseudo-random numbers.</td>
</tr>
<tr>
<td>malloc</td>
<td>Allocates dynamic memory.</td>
</tr>
<tr>
<td>memccpy, memchr, memcmp, memcmp, memcpy, memset</td>
<td>Memory operations.</td>
</tr>
<tr>
<td>mkfifo</td>
<td>Makes a FIFO special file (a named pipe).</td>
</tr>
<tr>
<td>mkstemp</td>
<td>Creates a unique temporary file.</td>
</tr>
<tr>
<td>mktemp</td>
<td>Makes a unique temporary file name.</td>
</tr>
<tr>
<td>mktime</td>
<td>Transforms binary date and time to ASCII.</td>
</tr>
<tr>
<td>mrand48</td>
<td>Generates pseudo-random number.</td>
</tr>
<tr>
<td>msgctl, msgget, msgsnd, msgrecv</td>
<td>Message operations.</td>
</tr>
<tr>
<td>mrand48</td>
<td>Generates uniformly distributed pseudo-random numbers.</td>
</tr>
<tr>
<td>opendir</td>
<td>Opens a directory.</td>
</tr>
</tbody>
</table>

*Appendix A. APIs* 165
<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pathconf</td>
<td>Retrieves file implementation characteristics.</td>
</tr>
<tr>
<td>pause</td>
<td>Suspends a process until a signal is received.</td>
</tr>
<tr>
<td>putenv</td>
<td>Sets an environment variable.</td>
</tr>
<tr>
<td>putmsg</td>
<td>Sends a message on a stream.</td>
</tr>
<tr>
<td>qsort</td>
<td>Sorts a table of data in place.</td>
</tr>
<tr>
<td>raise</td>
<td>Sends a signal to the current process.</td>
</tr>
<tr>
<td>rand</td>
<td>Generates pseudo-random number.</td>
</tr>
<tr>
<td>random</td>
<td>Generates pseudo-random number more efficiently.</td>
</tr>
<tr>
<td>re_comp</td>
<td>Regular expression handler.</td>
</tr>
<tr>
<td>reCompileFastmap</td>
<td>Service function for the Linux implementation of re_comp and re_exec.</td>
</tr>
<tr>
<td>reCompilePattern</td>
<td>Service function for the Linux implementation of re_comp and re_exec.</td>
</tr>
<tr>
<td>re_exec</td>
<td>Regular expression handler.</td>
</tr>
<tr>
<td>re_match</td>
<td>Returns number of characters that matched string.</td>
</tr>
<tr>
<td>re_search</td>
<td>Searches string for pattern.</td>
</tr>
<tr>
<td>re_search_2</td>
<td>Searches string for pattern.</td>
</tr>
<tr>
<td>re_set_registers</td>
<td>Undocumented library call.</td>
</tr>
<tr>
<td>re_set_syntax</td>
<td>Sets the current default syntax.</td>
</tr>
<tr>
<td>realloc</td>
<td>Memory allocator.</td>
</tr>
<tr>
<td>recv, recvfrom, recvmsg</td>
<td>Receives a message from a socket.</td>
</tr>
<tr>
<td>regcomp</td>
<td>Compiles a regular expression into an executable string.</td>
</tr>
<tr>
<td>regerror</td>
<td>Returns string that describes ErrCode parameter.</td>
</tr>
<tr>
<td>regexec</td>
<td>POSIX regex function.</td>
</tr>
<tr>
<td>API</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>regfree</td>
<td>Frees memory allocated by regcomp().</td>
</tr>
<tr>
<td>remove</td>
<td>Deletes a name and possibly the file it refers to.</td>
</tr>
<tr>
<td>res_init</td>
<td>Searches for default domain name and Internet address.</td>
</tr>
<tr>
<td>rewinddir</td>
<td>Resets directory stream.</td>
</tr>
<tr>
<td>reexec</td>
<td>Command execution on remote host.</td>
</tr>
<tr>
<td>sbrk</td>
<td>Changes data segment size.</td>
</tr>
<tr>
<td>seed48</td>
<td>Generates uniformly distributed pseudo-random number sequences.</td>
</tr>
<tr>
<td>seekdir</td>
<td>Sets the position of the next readdir() call in the directory stream.</td>
</tr>
<tr>
<td>semctl, semget, semop</td>
<td>Semaphore operations.</td>
</tr>
<tr>
<td>send, sendto, sendmsg</td>
<td>Sends a message from a socket.</td>
</tr>
<tr>
<td>setbuf, setvbuf</td>
<td>Streams buffering operations.</td>
</tr>
<tr>
<td>setegid</td>
<td>Sets real and/or effective group ID.</td>
</tr>
<tr>
<td>seteuid</td>
<td>Sets real and/or effective group ID.</td>
</tr>
<tr>
<td>sethostname</td>
<td>Sets host name.</td>
</tr>
<tr>
<td>setlocale</td>
<td>Sets the current locale.</td>
</tr>
<tr>
<td>setstate</td>
<td>Generates pseudo-random numbers more efficiently.</td>
</tr>
<tr>
<td>shmat, shmctl, shmdt, shmget</td>
<td>Shared memory operations.</td>
</tr>
<tr>
<td>shutdown</td>
<td>Shuts down part of a full-duplex connection.</td>
</tr>
<tr>
<td>sigaction, sigprocmask, sigpending, sigsuspend</td>
<td>POSIX signal set operations.</td>
</tr>
<tr>
<td>sigaddset, sigemptyset, sigfillset, sigdelset, sigismember</td>
<td>POSIX signal set operations.</td>
</tr>
<tr>
<td>sigalstack</td>
<td>Defines and examines the state of an alternate stack for signal handlers.</td>
</tr>
<tr>
<td>sigsetmask, sigmask</td>
<td>Manipulates the signal mask.</td>
</tr>
<tr>
<td>API</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>sigset, sighold, sigrelse, sigignore</td>
<td>Enhanced signal management.</td>
</tr>
<tr>
<td>siginterrupt</td>
<td>Allows signals to interrupt system calls.</td>
</tr>
<tr>
<td>siglongjmp</td>
<td>Non-local jump to a saved stack context.</td>
</tr>
<tr>
<td>sigstack</td>
<td>Sets and gets signal stack context.</td>
</tr>
<tr>
<td>sigvec</td>
<td>BSD software signal facilities.</td>
</tr>
<tr>
<td>sigwait</td>
<td>Handling of signals in threads.</td>
</tr>
<tr>
<td>sleep</td>
<td>Puts process to sleep.</td>
</tr>
<tr>
<td>socket</td>
<td>Creates an endpoint for communication.</td>
</tr>
<tr>
<td>socketpair</td>
<td>Creates a pair of connected sockets.</td>
</tr>
<tr>
<td>srand</td>
<td>Generates pseudo-random numbers.</td>
</tr>
<tr>
<td>srand48</td>
<td>Generates uniformly distributed pseudo-random number sequences.</td>
</tr>
<tr>
<td>srandrandom</td>
<td>Generates pseudo-random numbers more efficiently.</td>
</tr>
<tr>
<td>statvfs</td>
<td>Returns information about a file system.</td>
</tr>
<tr>
<td>strcmp, strncmp, strcoll</td>
<td>Compares strings.</td>
</tr>
<tr>
<td>strcat, strncat, strxfrm, strcpyp, strncpy, strdup</td>
<td>Copies and appends strings.</td>
</tr>
<tr>
<td>strlen, strchr, strrchr, strpbrk, strspn, strcspn, strstr, strtok</td>
<td>Determines the size, location, and existence of strings.</td>
</tr>
<tr>
<td>strerror</td>
<td>Returns string describing error code.</td>
</tr>
<tr>
<td>strftime</td>
<td>Formats monetary strings.</td>
</tr>
<tr>
<td>strftime</td>
<td>Formats date and time.</td>
</tr>
<tr>
<td>strpbrk</td>
<td>Searches a string for any of a set of characters.</td>
</tr>
<tr>
<td>strptime</td>
<td>Converts a string representation of time to a time tm structure.</td>
</tr>
<tr>
<td>strtod</td>
<td>Converts string to double.</td>
</tr>
<tr>
<td>strtof</td>
<td>Converts string to float.</td>
</tr>
<tr>
<td>API</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>strtol</td>
<td>Converts string to long.</td>
</tr>
<tr>
<td>strtol(), strtol()</td>
<td>Converts string to long double or long long.</td>
</tr>
<tr>
<td>strtoul</td>
<td>Converts string to unsigned long.</td>
</tr>
<tr>
<td>swab</td>
<td>Swaps adjacent bytes.</td>
</tr>
<tr>
<td>tcdrain</td>
<td>Waits for output to complete.</td>
</tr>
<tr>
<td>tcflow</td>
<td>Performs flow control functions.</td>
</tr>
<tr>
<td>tcflush</td>
<td>Discards data from the specified queue.</td>
</tr>
<tr>
<td>tcgetattr</td>
<td>Gets terminal state.</td>
</tr>
<tr>
<td>tcgetpgrp</td>
<td>Gets foreground process group ID.</td>
</tr>
<tr>
<td>tcgetsid</td>
<td>Gets foreground session ID</td>
</tr>
<tr>
<td>tcsendbreak</td>
<td>Sends a break on an asynchronous serial line.</td>
</tr>
<tr>
<td>tcsetattr</td>
<td>Sets terminal state.</td>
</tr>
<tr>
<td>tcsetpgrp</td>
<td>Sets foreground process group ID.</td>
</tr>
<tr>
<td>telldir</td>
<td>Returns current location in directory stream.</td>
</tr>
<tr>
<td>tempnam</td>
<td>Creates a name for a temporary file.</td>
</tr>
<tr>
<td>timezone</td>
<td>Global.</td>
</tr>
<tr>
<td>tolower</td>
<td>Converts letter to lower case.</td>
</tr>
<tr>
<td>toupper</td>
<td>Converts letter to upper case.</td>
</tr>
<tr>
<td>tname</td>
<td>Global</td>
</tr>
<tr>
<td>tzset</td>
<td>Converts the formats of date and time representations.</td>
</tr>
<tr>
<td>ulimit</td>
<td>Sets and gets user limits.</td>
</tr>
<tr>
<td>wait</td>
<td>Wait for child process to stop or terminate.</td>
</tr>
<tr>
<td>wait3</td>
<td>Wait for child, BSD style.</td>
</tr>
<tr>
<td>waitid</td>
<td>Wait for child matching idtype and ID.</td>
</tr>
<tr>
<td>API</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>unknown</td>
<td>Handles attempts to use non-existent commands.</td>
</tr>
</tbody>
</table>

Table A-8  LSB APIs not available

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sigqueue</td>
<td>POSIX signal handle function.</td>
</tr>
<tr>
<td>sigtimedwait</td>
<td>Waits for queue signals.</td>
</tr>
<tr>
<td>sigwaitinfo</td>
<td>Waits for queue signals.</td>
</tr>
<tr>
<td>strfry</td>
<td>Randomizes a string.</td>
</tr>
</tbody>
</table>

Table A-9  LSB APIs introduced in AIX 5L 5.1

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfsetspeed</td>
<td>Sets terminal input and output speed.</td>
</tr>
<tr>
<td>Initgroups</td>
<td>Initializes the supplementary group access list.</td>
</tr>
<tr>
<td>iswblank</td>
<td>Tests for alphanumeric wide character.</td>
</tr>
<tr>
<td>sigblock</td>
<td>Manipulates the signal mask.</td>
</tr>
<tr>
<td>siggetmask</td>
<td>Manipulates the signal mask.</td>
</tr>
<tr>
<td>strerror_r</td>
<td>Returns string describing error code.</td>
</tr>
<tr>
<td>strnlen</td>
<td>Determines the length of a fixed-size string.</td>
</tr>
<tr>
<td>strsep</td>
<td>Extracts token from string.</td>
</tr>
<tr>
<td>strsignal</td>
<td>Returns string describing signal.</td>
</tr>
<tr>
<td>strtok_r</td>
<td>Determines the size, location, and existence of strings.</td>
</tr>
<tr>
<td>sysconf</td>
<td>Determines current value of system limit or option.</td>
</tr>
</tbody>
</table>

Table A-10  LSB APIs available on AIX but not 100% compatible

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>insque, remque</td>
<td>Inserts/removes an item from a queue.</td>
</tr>
<tr>
<td>strcasecmp</td>
<td>Compares strings.</td>
</tr>
</tbody>
</table>
New APIs in AIX 5L 5.1

In this section, we introduce the new APIs that will be available on AIX 5L 5.1. These new APIs continue to make AIX more Linux compatible and increase the level of functionality, providing the developer more flexibility when programming or porting an application for deployment on AIX, or when writing code on AIX to be deployed on Linux-based system.

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strncasecmp</td>
<td>Compares strings.</td>
</tr>
</tbody>
</table>
Name  cfsetspeed: Terminal input and output speed

Linux synopsis  
#include <termios.h>
int cfsetspeed(struct termios *termios_p, speed_t speed);

AIX synopsis  None
**Name**
initgroups: Initializes the supplementary group access list.

**Linux synopsis**
```c
#include <grp.h>
#include <sys/types.h>
int initgroups(const char *user, gid_t group);
```

**AIX synopsis**
```
int initgroups (char *User, int BaseGID);
```

**Details**
The `gid_t` may cause compiler warnings or errors.

**Comment**
The prototype `int initgroups (const char *User, gid_t ggroup)` will be added to `grp.h`, and the `initgroups` function will be redefined to change the first arg to a `const`. The documentation page for `initgroups` will be brought up to date. This will not break compatibility in AIX, because either a `char *` or `const char *` can be an actual parameter in a `const char *` formal parameter.
Name
iswalnum: Tests for alphanumeric wide character.

Linux synopsis
#include <wctype.h>
int iswalnum(wint_t wc);
int iswalpha(wint_t wc);
int iswcntrl(wint_t wc);
int iswdigit(wint_t wc);
int iswgraph(wint_t wc);
int iswlower(wint_t wc);
int iswprint(wint_t wc);
int iswpunct(wint_t wc);
int iswspace(wint_t wc);
int iswupper(wint_t wc);
int iswxdigit(wint_t wc);
int iswblank(wint_t wc);

AIX synopsis
#include <wchar.h>
int iswalnum(wint_t WC);
int iswalpha (wint_t WC);
int iswcntrl (wint_t WC);
int iswdigit (wint_t WC);
int iswgraph (wint_t WC);
int iswlower (wint_t WC);
int iswprint (wint_t WC);
int iswpunct (wint_t WC);
int iswspace (wint_t WC);
int iswupper (wint_t WC);
int iswxdigit (wint_t WC);

Details
AIX does not define iswblank().
<table>
<thead>
<tr>
<th>Name</th>
<th>nanosleep: Pauses execution for a specified time.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Linux synopsis</strong></td>
<td><code>#include &lt;time.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>int nanosleep(const struct timespec *req, struct timespec *rem);</code></td>
</tr>
<tr>
<td><strong>AIX synopsis</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Comment</strong></td>
<td>Granularity is allowed to be as large as HZ.</td>
</tr>
</tbody>
</table>
### Name
ptrace: Process trace.

### Linux synopsis
```c
#include <sys/ptrace.h>

long int ptrace(enum ptrace_request request, pid_t pid, void *addr, void *data);
```

### AIX synopsis
```c
#include <sys/reg.h>
#include <sys/ptrace.h>
#include <sys/ldr.h>

int ptrace(int Request, int Identifier, int *Address, int *Data, int *Buffer);
#define _LINUX_SOURCE_COMPAT
#include <sys/ptrace.h>
long int ptrace(enum ptrace_request request, pid_t pid, void *addr, void *data);
```

### Details
Functions are not compatible.

### Comment
In AIX 5L, if `sys/ptrace.h` is compiled with 
`_LINUX_SOURCE_COMPAT`, the application will see 
Linux semantics. The Linux ptrace wrapper will call the 
underlying AIX ptrace function. Linux request values will 
be defined, but will return ENOTSUP for functions not 
supported by AIX ptrace.

### Errno
When called in AIX, this function can return these errnos, 
which are not documented in Linux:

- **ENOTSUP**: The request is not supported.
- **EINVAL**: The debugger and the traced process are the 
same, or the Identifier parameter does not identify the 
thread that caused the exception.
**Name**

quotactl: Manipulates disk quotas.

**Linux synopsis**

```c
#include <sys/types.h>
#include <sys/quota.h>
int quotactl(int cmd, const char *special, int id, caddr_t addr);
```

**AIX synopsis**

```c
#include <jfs/quota.h>
int quotactl (char *Path, int Cmd, int ID, char *Addr);
```

**Details**

Functions are not compatible. The header is in a different location. The arguments are in a different order. The Q_SETQLIM subcommand is not defined in AIX. In AIX, quotactl() is a system call.

**Errno**

When called in AIX, this function can return these errnos, which are not documented in Linux:

- **ELOOP**: Too many symbolic links were encountered in translating a path name.
- **ENAMETOOLONG**: A component of either path name exceeded 255 characters, or the entire length of either path name exceeded 1023 characters.
- **ENOENT**: A file name does not exist.
- **ENOTDIR**: A component of a path prefix is not a directory.
- **EOPNOTSUPP**: The file system does not support quotas.
- **EROFS**: In Q_QUOTAON, the quota file resides on a read-only file system.
- **EUSERS**: The in-core quota table cannot be expanded.
**Name**
reboot: Reboots or enables/disables Ctrl-Alt-Del.

**Linux synopsis**
```
#include <unistd.h>
#include <sys/reboot.h>
int reboot (int flag);
```

**AIX synopsis**
```
#include <sys/reboot.h>
void reboot (int HowTo, void *Argument)
#define LINUX_SOURCE_COMPAT
#include <sys/reboot.h>
int reboot (int flag);
```

**Details**
Functions are not compatible. In AIX, reboot is a system call. According to the Linux man page, reboot is Linux specific, and should not be used in programs intended to be portable.

**Comment**
Add the Linux reboot command definitions to sys/reboot.h. Flag values will be mapped as follows (Linux -> AIX):

- `LINUX_REBOOT_CMD_RESTART` -> `RB_SOFTIPL`
- `LINUX_REBOOT_CMD_HALT` -> `RB_HALT_POWERED`
- `LINUX_REBOOT_CMD_POWER_OFF` -> `RB_HALT`
- `LINUX_REBOOT_CMD_RESTART2` -> `RB_POWIPL`
- `LINUX_REBOOT_CMD_CAD_ON` -> return(ENOSYS)
- `LINUX_REBOOT_CMD_CAD_OFF` -> return(0)

AIX will not implement CAD (Ctrl-Alt-Del) for Linux compatibility.

**Errno**
When called in AIX, this function can return these errnos, which are not documented in Linux:

- `ENOSYS`: Function not supported
- (`LINUX_REBOOT_CMD_CAD_ON` only)
Name  
sigblock, siggetmask, sigsetmask, sigmask: Manipulates the signal mask.

Linux synopsis  
#include <signal.h>
int sigblock(int mask);
int siggetmask(void);
int sigsetmask(int mask);
int sigmask(int signum);

AIX synopsis  
#include <signal.h>
int sigblock(int mask);
int sigsetmask(int mask);
int sigmask(int signum);

Details  
The siggetmask() function does not exist in AIX. The sigmask() function is not listed in LSB. There is no AIX man page for sigmask(). These functions should not be used by applications anyway because of the usage of ints instead of sigset_ts.

A man page entry for sigmask() needs to be created.
The siggetmask function will be available on AIX 5L 5.1.
Name
strlen, strchr, strrchr, strpbrk, strspn, strcspn, strstr, strtok, and strtok_r: Determines the size, location, and existence of strings.

Linux synopsis
#include <string.h>
size_t strlen(const char *s);
char *strchr(const char *s, int c);
char *strrchr(const char *s, int c);
char *strpbrk(const char *s, const char *accept);
size_t strspn(const char *s, const char *accept);
size_t strcspn(const char *s, const char *reject);
char *strstr(const char *haystack, const char *needle);
char *strtok(char *s, const char *delim);
#if defined __USE_POSIX || defined __USE_MISC
char *strtok_r (char * restrict s, const char * restrict delim, char ** restrict save_ptr);
#endif

AIX synopsis
#include <string.h>
size_t strlen(const char *String);
char *strchr(const char *String, int Character);
char *strrchr(const char *String, int Character);
char *strpbrk(const char *String1, const char *String2);
size_t strspn(const char *String1, const char *String2);
size_t strcspn(const char *String1, const char *String2);
char *strstr(const char *String1, const char *String2);
char *strtok(char *String1, const char *String2);
char *strtok_r(char *String1, const char *String2, char **SavePtr);

Details
Functions are source compatible. AIX has no documentation for strtok_r(), but it is exported from libc and prototyped in string.h.

Errno
When called in AIX, this function can return these errnos, which are not documented in Linux:

EFAULT: A string parameter is an invalid address.
Name  strerror and strerror_r: Returns a string describing error code.

Linux synopsis  
#include <string.h>
char *strerror(int errnum);
#ifdef __USE_MISC
char *strerror_r(int errnum, char *buf, size_t buflen);
#endif

AIX synopsis  
#include <string.h>
char *strerror(int ErrorNumber);
#include <pthread.h>
#include <string.h>
int strerror_r(int ErrorNumber, char *Buffer, int BuffLen);

Details  The strerror() functions are source compatible. The prototype for strerror_r() is different.

Comment  In AIX 5L 5.1, a compatible version of this function will be added to libc. The prototype will be visible without any special definitions. The prototype, when compiled with -D_LINUX_SOURCE_COMPAT, will be:

#include <string.h>
char *strerror_r(int ErrorNumber, char *Buffer, size_t BuffLen);
<table>
<thead>
<tr>
<th>Name</th>
<th>strnlen: Determines the length of a fixed-size string.</th>
</tr>
</thead>
</table>
| Linux synopsis| `#include <string.h>`
                 | `size_t strnlen (const char *s, size_t maxlen);` |
| AIX synopsis  | None                                                 |
Name  strsep: Extracts token from string.

Linux synopsis  
```
#include <string.h>
char *strsep(char **stringp, const char *delim);
```

AIX synopsis  
```
char *strsep(char **stringp, const char *delim);
```
<table>
<thead>
<tr>
<th>Name</th>
<th>strsignal: Returns a string describing a signal.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Linux synopsis</strong></td>
<td>#define _GNU_SOURCE</td>
</tr>
<tr>
<td></td>
<td>#include &lt;string.h&gt;</td>
</tr>
<tr>
<td></td>
<td>char *strsignal(int sig);</td>
</tr>
<tr>
<td></td>
<td>extern const char * const sys_siglist[];</td>
</tr>
<tr>
<td><strong>AIX synopsis</strong></td>
<td>None</td>
</tr>
<tr>
<td>Name</td>
<td>sysconf: Determines the current value of system limit or option.</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| **Linux synopsis** | `#include <unistd.h>`  
`int sysconf(int name);` |
| **AIX synopsis** | `#include <unistd.h>`  
`long sysconf(int name);` |
| **Details** | We need to identify all of the valid values for the `_SC` defines for this call. |
| **Comment** | AIX needs to identify `_SC_PHYS_PAGES` and `_SC_AVPHYS_PAGES`. All others defined on Linux are defined on AIX. These new flags will be available on AIX 5L 5.1. |
**Name**
wait3 and wait4: Waits for process termination, BSD style

**Linux synopsis**
```c
#define _USE_BSD
#include <sys/types.h>
#include <sys/resource.h>
#include <sys/wait.h>

pid_t wait3(int *status, int options, struct rusage *rusage);

pid_t wait4(pid_t pid, int *status, int options, struct rusage *rusage);
```

**AIX synopsis**
```c
#include <sys/types.h>
#include <sys/resource.h>
#include <sys/wait.h>

pid_t wait3(int *StatusLocation, int Options, struct rusage *ResourceUsage);
```

**Details**
AIX 4 does not implement wait4().

**Comment**
The wait4 call is available in AIX 5L 5.1.

**Errno**
When called in AIX, this function can return these errnos, which are not documented in Linux:

EINVAL: This subroutine was terminated by receipt of a signal.

EFAULT: The StatusLocation or ResourceUsage parameter points to a location outside of the address space of the process.
Differences in commands

This appendix provides us with a quick reference of the differences between Linux and AIX commands.

The AIX Toolbox for Linux Application CD contains a complete and updated listing of all the commands in terms of syntax differences and attributes. Also, these commands are documented at the following Web site:


Table B-1 provides a listing of commands in AIX and Linux that have syntax and attribute differences. Not all of these commands are included in the AIX Toolbox applications. The Linux commands that are included in your system would depend on the Open Source Software you installed.

Table B-1 Commands with syntax differences

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ac</td>
<td>Prints connect-time records.</td>
</tr>
<tr>
<td>apropos</td>
<td>Locates commands by keyword lookup.</td>
</tr>
<tr>
<td>at</td>
<td>Runs commands at a later time.</td>
</tr>
<tr>
<td>atq</td>
<td>Displays the queue of jobs waiting to be run.</td>
</tr>
<tr>
<td>atrm</td>
<td>Removes jobs spooled by the at command.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>awk</td>
<td>Finds lines in files matching patterns and then performs specified actions on them.</td>
</tr>
<tr>
<td>banner</td>
<td>Writes ASCII character strings in large letters to standard output.</td>
</tr>
<tr>
<td>batch</td>
<td>Runs jobs when the system load level permits it.</td>
</tr>
<tr>
<td>bc</td>
<td>Provides an interpreter for arbitrary-precision arithmetic language.</td>
</tr>
<tr>
<td>bsh</td>
<td>Invokes the Bourne shell.</td>
</tr>
<tr>
<td>cal</td>
<td>Displays a calendar.</td>
</tr>
<tr>
<td>cat</td>
<td>Concatenates or displays files.</td>
</tr>
<tr>
<td>chroot</td>
<td>Changes the root directory of a command.</td>
</tr>
<tr>
<td>cksum</td>
<td>Displays the checksum and byte count of a file.</td>
</tr>
<tr>
<td>cmp</td>
<td>Compares two files.</td>
</tr>
<tr>
<td>compress</td>
<td>Compresses data.</td>
</tr>
<tr>
<td>cp</td>
<td>Copies files.</td>
</tr>
<tr>
<td>cpio</td>
<td>Copies files into and out of archive storage and directories.</td>
</tr>
<tr>
<td>crontab</td>
<td>Submits, edits, lists, or removes cron jobs.</td>
</tr>
<tr>
<td>csplit</td>
<td>Splits files by context.</td>
</tr>
<tr>
<td>ctags</td>
<td>Makes a file of tags to help locate objects in source files.</td>
</tr>
<tr>
<td>cut</td>
<td>Writes out selected bytes, characters, or fields from each line of a file.</td>
</tr>
<tr>
<td>date</td>
<td>Displays or sets the date or time.</td>
</tr>
<tr>
<td>dd</td>
<td>Converts and copies a file.</td>
</tr>
<tr>
<td>diff</td>
<td>Compares text files.</td>
</tr>
<tr>
<td>diff3</td>
<td>Compares three files.</td>
</tr>
<tr>
<td>du</td>
<td>Summarizes disk usage.</td>
</tr>
<tr>
<td>echo</td>
<td>Writes character strings to standard output.</td>
</tr>
<tr>
<td>ed</td>
<td>Edits text by line.</td>
</tr>
<tr>
<td>egrep, fgrep, and grep</td>
<td>Searches a file for a pattern.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>env</td>
<td>Displays the current environment or sets the environment for the execution of a command.</td>
</tr>
<tr>
<td>expand</td>
<td>Writes to a standard output with tabs changed to spaces.</td>
</tr>
<tr>
<td>file</td>
<td>Determines file type.</td>
</tr>
<tr>
<td>find</td>
<td>Finds files with a matching expression.</td>
</tr>
<tr>
<td>fortune</td>
<td>Displays a random fortune from a database of fortunes.</td>
</tr>
<tr>
<td>getopt</td>
<td>Parses command line flags and parameters.</td>
</tr>
<tr>
<td>gprof</td>
<td>Displays call graph profile data.</td>
</tr>
<tr>
<td>halt</td>
<td>Stops the processor.</td>
</tr>
<tr>
<td>head</td>
<td>Displays the first few lines or bytes of a files or files.</td>
</tr>
<tr>
<td>help</td>
<td>Provides information for new users.</td>
</tr>
<tr>
<td>indent</td>
<td>Reformats a C language program.</td>
</tr>
<tr>
<td>init</td>
<td>Initializes and controls processes.</td>
</tr>
<tr>
<td>install</td>
<td>Installs a command.</td>
</tr>
<tr>
<td>ipcs</td>
<td>Reports interprocess communication facility status.</td>
</tr>
<tr>
<td>jobs</td>
<td>Displays the status of jobs in a current session.</td>
</tr>
<tr>
<td>join</td>
<td>Joins the data fields of two files.</td>
</tr>
<tr>
<td>kill</td>
<td>Sends a signal to running processes.</td>
</tr>
<tr>
<td>killall</td>
<td>Cancels all processes except the calling process.</td>
</tr>
<tr>
<td>ksh</td>
<td>Invokes the Korn shell.</td>
</tr>
<tr>
<td>last</td>
<td>Displays information about previous logins.</td>
</tr>
<tr>
<td>lastcomm</td>
<td>Displays information about the last commands executed.</td>
</tr>
<tr>
<td>lex</td>
<td>Generates a C language program that matches patterns for a simple lexical analysis of an input stream.</td>
</tr>
<tr>
<td>ln</td>
<td>Links files.</td>
</tr>
<tr>
<td>logger</td>
<td>Make entries in the system log.</td>
</tr>
<tr>
<td>look</td>
<td>Finds lines in a sorted file.</td>
</tr>
<tr>
<td>ls</td>
<td>Displays the contents of a directory.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>make</td>
<td>Maintains up-to-date versions of programs.</td>
</tr>
<tr>
<td>man</td>
<td>Displays manual entries online.</td>
</tr>
<tr>
<td>mknod</td>
<td>Creates a special file.</td>
</tr>
<tr>
<td>mkfifo</td>
<td>Makes a first-in-first-out (FIFO) special file.</td>
</tr>
<tr>
<td>more</td>
<td>Displays continuous text, one screen at a time, on a display screen.</td>
</tr>
<tr>
<td>mt</td>
<td>Gives subcommands to a streaming tape device.</td>
</tr>
<tr>
<td>mv</td>
<td>Moves files.</td>
</tr>
<tr>
<td>nice</td>
<td>Runs a command at a lower or higher priority.</td>
</tr>
<tr>
<td>nohup</td>
<td>Runs the command without hangups.</td>
</tr>
<tr>
<td>patch</td>
<td>Applies changes to files.</td>
</tr>
<tr>
<td>patchchk</td>
<td>Checks pathnames.</td>
</tr>
<tr>
<td>pr</td>
<td>Writes a file to a standard output.</td>
</tr>
<tr>
<td>printf</td>
<td>Writes formatted output.</td>
</tr>
<tr>
<td>ps</td>
<td>Shows the current status of the processes.</td>
</tr>
<tr>
<td>rdist</td>
<td>Maintains identical copies of files on multiple hosts.</td>
</tr>
<tr>
<td>reboot</td>
<td>Restarts the system.</td>
</tr>
<tr>
<td>red</td>
<td>Edits text by line.</td>
</tr>
<tr>
<td>remove</td>
<td>Delete files from /var/adm/acct sub-directories.</td>
</tr>
<tr>
<td>restore</td>
<td>Copies previously backed-up file systems or files. Created by the backup command from a local device.</td>
</tr>
<tr>
<td>rm</td>
<td>Removes (unlinks) files or directories.</td>
</tr>
<tr>
<td>rmt</td>
<td>Allows remote access to magnetic tape devices.</td>
</tr>
<tr>
<td>rsh</td>
<td>Executes the specified command at the remote host or logs into the remote host.</td>
</tr>
<tr>
<td>sa</td>
<td>Summarizes accounting records.</td>
</tr>
<tr>
<td>sdiff</td>
<td>Compares two files and displays the differences in a side-by-side format.</td>
</tr>
<tr>
<td>sed</td>
<td>Provides a stream editor.</td>
</tr>
</tbody>
</table>
## Differences in commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sh</td>
<td>Invokes the default shell.</td>
</tr>
<tr>
<td>shutdown</td>
<td>Ends system operation.</td>
</tr>
<tr>
<td>sleep</td>
<td>Suspend execution for an interval.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort files, merges files that are already sorted, and checks files to determine if they have been sorted.</td>
</tr>
<tr>
<td>split</td>
<td>Splits a file into pieces.</td>
</tr>
<tr>
<td>strings</td>
<td>Finds the printable strings in an object or binary file.</td>
</tr>
<tr>
<td>sum</td>
<td>Displays the checksum and block count of a file.</td>
</tr>
<tr>
<td>tail</td>
<td>Writes a file to standard output, beginning at a specified point.</td>
</tr>
<tr>
<td>tar</td>
<td>Manipulates archives.</td>
</tr>
<tr>
<td>tee</td>
<td>Display the output of a program and copies it into a file.</td>
</tr>
<tr>
<td>telinit</td>
<td>Initializes and controls processes.</td>
</tr>
<tr>
<td>test</td>
<td>Evaluates conditional expressions.</td>
</tr>
<tr>
<td>time</td>
<td>Prints the time for when the command was executed.</td>
</tr>
<tr>
<td>touch</td>
<td>Updates the access and modification times of a file.</td>
</tr>
<tr>
<td>tty</td>
<td>Writes the full path name of your terminal to standard output.</td>
</tr>
<tr>
<td>type</td>
<td>Writes a description of the command type.</td>
</tr>
<tr>
<td>ulimit</td>
<td>Sets or reports user resource limits.</td>
</tr>
<tr>
<td>uname</td>
<td>Displays the name of the current operating system.</td>
</tr>
<tr>
<td>unpack</td>
<td>Restores compressed files.</td>
</tr>
<tr>
<td>uniq</td>
<td>Deletes repeated lines in a file.</td>
</tr>
<tr>
<td>units</td>
<td>Converts units in one measure to equivalent units in another measure.</td>
</tr>
<tr>
<td>users</td>
<td>Displays a compact list of the users currently on the system.</td>
</tr>
<tr>
<td>vedit</td>
<td>Edits files on a full screen display.</td>
</tr>
<tr>
<td>vi</td>
<td>Edits files on a full screen display.</td>
</tr>
<tr>
<td>vmstat</td>
<td>Reports virtual memory statistics.</td>
</tr>
<tr>
<td>whatis</td>
<td>Describes what function a command performs.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>which</strong></td>
<td>Locates a program file, including aliases and paths.</td>
</tr>
<tr>
<td><strong>who</strong></td>
<td>Identifies the users currently logged in.</td>
</tr>
<tr>
<td><strong>whoami</strong></td>
<td>Displays your login name.</td>
</tr>
<tr>
<td><strong>xargs</strong></td>
<td>Constructs parameter lists and runs a command.</td>
</tr>
<tr>
<td><strong>yace</strong></td>
<td>Generates a LALR(1) parsing program from input, which consists of a context-free grammar specification.</td>
</tr>
<tr>
<td><strong>yes</strong></td>
<td>Outputs an affirmative response repetitively.</td>
</tr>
<tr>
<td><strong>zcat</strong></td>
<td>Expands a compressed file to a standard output.</td>
</tr>
</tbody>
</table>
Other Open Source Software for AIX

This appendix provides information on other sources for Open Source Software.
Overview

Open source software is generally software whose source is available to all without restrictions on use. The Linux kernel, and the GNU software packages, are the most well-known examples of open source software. This software is commonly distributed with the source code and the executable program included in one package. It also includes the license and agreement that will allow modifications, enhancements, and redistribution of the software under the same license terms. For more information about licenses, see Section 1.2.2, “About Linux's copyright” on page 5.

This alternative method of software development and distribution gives people, who are interested in the program, a possibility to contribute to the program by reporting errors and bugs or fixing problems on their own. Since different people have diverse techniques on tracing problems, the product is continuously enhanced, becoming much more robust and reliable.

Other sources

As discussed in Chapter 2, “AIX Toolbox for Linux Applications” on page 13, the AIX Toolbox contains many kinds of software that is commonly used in Linux systems. Because of the deeper integration with the basic AIX operating system, the AIX Toolbox should be used as the main source for Open Source Software.

However, there are Web sites that provide Open Source Software for AIX. Most of this software is distributed in .bff or .tar format. .bff format has the installation image file on an installation media that is used by installp (or SMIT) (see Section 2.5.1, “AIX installp” on page 24 for more details).

Two examples of open source software Web sites where you can download a wide ranges of software in addition to the AIX Toolbox are:

- Bull®
  http://www-frec.bull.fr/docs/download.htm
  http://www-frec.bull.fr/cgi-bin/list_dir.cgi/download/aix432/

- UCLA - University of California, Los Angeles
  http://aixpdslib.seas.ucla.edu/aixpdslib.html
Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 198.

- *IBM Certification Study Guide AIX 4.3 System Administration*, SG24-5129

Other resources

These publications are also relevant as further information sources:


Referenced Web and FTP sites

These Web sites are also relevant as further information sources:


http://linuxppc.org/ - Linux on PowerPC Web site


http://www.gnu.org/ - GNU Project Web site

http://www.trolltech.com/ - TrollTech Web site

http://www.rpm.org/ - Red Hat Package Manager Web site


http://www.gnome.org/ - GNOME Organization Web site

http://www.gnu.org/ - GNU Project Web site

http://www.trolltech.com/ - TrollTech Web site

http://www.rpm.org/ - Red Hat Package Manager Web site

http://sunsite.dk/zsh/ - Z Shell Web site


http://techsupport.services.ibm.com/rs6000/support - IBM RS/6000 Support Web site


www.wget.org - wget Web site

www.ncftp.com/ncftp/ - ncFTP Web site


http://rpmfind.net - RPMfind Web site


- [www.rpm.org/RPM-HOWTO/index.html](http://www.rpm.org/RPM-HOWTO/index.html) - RPM HOWTO Web site
- [www.rpm.org/maximum-rpm.ps.gz](http://www.rpm.org/maximum-rpm.ps.gz) - *Maximum RPM* by Ed Bailey download site
- [www.rpmdp.org/rpmbook](http://www.rpmdp.org/rpmbook) - *Maximum RPM* by Ed Bailey Web site
- [www.gnu.org/prep/standards_toc.html](http://www.gnu.org/prep/standards_toc.html) - GNU Coding Standards Web site
- [www.gnu.org/software/autoconf](http://www.gnu.org/software/autoconf) - GNU autoconf Web site
- [http://fvwm.org](http://fvwm.org) - FVWM Web site
- [http://xwinman.org](http://xwinman.org) - Xwinman Web site
- [www.gnu.org/software/libtool](http://www.gnu.org/software/libtool) - GNU libtool Web site
- [www.gtk.org](http://www.gtk.org) - GIMP Toolkit Web site
- [www.zsh.org](http://www.zsh.org) - zsh Web site
- [www.linuxbase.org](http://www.linuxbase.org) - Linux Standard Base Web site
- [http://www-frec.bull.fr/docs/download.htm](http://www-frec.bull.fr/docs/download.htm) - Bull's Large Freeware and Shareware Archive for AIX 4 Web site
- [http://www-frec.bull.fr/cgi-bin/list_dir.cgi/download/aix432/](http://www-frec.bull.fr/cgi-bin/list_dir.cgi/download/aix432/) - Bull's downloadable resources for AIX 4.3.2 Web site
How to get IBM Redbooks

Search for additional Redbooks or redpieces, view, download, or order hardcopy from the Redbooks Web Site

ibm.com/redbooks

Also download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web Site for information about all the CD-ROMs offered, updates and formats.
Special notices

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of a specific Statement of General Direction.
The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere., The Power To Manage., Anything. Anywhere., TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

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

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

Linux is a registered trademark of Linus Torvalds.

GNU Project, GNU, GPL and all GNU-base trademarks and logos are trademarks or registered trademarks of the Free Software Foundation.

POSIX is a trademark of the IEEE (Institute of Electrical and Electronics Engineers, Inc.).

BSD is a registered trademark of Berkeley Software Design, Inc.

OSF/Motif is a registered trademark of the Open Software Foundation.

Red Hat and RPM are trademarks of Red Hat Software.

SuSE is a registered trademark of SuSE Linux AG.
Caldera, OpenLinux and all OpenLinux-base trademarks and logos are trademarks or registered trademarks of Caldera.

Corel Linux and all Corel Linux-base trademarks and logos are trademarks or registered trademarks of Corel.

Mandrake and Linux Mandrake are registered trademarks of MandrakeSoft SA and MandrakeSoft Inc.

TurboLinux and all TurboLinux-base trademarks and logos are trademarks or registered trademarks of TurboLinux.

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

Yellow Dog Linux is trademark of Terra Soft Solutions.

Slackware is a registered trademark of Slackware Linux, Inc.

Bull is a registered trademark of Bull S. A.

Apple, Macintosh and Mac OS are trademarks of Apple Computer, Inc.

XWindows is a trademark of MIT.

Trolltech and Qt are trademarks of Trolltech AS.

KDE, K Desktop Environment, KChart, KFormula, KIllustrator, KOffice, KPresenter, Krayon, KSpread and KWord are trademarks of KDE e.V.

CDE, Java, SOLARIS, and Ultra are trademarks of Sun Microsystems, Inc.

SPARC trademarks refers to SPARC and related marks owned by SPARC International, Inc. and licensed to Sun for use on products based upon a particular architecture.

Alpha is a trademark of Compaq Computer Corporation.

Intel, IA-32, IA-64, Itanium, and all Intel-base trademarks and logos are trademarks or registered trademarks of Intel Corporation.

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

Symbols
/opt/freeware 18
/usr/opt/freeware 18

A
affinity between Linux and AIX
(See AIX Toolbox for Linux Applications)
AIX 1, 2
  affinity between Linux and AIX 8
  boot process 143
  installation method 22
  installp command 24
  Linux Applications on AIX 8
  new features 3
  ODM 22
strategy 8
  trends and directions 10
  useful URLs 53
  version 2
  XCOFF 56
AIX 5L 8, 10
AIX Toolbox for Linux Applications 13
  benefits 15
  content 16
  design 17
  documentation 13, 16
  main reasons for using it 15
  Open Source Software 16
  Overview 15
  RPM 15
  useful URLs 53
APIs
  compatibility 55
  fcntl 64
  LSB (Linux Standard Base) 57, 153
  on AIX 55
  poll 65
  similarities and differences 55

B
build environment 79

C
Caldera 6
CDE 21
commands and tools
  adduser 134
  administration tools 134
  basic utilities 43
  cpio 49
  diag 129
  differences 89
  dodisk 149
  fsck 149
  ftp 41
  GnomeRPM 33
  GNU 17
  installp 23, 24, 25
  KPackage 35
  libtool
    (see libtool)
  Linuxconf 134
  lslpp 38
  ltconfig 83
  make 76
  mkfs 149
  mount 150
  ncf tp 41
  passwd 134
  RPM
    (see RPM)
  rpm2cpio 49
  sar 129
  shell
    (see shell)
  SMIT 8, 26, 134
  touch 59
  umount 150
  VSM 109
  wget 41
  YaST 134
configuration files
  /etc/filesystems 149
Corel 6
Debian 6
deinstall package 47
directory structure 17
   /opt/freeware 18
   /usr/opt/freeware 18

ELF 56

file access modes 60
Free Software Foundation 5

GNU 5
   General Public License (GPL) 5
   GNU coding standards 81
   GNU Project 5
   useful URL 5

installation
   getting individual files out of a package file 49
   GNOME 40, 52
   KDE2 40, 50
   process 23
   RPM 23, 39
   system requirements 38
   Toolbox 23, 37, 39
   Toolbox base 43
   useful URLs 37, 53

installation methods 22
   command line 22
   GnomeRPM 22
   installp 23
   KPackage 22
   SMIT 22

KDE2 21
   installation 40, 50

starting 51
KPackage 35

libtool 83
   building applications 69
   usage 83
LILO (Linux LOader) 142
Linux 1, 4
   affinity between Linux and AIX 8
   at IBM 7
   flavors and distributions 6
   Linux Applications on AIX 8
Linuxconf 134
LSB (Linux Standard Base) 57, 153

macro values 58
Mandrake 6

Open Source Software (OSS) 15

package building 69
package rebuilding 72
portable code 56
porting 69

Red Hat 6
Redbooks Web Site 198
Contact us xv
RPM 15, 22, 30, 69
   command line 31
   database 24
   GnomeRPM 33
   installation 39
   KPackage 35
   package label convention 31
   spec file 71, 72
   using 44
RS/6000 2

scripts
/etc/rc.d/rc.sysinit 142
destroyRPMS 48
ltconfig 83
rc 142
rc.boot 143
startkde 91
shell 112
bash 115
bsh 112
csh 112
ksh 113
psh 113
Rsh 113
sh 113
startup files 123, 133
tcsh 117
tsh 114
zsh 120
Smit 134
source compatibility 55
SRPM 22, 69
standards
  POSIX 15
  UNIX98 15
  X/Open 15
SuSE 6
system variables 20
  MANPATH 21
  PATH 20, 70

T
  Toolbox
    (See AIX Toolbox for Linux Applications)
troubleshooting
  corrupt package files 49
  running out of disk space 46
TurboLinux 6

U
  UNIX98 56

V
  Visual Age 21

X
  XCOFF 56
Running Linux Applications on AIX
Running Linux Applications on AIX

AIX affinity with Linux

AIX Toolbox for Linux Applications

Porting and source compatibility

The strengths of the AIX operating system are well known among the UNIX software community. Its reliability and great degree of scaling makes AIX the perfect choice for hosting mission-critical applications. It is a robust and flexible operating system that meets all the requirements for the current demands of e-business environments. At the same time, Linux is emerging and generating excitement among software developers that has not been seen in years.

This redbook presents the AIX Toolbox for Linux Applications. The toolbox contains a collection of open source and GNU software built for AIX 4.3.3 and AIX 5L 5.1 for IBM @server pSeries systems and IBM RS/6000. It provides the basics for the development environment of choice for many Linux application developers. All the tools are packaged using the easy-to-install RPM format. There is a strong affinity between Linux and AIX for applications. AIX has a long history of standards compliance and it is generally straightforward to rebuild Linux applications for AIX. Now you can easily port these applications and run them directly on AIX while taking advantage of all the features and benefits that the AIX operating system offers.

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