Draft Document for Review April 12, 2002 11:23 am

SG24-6551-00

Linux Integration with OS/400

on the IBM @server iSeries Server

- Learn the advantages of integrating Linux with OS/400
- Access DB2 UDB for iSeries data from Linux applications
- Install Linux for the intranet using NetServer

Sungsim Park
Abhijit Chavan
Jian Hui Chen
Larry Dunn
Takaki Kimachi
Alexander Marquis
Patrick Wildt

Redbooks

ibm.com/redbooks

International Technical Support Organization

Linux Integration with OS/400 on the IBM @server iSeries Server

December 2002

Take Note! Before using this information and the product it supports, be sure to read the general information in “Special notices” on page ix.

First Edition (December 2002)

This edition applies to Version 5, Release 1 of OS/400.

This document created or updated on April 12, 2002.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JLU Building 107-2
3605 Highway 52N
Rochester, Minnesota 55901-7829

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

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.

IBM trademarks

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

e (logo)® AFP™
AS/400®
Balance®
DB2®
DB2 Universal Database™
Home Director™
IBM®
IBM.COM™
Infoprint®
Informix™
IPDS™
iSeries™
OS/400®
Perform™
PowerPC®
Redbooks™
Service Director™
SP™
VisualAge®
xSeries™
zSeries™
Redbooks (logo)™

Other company trademarks

The following terms are trademarks of other companies:

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.

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

Preface

The IBM @server iSeries server offers many points of integration that support Linux applications leveraging OS/400 applications and data. This includes iSeries ODBC Driver for Linux, OS/400 NetServer with Samba support, IBM Java Virtual Machine, and JTOpen support.

iSeries ODBC Driver for Linux provides access from a Linux application running on iSeries to the DB2 UDB for iSeries database. OS/400 NetServer supports the exchange of files between OS/400 and Linux, and Linux applications can print to OS/400 print queues. IBM Java Virtual Machine is available for iSeries Linux. This JVM in combination with the iSeries Toolbox for Java or its OpenSource version, JTOpen, allows developers to access DB2 UDB for iSeries data via JDBC as well as leverage OS/400 programs and services.

This IBM Redbook covers each integration technique with sample code. It also shows you the methods that you can use to integrate Linux applications with OS/400. This redbook is intended to help beginner and intermediate Linux users, with an OS/400 background, to integrate the Linux application with OS/400 application and data.

The team that wrote this redbook

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

Sungsim Park is an ITSO Specialist at the International Technical Support Organization, Rochester Center. She has over 16 years of experience in working with S/36, S/38, and iSeries Servers. Before joining ITSO in 2000, she taught IBM classes on all areas of iSeries as a senior education specialist in IBM Korea and provided technical marketing support to AS/400 Sales Representatives, IBM Business Partners and Customers. Her areas of expertise include server consolidation and application development. She can be reached at: sungsim@us.ibm.com

Abhijit Chavan is a Systems Engineer in Starcom Software Pvt Ltd, Mumbai, India. Starcom Software is a Business Partner of IBM and is involved with Web Applications Development and Network Infrastructure Management on Unix/Linux systems. Abhijit has two years of experience in workding on Linux. His present job is as a Linux Systems Administrator and his areas of expertise include TCP/IP networking, network management, and security with services like Apache http, Sendmail, Bind, Samba, SSH, and Linux Firewall. He has two years of experience in programming with C and 1 year with Perl and Shell. He holds a degree in Computer Engineering from University of Mumbai (Bombay). He can be reached at abhijit.chavan@starcomsoftware.com

Jian Hui Chen is a IT specialist from IBM China. He has four years of experience in iSeries and zSeries solution development field. Over the last two years, he supported customers and ISVs porting their applications to the iSeries platform from other DBMS, such as Oracle, Sybase, and Informix. He also provided technical marketing support to the iSeries sales team, and hold the technical trainning for customers/BPs. He holds a bachelor degree in Computer Science from Beijing Polytechnic University. Before joining IBM, he worked as a billing systems developer for ChinaTelecom. His areas of expertise include DB2 UDB, WebSphere, Linux OS, etc. He can be reached at: jianhuic@cn.ibm.com

Larry Dunn is an Advisory IT Specialist with IBM in the United Kingdom. He has seven years of experience with iSeries, learning Linux in his spare time. The last five years he has spent with specialising in iSeries printing, rising to the role of iSeries Printing Technical Advisor within the IBM Software Support OS/400 Center. He is also a member of the IBM Technical Council in the UK. Larry holds a Master of Science (MSc) degree in Information Systems from the University of Portsmouth. His areas of expertise include Linux OS, anything related to iSeries printing, LPAR and TCP/IP. He has written extensively on iSeries printing, Facsimile Support and TCP/IP and PSF trace interpretation. He can be reached at: larry_dunn@uk.ibm.com

Takaki Kimachi is a Senior Advisory IT Specialist in IBM Japan. He has experience in iSeries and S/36 area in 14 years. He had been a field SE supporting iSeries and AS/400 customers. For the past two years, he has worked as a Techline Japan specialist supporting IBM Business Partners and IBM Representatives with iSeries hardware or LPAR. He developed the prototype of LPAR Validation Tool (LVT) in 2000, which is the LPAR configuration tool used world wide today. He has been a Linux user for seven years and a member of the IBM Japan iSeries Linux task team one year. He can be reached at: E05213@jp.ibm.com

Alexander Marquis is a Staff Software Engineer from the USA. He has five years of experience with the iSeries and has spent the last two years supporting Client Access Data Transfer, APIs, ODBC, and NetServer in the Rochester Support Center. He holds a degree in Interdisciplinary Studies from the University of Minnesota - Duluth where he was introduced to POSIX-style operating systems. His areas of expertise include various Data Access methods to the iSeries, NetServer, and the Linux OS. He can be reached at: marquis@us.ibm.com

Patrick Wildt is a Staff Software Engineer from the USA. He has 13 years of experience on the iSeries platform with eleven years spent working in the OS/400 development lab in the Systems and Network Management area. The last two years have been spent in the eServer Custom Technology Center working on various customer contracts and engagements. He holds a bachelors degree in Computer Science from North Dakota State University. His areas of expertise include systems management and communications. He can be reached at: wildt@us.ibm.com

Thanks to the following people from IBM Rochester for their contributions to this project:

David Boutcher
Pamela Bowen
Monte Bruesewitz
Jay Bryant
Keith Cooper
Selwyn Dickey
Erwin Earley
David Engebretsen
Steven Janssen
Craig Johnson
Brian King
Larry Loen
Vess Natchev
Brent Nelson
Carl Pecinovsky
Jeffrey Scheel
Kay Tate
Dennis Towne
Mark Vanderwiel
Dave Wall

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to the address on page ii.

1

Introduction to Linux integration

This chapter provides a brief overview of the integration of Linux with OS/400 on the iSeries server. It discusses the following areas:

- Advantages of integrating Linux with OS/400
- Overview of Linux integration techniques

1.1 Advantages of integrating Linux with OS/400

IBM is committed to supporting the Linux operating system on the iSeries platform. There is the capability of installing Linux onto one or more iSeries logical partitions, therefore, reducing the capital expenditure of purchasing extra hardware. This configuration is covered in detail in *Linux on iSeries: An Implementation Guide,* SG24-6232. IBM has now further enhanced its support of Linux by developing several methods by which the actual data and applications running on both Linux and OS/400 can be integrated. An example of this is the Linux ODBC driver that enables a Linux application to access data in an OS/400 database.

The ability to integrate Linux and OS/400 applications on the iSeries draws together the main strengths of both operating systems. For example, the database capabilities of OS/400 and the Web serving qualities of Linux, provide a powerful, consolidated computing environment.

Furthermore, in addition to Web serving, the nature of Linux makes it an ideal foundation on which to run many other application types, such as: file serving, e-mail, and firewall and network management. Integrating Linux with OS/400 on the same iSeries platform will result in a very flexible, economic, and comprehensive processing solution.

Traditionally, the iSeries server (and its predecessor, the AS/400) has been used as a database machine by customers wishing to store and retrieve vast amounts of information relating to their business. *DB2 Universal Database for iSeries (DB2 UDB)* is fully integrated into the OS/400 operating system. Customers can also benefit from the single level storage, security, object, file and print management, reliability and scalability of the iSeries. Unlike some other system types, as a customer's business expands, the iSeries system hardware can be upgraded to accommodate the requirement for greater processing power or storage space.

Linux is recognized as an e-business system. At present a high percentage of business implementations of Linux are Web servers using the Apache Web Server and TomCat Web Application Server. The Linux operating system provides an excellent environment in which to run applications based on Hypertext Pre-Processor (PHP) and Perl.

However, the following implementation types are also popular:

- Mail serving with products like Sendmail mail gateway
- File and Print Serving with Samba
- Firewall with NetFilter

Given the world-wide development community and the open source nature of Linux, new and appealing applications reflecting the ever-changing domain of e-business can be developed relatively quickly and economically. Applications requiring access to a database may now be enhanced by the ability to access an existing DB2 UDB database running on an OS/400 logical partition, thus reducing development time and costs.

1.2 Overview of Linux integration techniques

This section presents an overview of the methodologies that can be used to integrate the Linux operating system with OS/400. The techniques and methods are expanded in subsequent chapters of this book and in some cases, in conjunction with sample program code.

1.2.1 The iSeries ODBC Driver for Linux

The iSeries ODBC Driver for Linux allows access to an iSeries DB2 UDB database from a Linux application. This driver is based on the ODBC driver shipped with the *Client Access Express for Windows (5722-XE1)* product and similarly accesses the system using the Host Servers database interface via a socket connection.

The ODBC Driver for Linux comes in two different versions. Figure 1-1 shows the Power PC version running on an iSeries logical partition.

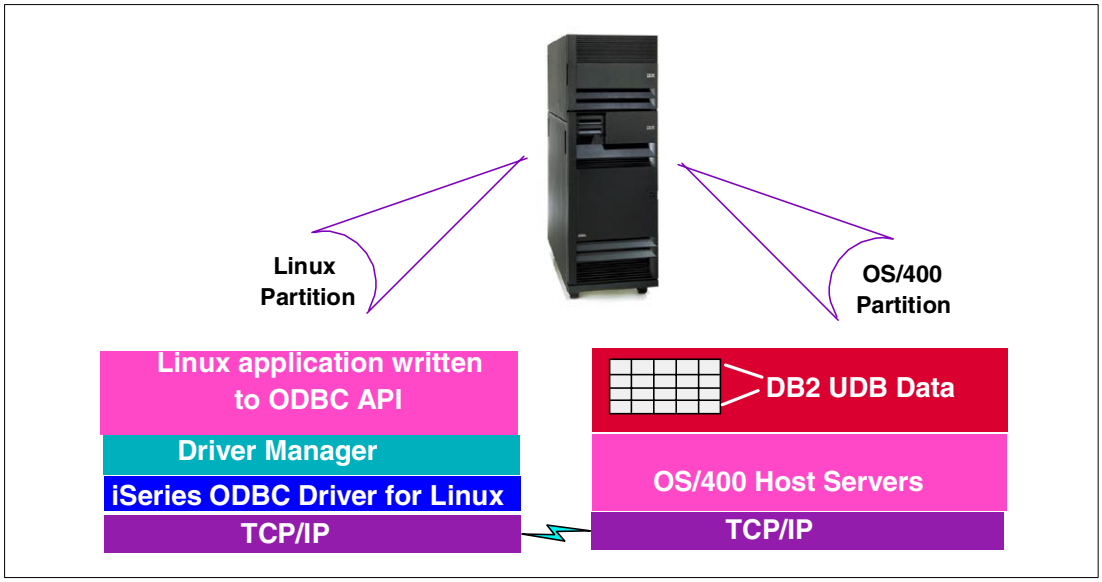

Figure 1-1 ODBC Driver for Linux version running on an iSeries logical partition

For Linux systems installed on Intel-based systems, another version of the ODBC driver is available. See Figure 1-2.

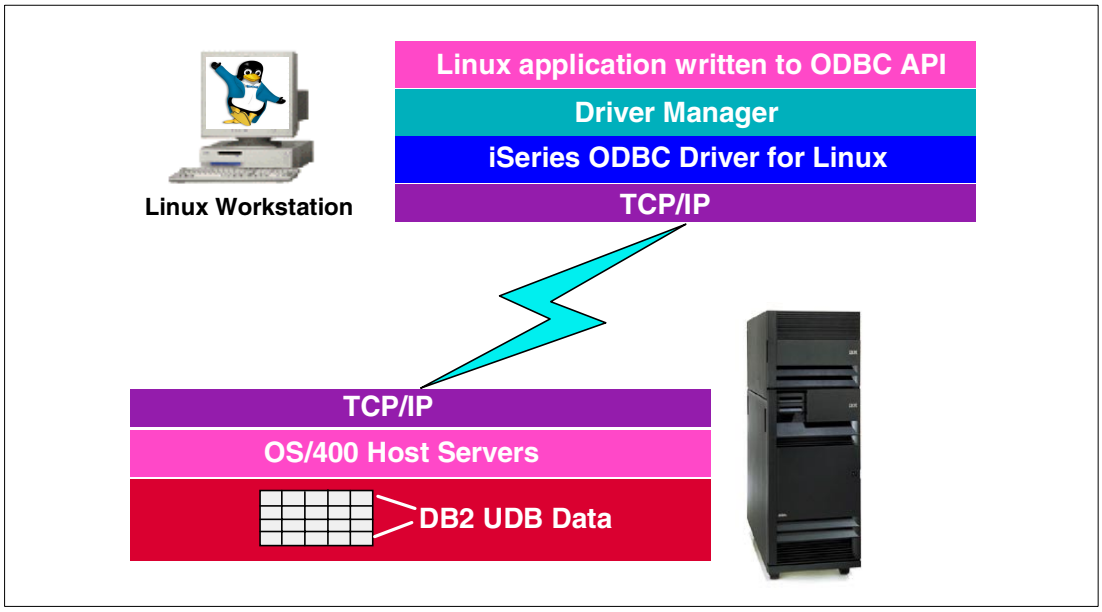

Figure 1-2 ODBC Driver for Linux Intel systems

For further details on using the ODBC Driver for Linux, see:

- Chapter 2, “iSeries ODBC Driver for Linux” on page 7, which documents installation, configuration, and sample program code demonstrating its use in a real life situation.
- *iSeries ODBC Driver for Linux Installation and Usage Guide* has a vast amount of information on the ODBC for Linux Driver ranging from system requirements and configuration through to restrictions and troubleshooting: `http://www.ibm.com/eservers/iseries/linux/odbc`

1.2.2 AS/400 NetServer support for Linux clients using Samba

Support has now been added to V5R1 of OS/400 allowing Linux Samba clients, using the Server Message Block (SMB) protocol, to access iSeries NetServer file and printer shares. The NetServer is in fact a SMB server by a different name. This support is attained using the following Samba utilities:

- **smbclient**: Provides FTP type functionality.
- **smbmount**: Can present a transparent, hierarchical view of the iSeries Integrated File System (IFS). Its use is similar to mapping a network drive from a Windows client. Figure 1-3 shows an example view of a NetServer share from a Linux GUI display. In this example, *dunnl* is a Linux directory, and *dunnmount* s the Samba mountpoint. The indented directories, such as *dunnlDBfilesOniSeries*, are also on the iSeries server.

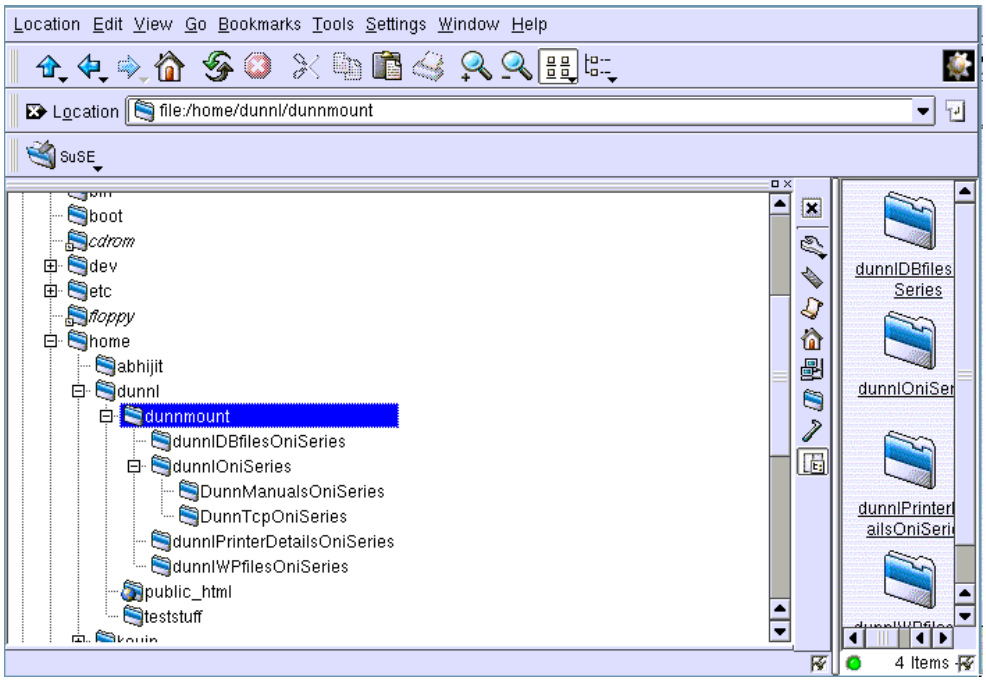

Figure 1-3 Example view of Netserver share from a Linux graphical display

For accessing an iSeries printer, one of the various Linux GUI applications allows easy configuration of a Linux Samba printer session.

You can find further information and examples on how to implement Samba support for Linux by consulting the following sources:

- Chapter 3, “AS/400 NetServer support for Samba” on page 27, demonstrates how to use the **`smbmount`** and **`smbclient`** utilities.

- 3.7, “Sharing an iSeries printer using Samba” on page 40, documents how a Linux system can access an iSeries configured printer.
- AS/400 Support for Windows Network Neighborhood presents extensive information on AS/400 NetServer support for Linux ranging from the required iSeries PTFs and Linux kernel, to Samba levels and background details on Samba:
  http://www-1.ibm.com/servers/eserver/iseries/netserver/linux.htm

1.2.3 Linux integration using the JDBC driver

IBM Toolbox for Java (5722-JC1) contains a JDBC driver that allows you to use JDBC API interfaces to issue structured query language (SQL) statements to and process results from the iSeries database. The Toolbox for Java also contains a set of Java classes enhancing the ease in which programs can be developed to access iSeries data and resources. This JDBC driver is used in a client/server environment. For example, in the case of a Linux client (running on an OS/400 partition), the Java application and the Java classes, using the JDBC driver can access DB2 UDB data residing on the iSeries server.

Note: There is also a *native* JDBC driver. This is shipped with the *AS/400 Developer Kit for Java* and runs on the OS/400 Java Virtual Machine (JVM). However, this driver is only used when the Java application and database are on the same OS/400 partition.

Refer to Chapter 4, “Linux integration with JDBC” on page 49, to learn more about the JDBC driver. Sample program code demonstrating its use is also presented.

You should also go to the following Web site for information on the JDBC driver:
http://www.ibm.com/eservers/iseries/toolbox

1.2.4 Other access methods

In addition to the Linux ODBC driver and Samba support, other traditional methods exist to facilitate the integration of Linux and OS/400 applications and data. Here are examples of such techniques:

- FTP
- Programs using Toolbox for Java
- Programmed socket connections
- Telnet

These techniques are discussed in further detail in Chapter 5, “Other access methods” on page 67:

1.2.5 Advantages of using the virtual LAN and virtual DASD

A significant additional benefit of installing Linux onto an iSeries partition is the ability of the Linux partitions to make use of virtual LAN and virtual DASD. In many instances, this is advantageous compared to using the native LAN and native DASD options. For example, the benefits range from reduced hardware costs through an improvement in data retrieval performance and enhanced disk protection.

You can learn more details about this topic in Chapter 6, “Advantages of using virtual LAN and virtual DASD” on page 83.

1.2.6 NFS export of Linux files from OS/400 Integrated File System

This technique is adapted from the implementation of Linux on diskless workstations. It is useful in situations where two or more “similar” Linux partitions are configured. The concept is to install a “footprint” of the common file system or systems onto the OS/400 IFS. These files can then be shared with any of the Linux partitions.

One benefit of this strategy is that an iSeries server can be sized more economically. For example, the Network Server Storage Space for each partition may only need to be 500 MB, rather than several GB in size, therefore, leading to a reduction in the required DASD.

You can learn more about this topic in Chapter 7, “NFS export of Linux files from OS/400 Integrated File System” on page 87.

2

iSeries ODBC Driver for Linux

This chapter introduces the iSeries ODBC driver for Linux and its installation and usage in both 2-tier and 3-tier environments. It discusses the following topics:

- Introduction
- Requirements
- Installation
- DSN configuration
- Accessing theDB2 UDB for iSeries via OpenOffice and ODBC
- Introduction to 3-tier applications using PHP and Apache
- Simple ODBC application written in C
- Troubleshooting

2.1 Introduction to the iSeries ODBC Driver for Linux

The iSeries ODBC Driver for Linux allows you to access the DB2 UDB for iSeries from any Linux application written to the ODBC APIs. It is an ODBC 3.5 ANSI driver with the ability to store and process Unicode data and is based on the Client Access Express ODBC driver. Just like the Client Access Express ODBC driver, it uses a TCP/IP socket connection to the database host server to access data on the iSeries.

There are two versions of the iSeries ODBC driver for Linux. One for the Power PC (PPC) architecture and another for the Intel/i386 platform. When Linux has been installed on a partition of the iSeries, the PPC version is used. When Linux is installed on a standalone, intel architecture PC, the i386 version is needed.

Just like Microsoft Windows, Linux requires a driver manager application to manage the various data sources and load the ODBC drivers. On Linux, there are several different ODBC driver managers. The ODBC driver manager that the iSeries ODBC Driver for Linux works with is the unixODBC driver manager.

ODBC Driver for Linux versus Microsoft WIndows ODBC driver

Table 2-1 describes some differences between the Linux ODBC implementation and the MS Windows ODBC driver that is packaged with Client Access Express.

Table 2-1 Linux and MS Windows ODBC differences

Linux ODBC	MS Windows ODBC
The ODBC driver supports the ODBC 3.5 specification, but is not Unicode driver. The only real restriction in the driver not being a Unicode driver is that the driver does not allow Unicode SQL statements to be passed on to the server.	The driver is an ODBC 3.5 Unicode driver. A Unicode driver accepts Unicode strings as arguments to the APIs
To sign on, you must specify a user ID and password when calling the connection API or have the user ID and password entered into the DSN. The ODBC driver will not prompt for iSeries user IDs or passwords. User ID and password updates must be done through a Telnet session with the iSeries. If a user's password has expired, a dialog is displayed to allow a user to change it.	The user has signon options that control which user ID and password to use when connecting. When connecting, cached passwords might be used.
When binding a parameter or a column with SQL_C_WCHAR as the C type, wchar_t buffers should *not* be passed in. wchar_t is defined as 4-bytes in Linux and 2-bytes in Windows.	When binding a parameter or a column with SQL_C_WCHAR as the C type, wchar_t buffers should be passed in. The driver manager and driver both handle the SQL_C_WCHAR data type as a 2 byte UCS-2 string.

2.2 Requirements

Before you attempt to use the iSeries ODBC driver for Linux, please verify that you meet the following requirements.

2.2.1 Linux requirements

- The client must be running one of the versions of Linux which contains glibc level 2.2 or later such as:
  - Red Hat for Intel version 7.x
  - Red Hat for iSeries version 7.x
  - SuSE for Intel version 7.x
  - SuSE for iSeries version 7
  - TurboLinux for Intel version 7.x

    TurboLinux 6.5 for PPC contains a version of libc that is not compatible with the iSeries ODBC driver. Later versions of TurboLinux for iSeries are expected to support the iSeries ODBC driver for Linux.

    To check the dependencies for installing the package we can use the command:

    ```
    # rpm -qpR iSeriesODBC-5.1.0-0.10.i386.rpm
    ```
- unixODBC driver manager versions 2.0.11 and above are supported for use with the iSeries ODBC driver for Linux. The Linux for iSeries distributions include pre-compiled, PPC versions of unixODBC and offer the easiest means of installation on the PPC architecture (using rpm). See the unixODBC Project Web site for more information on the driver manager: http://www.unixodbc.org/
- The iSeries ODBC driver for Linux uses a socket connection to the iSeries to exchange data. Therefore, TCP/IP must be properly configured for communication with OS/400.

2.2.2 OS/400 requirements

- The driver is supported only when connecting to servers running OS/400 Version 4 Release 5 or later version. The driver may work to earlier releases of OS/400, but is supported accessing only V4R5 and later version systems.
- TCP/IP must be running and properly configured.
- The host servers must be started. Type:

  ```
  STRHOSTSVR *ALL
  ```

  Press Enter to start the OS/400 host servers.

2.3 Installing the iSeries ODBC driver for Linux

1. Verify whether the iSeries ODBC driver is already installed with the command:

   ```
   # rpm -qa | grep ODBC
   ```

   If it is, check the version and see if it is the most current release.
2. If you do not have the most current release, download the latest ODBC driver for your architecture from: http://www-1.ibm.com/servers/eserver/iseries/linux/odbc/
   - If you are connecting to the iSeries server from an i386/Intel PC running Linux, download *iSeriesODBC-5.1.0-0.x.i386.rpm*.
   - If you are connecting to the iSeries server from a logical partition (LPAR) running Linux, download *iSeriesODBC-5.1.0-0.x.ppc.rpm*.
3. Uninstall previous versions of the iSeries ODBC driver for Linux with the command:

   ```
   # rpm -e <RPM module name>
   ```

   Prior versions of the iSeries ODBC driver for Linux should be removed before you install newer versions.

4. Install the driver with the command:

   ```
   # rpm -ivh <RPM module name>
   ```

   On Red Hat systems, it's common for a dependency error to occur on *libodbcinst.so.1*. See Figure 2-1.

```
[root@sander /install]# rpm -ivh iSeriesODBC-5.1.0-0.x.i386.rpm
error: failed dependencies:
        libodbcinst.so.1   is needed by iSeriesODBC-5.1.0-0.1
[root@sander /install]#
```

Figure 2-1 Failed dependencies

   If unixODBC is installed (compiled from source or via .rpm), this particular dependency can be ignored. Add `--nodeps` to the end of the **`rpm`** command:

   ```
   # rpm -ivh iSeriesODBC-5.1.0-0.x.i386.rpm --nodeps
   ```

5. Test the installation with the **`cwbping`** command:

   ```
   # /opt/ibm/iSeriesODBC/bin/cwbping <iSeries system name>
   ```

   If **`cwbping`** fails for any reason, the driver will likely not work. See 2.8, "Troubleshooting" on page 21, for possible causes.

> **Note:** The **`cwbping`** command is installed into /opt/ibm/iSeriesODBC/bin. To execute **`cwbping`** from any directory, your PATH shell variable needs to have an entry for /opt/ibm/iSeriesODBC/bin. See your Linux distribution documentation for details on shell variables.

2.4 Data Source Name (DSN) configuration

There are two options for configuring the iSeries ODBC driver:

- Using the graphical tool ODBCConfig
- Editing the .odbc.ini configuration file

The GUI tool ODBCConfig is modeled closely on Microsoft's ODBC administrator and is the best choice for those already familiar with ODBC in a Microsoft environment.

You must also choose the type of DSN. DSNs can be either a user DSN or a system DSN. User DSNs are stored in an .odbc.ini file in the home directory of the user that created the DSN, for example `/home/testuser/.odbc.ini`. They can only be accessed by the user that created them. System DSNs may only be created by root and are stored in the file `/usr/local/etc/odbc.ini` or `/etc/odbc.ini` (depending on the unixODBC version) and are accessible to all users on the system.

2.4.1 ODBCConfig

A data source can be configured using the ODBC data source graphical user interface (GUI). The GUI contains fields to set required and frequently used options. To create or configure an iSeries ODBC Driver for Linux DSN, follow these steps:

1. Open the Data Source Administrator that comes with the unixODBC driver manager by executing the ODBCConfig command in an X-windows terminal. The ODBC Data Source Administrator window should open as shown in Figure 2-2.

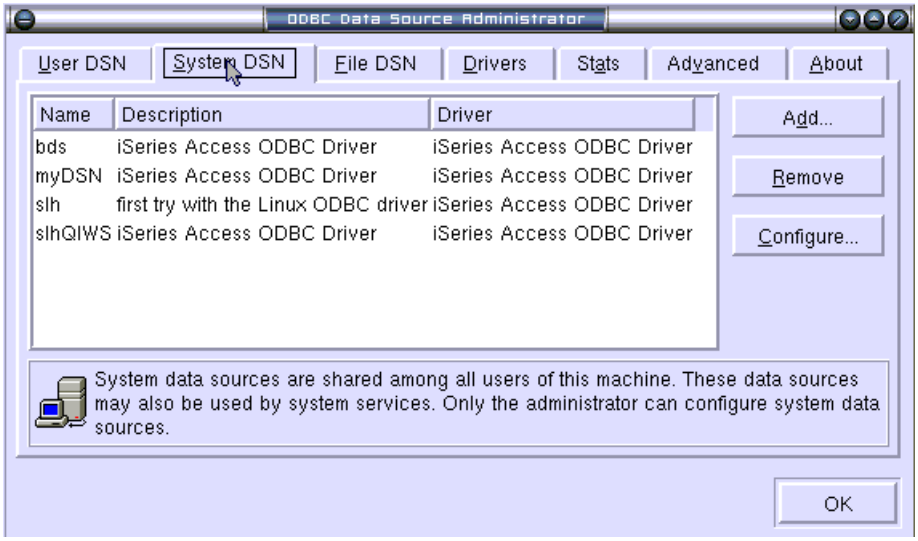

Figure 2-2 ODBCConfig

2. Decide what type of data source you want to use. User data sources (DSNs) are only accessible by the user who created them. System DSNs are accessible by any user on the machine, and must be configured by root
3. Click **Add...** to create a new data source or **Configure...** to configure a data source that already exists. If you clicked Configure..., go to step 5. Otherwise continue to the next step.
4. Select an ODBC driver. Click **iSeries Access ODBC Driver** and click the **OK** button.
5. A window similar to the example in Figure 2-3 appears. You may need to resize the window to see all the fields. A “tool-tip” help text appears if you leave the cursor over the input field.

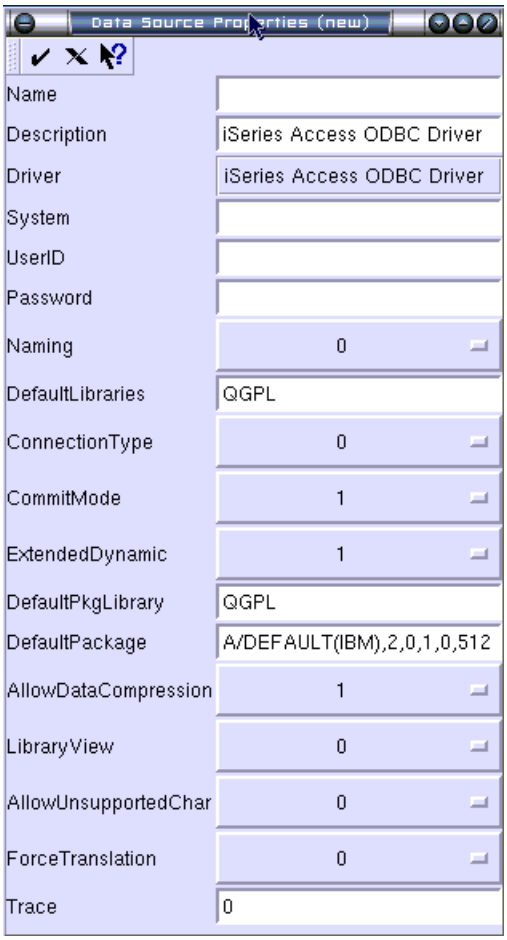

Figure 2-3 Data Source Configuration

6. Fill in the data source name in the Name field and your system name in the System field. All other fields are optional. If you are planning to use the OpenOffice walkthrough later in this chapter, you should change the Default Libraries setting to QIWS.
7. Click the **check mark** in the upper-left corner of the window to save the configuration. If this is a System DSN, the information is saved to /usr/local/etc/odbc.ini or /etc/odbc.ini.

2.4.2 Editing the .odbc.ini file

Some of the driver connection options are not available via the ODBCConfig GUI. In this case, you need to edit an .odbc.ini file or have the ODBC application send the connection options when connecting. See the full list of connection options at:
http://www-1.ibm.com/servers/eserver/iseries/linux/odbc/guide/odbcproperties.html

Follow these steps to manually add connection options to the odbc.ini file:

1. Open the .odbc.ini file on your workstation using a text editor such as **`vi`** or **`pico`**. For user data sources, this file is in the home directory of the user who created the data source. For example, the file for user "testuser" is /home/testuser/.odbc.ini. If there are multiple ODBC data sources in the .odbc.ini file, locate the section in the data source where you want to add the connection options.
2. Add a new row after the last entry in the data source and enter the new connection option and its value. The syntax is "keyword = value". For example, if you want to change the date

format from its default of 5 (yyyy-mm-dd or *ISO), to 1 (mm/dd/yy or *MDY), you would add "DFT = 1" in the new row.

3. Repeat step 2 to add additional connection options to the data source.
4. Save the .odbc.ini file.

Note: Do not add multiple entries for the same connection option to a section for a specific data source. This can lead to unpredictable behavior.

After manually editing the registry, you can still use ODBCConfig to configure your data source. Older versions of the unixODBC driver manager version had a problem where manually added options were removed from the odbc.ini file when ODBCConfig was used to configure the data source. If you experience this problem, install a newer version of the unixODBC driver manager.

Options specified by the application in the connection string override any options specified in the .odbc.ini file.

2.5 Accessing the iSeries DB2 UDB via OpenOffice and ODBC

OpenOffice.org is an open-source office productivity suite quite similar to Microsoft Office. The source code for the OpenOffice.org project was originally provided by the Sun StarOffice project. OpenOffice and Sun's StarOffice are functionally very similar and both include word processing, spreadsheet, presentation and graphics applications.

StarOffice has been successfully tested with the iSeries ODBC Driver for Linux. However, for the purpose of this demonstration, we use the open-source OpenOffice suite. The spreadsheet application includes a great deal of the same functionality as Microsoft Excel, including the ability to query remote databases via ODBC.

This section walks you through using OpenOffice.orgs' spreadsheet application to pull data from the iSeries DB2/UDB via ODBC.

2.5.1 Requirements

1. Before you use the OpenOffice.org ODBC functions, verify that the ODBC driver and the connections and DSNs that have been configured work. Use the cwbping utility to test connectivity with the iSeries and DataManager to verify the ODBC configuration.
2. A recent version of OpenOffice.org must be installed. The version of OpenOffice.org used in our testing was OpenOffice.org641. For the installation code, installation instructions, and other documentation, see: http://www.openoffice.org/

2.5.2 Instructions

1. From an X terminal, start OpenOffice. On our system, we execute:

   ```
   /usr/share/OpenOffice.org641/soffice &
   ```

   The *soffice* executable will likely be in a different location on your system.
2. Select **File-> New-> Spreadsheet** to open the Spreadsheet application. The word processing application that starts by default may be closed.
3. Select **Tools-> Data Sources...** and the OpenOffice Data Source Administration window (Figure 2-4) appears. This window helps you configure an OpenOffice data source. This

new data source is usable only by OpenOffice. It is *not* an ODBC DSN like the one created in the ODBCConfig tool.

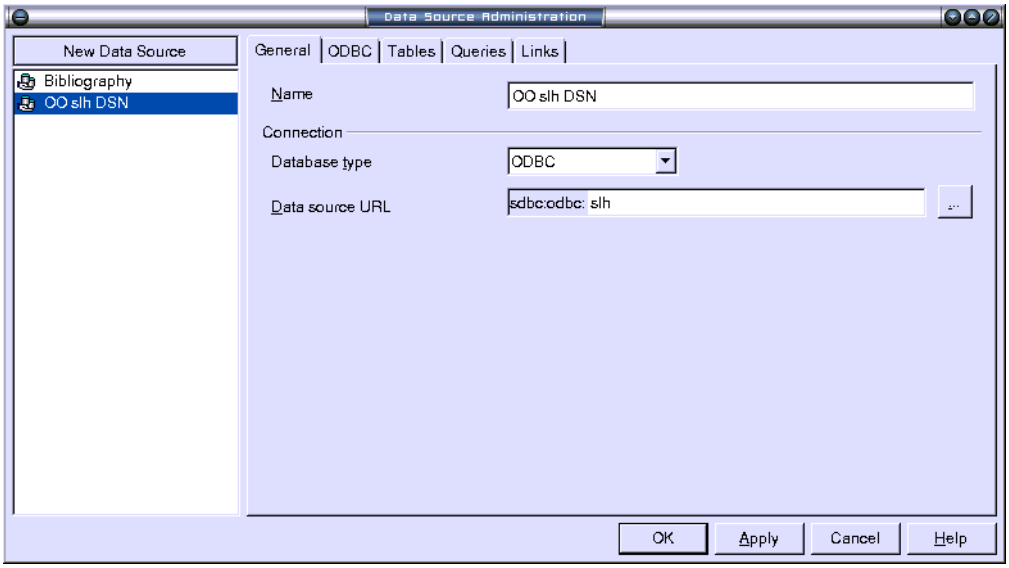

Figure 2-4 OpenOffice data source configuration

4. Click the **New Data Source** button in the upper-left corner. *Data source 1* appears in the list of OpenOffice data sources.
5. Give the new data source a name. Again, this data source is only available for use by OpenOffice.
6. Change the Database type to **ODBC**.
7. In the Data source URL parameter, click the browse button indicated by the ellipses (**...**). You are presented with a list of DSNs that you've configured in ODBCConfig.
8. Select the **DSN** that was configured for access to your iSeries database and click **OK**.
9. You may view the other data source configuration options available in the Data Source Administrator, but it is not necessary to change them. Click **OK** to finish configuring the Data Source.
10. You are now ready to bring the data into the spreadsheet. Click **Data-> DataPilot-> Start...**.
11. The Select Source window appears. Select **Data source registered in OpenOffice.org** and click **OK**.
12. Change the Database to the Data source you configured in step 5.
13. For the Data source parameter, select a table from the list. Select **QCUSTCDT** if it is listed.

Note: The list of tables you see is your default collection. The default collection consists of the iSeries Library for your job (based on your iSeries UID) and the default library you configured in the unixODBC DSN with ODBCConfig. The default library for the iSeries ODBC driver is QGPL. Since that has more tables than QIWS, for example, and QIWS contains the simple sample table QCUSTCDT, it may be worth changing the default Library to QIWS in the unixODBC DSN.

We found that by default, the OpenOffice.org ODBC application sends the system name as part of the collection and sends the library and file as the tablename, for example:

```
"SYSNAME"."LIB.TABLENAME"
```

This generates an SQL0204 error message. To get the collection and table name sent correctly, you must simply remove the System name from the Data source parameter. For example, change:

```
SYSNAME.MARQUIS.QCUSTCDT
```

to:

```
MARQUIS.QCUSTCDT
```

or simply:

```
QCUSTCDT
```

OpenOffice issue #3289 has been opened to address this issue.

14. Click **OK**.

    Now you see the DataPilot display (Figure 2-5). This window allows you to select fields and place them into the desired positions.

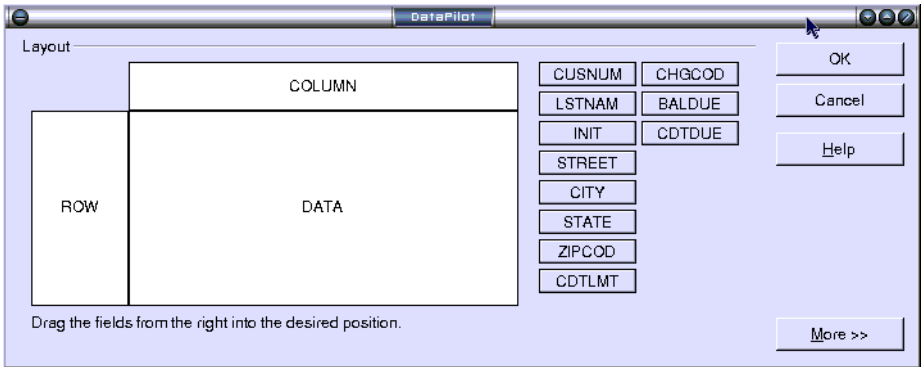

Figure 2-5 Data Pilot

Click and drag the fields from the right into either the Row or Column areas. Then, click **OK**.

15. You should now have a spreadsheet with iSeries data that you can save to any of the many OpenOffice.org data types.

This walkthrough demonstrates the ability of a Linux workstation to easily perform the same data-access functions as a MS Windows based PC, but without any license costs.

2.6 Three-tier application setup using PHP and Apache

This section gives a brief overview of how to setup a PHP/ODBC Web application on Linux that users can access via Web browsers.

What is PHP?

PHP stands for *PHP Hypertext Processor*. (It is a self-referencing acronym like GNU - *GNU is Not UNIX*). PHP is a widely-used general-purpose scripting language that is especially well-suited for Web development. It is commonly embedded into HTML.

What PHP can do

PHP is mainly focused on server-side scripting, so you can do anything any other CGI programs can do, such as collect form data, generate dynamic page content, or send and receive cookies. These tasks are the most common uses and the main target for PHP development. To make this work, you need:

- The PHP parser (CGI or server module)
- A webserver
- A Web browser

You need to run the webserver with a linked PHP installation. You can access the PHP program output with a Web browser, viewing the PHP page through the server. For more information on PHP, see: http://www.php.net

How Apache/PHP and the ODBC driver work together in a Linux partition

The PHP module (if compiled with unixODBC support) provides built-in ODBC functions and can access DB2 UDB for iSeries using the iSeries ODBC driver for Linux. When the Apache Web server receives a PHP request from a client Web browser, the Apache server forwards the request to the PHP module. The PHP module then finds the source code of the requested PHP page and interprets and executes that source code.

If the PHP module finds a database access function in the source code, the PHP module invokes the iSeries ODBC driver to access DB2 UDB for iSeries and executes the related SQL statement. The iSeries ODBC driver returns the data result set to the PHP module. The PHP module then embeds the resulting data with HTML tags and other content, and forwards it to the Apache server. Apache then sends the HTML page to the client's Web browser. This process is shown in Figure 2-6.

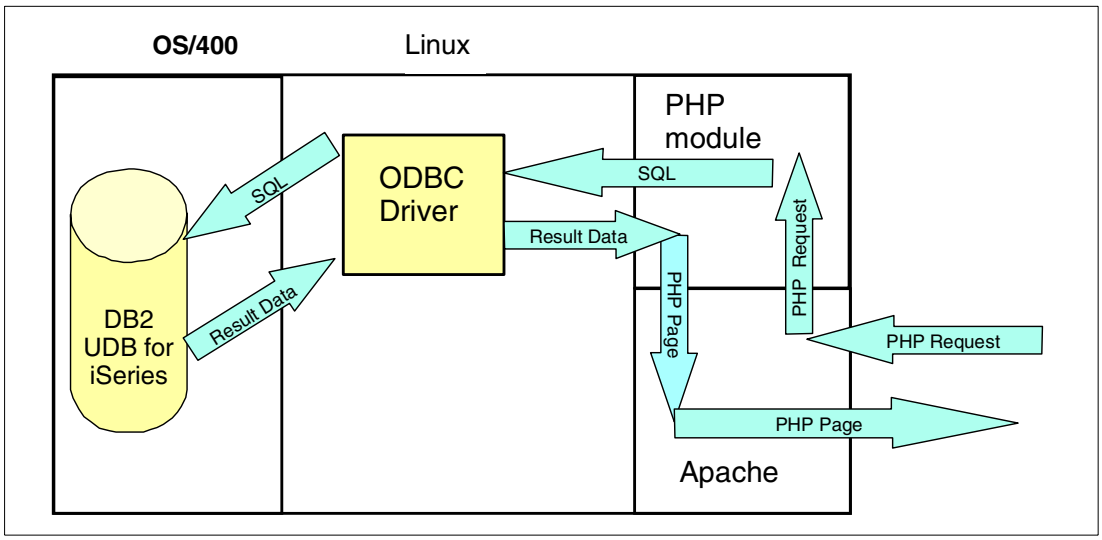

Figure 2-6 Apache and PHP with ODBC driver in a Linux partition

PHP support for unixODBC

The default installation of PHP does not support the unixODBC application. Therefore, PHP must be compiled from the source code with the option to enable unixODBC support. Manually compiling PHP should not be necessary with the next versions of TurboLinux and SuSE for the iSeries which will have unixODBC support built-in.

Note: Modules (like PHP) can be installed into Apache statically (during compile of Apache) or as a Dynamic Shared Object (DSO). There are some performance benefits using the static method with the tradeoff being ease of maintenance. You can find further details on the DSO-method versus the static method at: http://httpd.apache.org/docs/dso.html

The default installation of the Apache Web server on all current supported Linux distributions does not enable DSO support. Since pluggable modules (like the Tomcat Java servlet engine) require Apache DSO support to be available, Apache needs to be re-compiled from the source.

2.6.1 Installing Apache and PHP for use with ODBC

These instructions use the DSO method for the reasons listed earlier.

Software installation

1. Check to see if Apache is installed (it is by default on the Linux for iSeries distributions):

   ```
   # rpm -qa | grep apache
   ```

   If it lists an apache .rpm, it needs to be removed:

   ```
   # rpm -e apache_x.x.xx.rpm
   ```

2. Once it is uninstalled, download the Apache source from the following Web site to /usr/local:

   http://httpd.apache.org/

   Then unzip it as shown here:

   ```
   # gunzip < apache_1.3.23.tar.gz | tar xvf -
   ```

   Note that the version of Apache may differ from what we show.

3. Change to the directory you just created:

   ```
   # cd apache_1.3.23
   ```

4. Run the configure script with an option to enable DSO support. The prefix option determines where the root of the http server will reside:

   ```
   # ./configure --prefix=/www --enable-shared=max
   ```

5. Compile and install Apache:

   ```
   # make
   # make install
   ```

6. If there were no errors, you should verify that unixODBC is installed as well as the IBM iseries ODBC driver for Linux:

   ```
   # rpm -qa | grep ODBC
   ```

   If unixODBC or the iSeries ODBC driver are not installed, install them from the Linux distribution CDs with the command:

   ```
   # rpm -Uvh <file name>.rpm
   ```

7. Before you can compile the PHP source, you need to verify that either the **flex** or **lex** programs are installed. The default installation of SuSE for iSeries did not install these utilities. Run the following command:

   ```
   # rpm -qa | grep flex
   ```

   If nothing is returned, install the flex lexical analyzer from the distribution CDs (using **yast2** in SuSE SLES 7 for iSeries).

8. You need to verify that the unixODBC source code is available when PHP is compiled. We already verified that the unixODBC binary is installed in step 6. Copy unixODBC-2.x.x.tar.gz to /usr/local and run:

```
# gunzip < unixODBC-2.2.0.tar.gz | tar xvf -
```

9. Do the same for the PHP source code:

```
# gunzip < php-4.1.1.tar.gz | tar xvf -
```

Change directory to the PHP source directory:

```
# cd php-4.1.1
```

10. You are now ready to configure PHP. The following command assumes the Apache root directory is /www (this was set in step 3). Apxs stands for “APache eXtenSion”.

```
# ./configure --with-apxs=/www/bin/apxs --with-unixODBC=/usr/local/unixODBC-2.x.x
# make
# make install
```

Note: You must supply the path to the unixODBC source code so it can be used during compilation of PHP.

Software configuration

1. Configure some Apache options so that the PHP script interpreter will run when a Web client accesses your .php files. Run the following command:

```
# cp /usr/local/php-4.1.1/php.ini-dist /usr/local/lib/php.ini
```

Edit your httpd.conf file (Located in /www/conf) and make sure the PHP 4 mime type is there and uncommented. You need a line that looks like this:

```
AddType application/x-httpd-php .php
```

2. The php.ini file contains PHP configuration settings. Copy the php.ini-dist file to /usr/local/lib/php.ini:

```
# cp -f /usr/local/php-4.1.1/php.ini-dist /usr/local/lib/php.ini
```

3. Make your PHP application available to Apache. Usually, applications are in a directory off the Apache documents directory. See Figure 2-7 for a sample 3-tier ODBC application.

4. You are now ready to start the Apache Web server:

```
# /www/bin/apachectl start
```

5. Test the Web server and PHP support by creating a file accessible via Web client browsers:

```
# echo “<?   phpinfo()  ?>” > /www/htdocs/phpinfo.php
```

This creates a simple .php file that you can access from a Web browser to test the PHP installation. It contains information on how PHP was configured and how ODBC support was compiled.

6. Point your Web browser to:

```
http://<linux server name>/phpinfo.php
```

Sample PHP database application for accessing the iSeries

Figure 2-7 contains a sample 3-tier application that uses PHP and unixODBC to obtain data from iSeries DB2 UDB. Simply change the dbname, dbuser, and dbpwd values to match your system.

```
[root@sander /www/htdocs]# cat testODBC.php
<html>
<body>
<h1 align=center>Employee Name List</h1>
<table border=1 bgcolor='#7F8F9F' align=center>
<tr> <th>Customer Number</th>  <th>Last Name</th>  <th>Balance Due</th> </tr>
<?
$dbname="slh";                                          //unixODBC DSN name
$dbuser="marquis";                                      //OS400 username
$dbpwd="xxxxxxx";                                       //OS400 password

$db=odbc_connect($dbname,$dbuser,$dbpwd);               //connect to DB2 UDB for
iSeries

$sql="select CUSNUM, LSTNAM, BALDUE from qiws.qcustcdt";    //query Statement
$result=odbc_exec($db,$sql);                            //execute Query statement
While (odbc_fetch_row($result)) {                       //fetch result set
   printf("<tr><td>%s</td><td>%s</td><td>%s</td></tr>",
   odbc_result($result,1),odbc_result($result,2),odbc_result($result,3));
}
odbc_close($db);                                        //close connection
?>
</table>
</body>
</html>

[root@sander /www/htdocs]#
```

Figure 2-7 Sample 3-tier ODBC application

After you create the PHP script and save it in your Apache documents directory, you can access it via a client Web browser and see the results returned in your browser window.

2.7 Simple ODBC application written in C

testODBC.c is a working example of an application fetching an ordinary database file (QIWS/QCUSTCDT) from a backend OS/400 using the iSeries ODBC driver for Linux. It assumes that unixODBC is installed, the iSeries ODBC driver is installed (with any required PATH settings to reach it), and that the target iSeries database is registered as a data source in unixODBC.

The command to compile it is:

```
# gcc -o testODBC -lodbc testODBC.c
```

The *-o* switch outputs the “testODBC” executable. *-lodbc* links the ODBC libraries.

To run ODBC, use the following command:

```
# ./testODBC <unixODBC DSN name> <iSeries User id> <iSeries password>
```

Example 2-1 testODBC.c

```
/*
   Simple ODBC example for iSeries Linux or Linux generally.

   Pass in:   datasourcename ("name" in ODBCConfig System or User or File DSN)
              validUserID
```

```
              password

   Thus:   testODBC myiSeries myuserid mypassword

   LWL, March 2002
*/


#include <sql.h>
#include <stdio.h>
#include <stdlib.h>

/* Simple query to DB table very likely to exist.  Takes first OS/400 "member" */
char querystring[]="SELECT * FROM QIWS.QCUSTCDT";

/* dieXXX are to notify about termination and maybe do it */
void diehappy(char *reason) {
  printf("%s\n",reason);
}

void die(char *reason) {
  printf("Failure because: %s\n",reason);
  exit(1);
}

void die2(char *reason,SQLRETURN code) {
  printf("Error code: %d\n",code);
  die(reason);
}

int main(int argc, char *argv[]) {

#define MAX_CONNECTIONS 5

 char field1[255];   /* receives values from table row fetch */
 char field2[255];
 char field3[255];
 SQLINTEGER retfld1;
 /* For completelness. */
 SQLINTEGER retfld2; /* For this example, NULL for SQLBindCol */
 SQLINTEGER retfld3; /* would be enough    */

 SQLHENV henv;
 SQLHDBC hdbc[MAX_CONNECTIONS ]; /* For multiple connections */
 SQLHSTMT statement;
 SQLRETURN sqe=0;
 SQLRETURN res=0;

 /* allocate environment */
 if ( SQLAllocEnv(&henv)) die ("Alloc Environment");

 /* allocate connection */
 if ((sqe=SQLAllocConnect(henv,&hdbc[0]))) die2("Allocate Connect",sqe);

 /* connect to data base */
 if ( (sqe= SQLConnect(hdbc[0],argv[1],SQL_NTS,
                   argv[2],SQL_NTS,
                   argv[3],SQL_NTS))
  ) die2("Connect",sqe);
```

```
/* allocate statement(s) */
if (SQLAllocStmt(hdbc[0],&statement)) die("Alloc stmt");


/* actual processing */


/* Do query */
if ((sqe=SQLExecDirect(statement,querystring, SQL_NTS)))
    die2("exec direct",sqe);

/* Bind coming SQLFetch operations to fields */
if ((sqe=SQLBindCol(statement,1,SQL_CHAR,field1,255,&retfld1)))
  die2("Bind field 1",sqe);
if ((sqe=SQLBindCol(statement,2,SQL_CHAR,field2,255,&retfld2)))
  die2("Bind field 2",sqe);
if ((sqe=SQLBindCol(statement,3,SQL_CHAR,field3,255,&retfld3)))
  die2("Bind field 3",sqe);

/* While loop to fetch all records */
res = SQLFetch(statement);

/* field1,2,3 take new values each time*/
while (SQL_SUCCEEDED(res)) { /* Create tab separated values */
  printf("%s\t%s\t%s\n",field1,field2,field3);
  res= SQLFetch(statement);  /* next record */
}

/* end actual processing */

/* free statement(s) pretty completely  */
if (SQLFreeStmt(statement,SQL_DROP)) die("Free Statement with Drop");

/* free connection */
if ((sqe=SQLDisconnect(hdbc[0] ))) die2("Disconnect ",sqe);
if ((sqe=SQLFreeConnect(hdbc[0]))) die2("Free Connection",sqe);

/* free environment */
if (SQLFreeEnv(henv)) die("Free Environment");

diehappy("Got to end");

return 0;
}
```

Note: The unixODBC driver manager comes with a small sample application called “isql” that can also be used as a sample.

2.8 Troubleshooting

There are many troubleshooting and error logging tools available for use with the iSeries ODBC driver for Linux.

The most common way to start debugging possible communication problems with the ODBC driver is to use the "cwbping" tool shipped in the /opt/ibm/iSeriesODBC/bin directory. It displays detailed messages about any possible communication problem. The tool “cwbping” is usually the first thing to check when having communications problems with the driver.

The most common way to start debugging possible ODBC problems with the ODBC driver is to use the "isql" or "DataManager" tool shipped with unixODBC. Both of these simple ODBC applications can be used to run a quick test to make sure your data source is configured and running properly. These tools are usually the first thing to check when having ODBC problems with the driver.

Usually the order of debugging an ODBC driver problem is:

1. Look up the help text for the error message on the Web site:
   http://www-1.ibm.com/servers/eserver/iseries/linux/odbc/guide/

   Or you can look in /opt/ibm/iSeriesODBC/doc for CWBxxzzzz messages.
2. If there is any possibility of a communications-related problem, use **cwbping** to verify the connection to the iSeries.
3. Try iSql or DataManager to test the basic DSN configuration.
4. Examine the SQL.log file.
5. Use the tool “cwbtrc” to take a history log and detail trace of the driver. Then send it to IBM for analysis.

2.8.1 Error messages

When an error occurs, the iSeries Access ODBC Driver for Linux returns the SQLSTATE (an ODBC error code) and an error message. The driver obtains this information both from errors that are detected by the driver and from errors that are returned by the iSeries server.

For errors that occur in the data source, the iSeries Access ODBC Driver for Linux maps the returned native error to the appropriate SQLSTATE. When both the driver and the driver manager detect an error, they generate the appropriate SQLSTATE. The iSeries Access ODBC Driver for Linux returns an error message based on the message returned by the iSeries server.

For errors that are detected within the iSeries Access ODBC Driver for Linux, the driver returns an error message based on the text associated with the SQLSTATE. These error messages are translated messages. Help text for error messages found in the underlying components of the iSeries Access product is also shipped in the /opt/ibm/iSeriesODBC/doc directory.

Error message format

Error messages have the following format:

```
[vendor][ODBC-component][data-source]
error-message
```

The prefixes in brackets ([]) identify the source of the error. Table 2-2 shows the values of these prefixes returned by the iSeries ODBC Driver for Linux.

When the error occurs in the data source, the [vendor] and [ODBC-component] prefixes identify the vendor and name of the ODBC component that received the error from the data source.

Table 2-2 Error messages

Error source	Value
Driver Manager	[unixODBC] [Driver Manager]
iSeries Access ODBC Driver for Linux	[unixODBC] [IBM] [iSeries Access ODBC Driver]
NLS messages	[unixODBC] [IBM] [iSeries Access ODBC Driver] Column #: NLS error message number NLS error message text
Communication	[unixODBC] [IBM] [iSeries Access ODBC Driver] Communications link failure. comm rc=xxxx - (message text) *xxxx* is the error number in decimal, not hexadecimal, format. Message text describing the nature of your error appears with the error number.
DB2 UDB for iSeries	[unixODBC] [IBM] [iSeries Access ODBC Driver] [DB2 UDB] Server error message

To view error message text for DB2 UDB for iSeries errors, see Table 2-3.

Table 2-3 DB2 UDB for iSeries errors

For errors that begin with:	Use this OS/400 command
SQL	`DSPMSGD RANGE(SQLxxxx) MSGF(QSQLMSG)`
IWS or PWS	`DSPMSGD RANGE(ZZZxxxx) MSGF(QIWS/ QIWSMSG)` *ZZZ* is either IWS or PWS.

Table 2-4 displays other error prefixes that you may see through the iSeries Access ODBC Driver for Linux.

Table 2-4 Other error prefixes

Message Prefix	Message File	Description
CWB####	cwber.htm	Base error messages
CWBCO####	cwbcoer.htm	Communication error messages
CWBNL####	cwbnler.htm	Conversion error messages
CWBSY####	cwbsyer.htm	Security error messages

2.8.2 SQL.log

The ODBCConfig utility has the ability to trace calls to ODBC functions from an application. This ability is quite similar to Microsoft's SQL logging facility available in the MS ODBC Administrator. SQL logging for unixODBC can be enabled and disabled on the `Advanced` tab of the ODBCConfig GUI. See Figure 2-8. The name and location of the SQL.log file may also be changed.

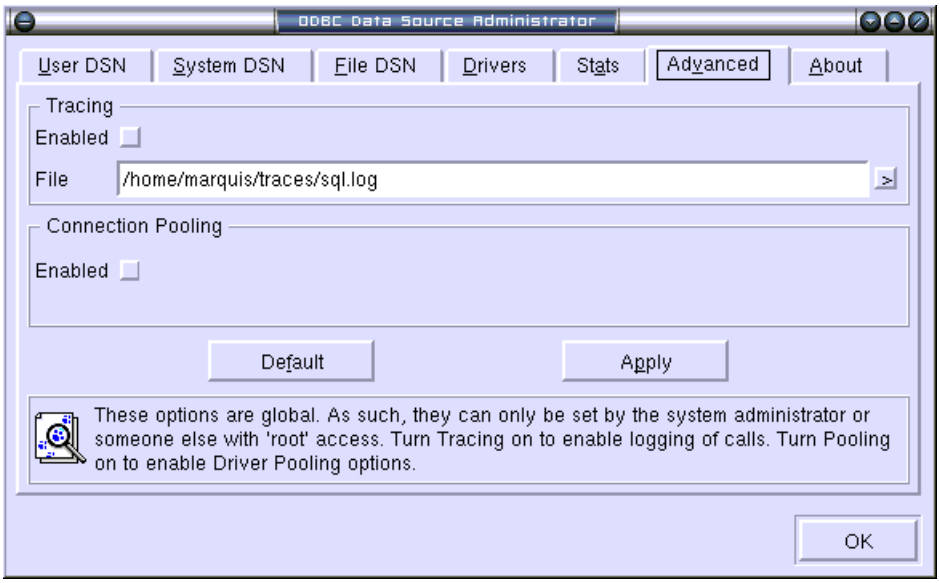

Figure 2-8 SQL.log

SQL.log files can be quite useful when troubleshooting ODBC problems and are often requested by IBM support. Be careful not to leave tracing enabled since the SQL.log file continues to grow! Also note that most ODBC applications need to be started after logging has begun. This allows the logging facility to start intercepting the ODBC calls that the application is making.

2.8.3 Tracing and history logging

The iSeries ODBC driver for Linux contains tracing and logging functions. To enable and configure tracing and logging, use:

```
/opt/ibm/iSeriesODBC/bin/cwbtrc
```

We recommend that you enable history logging to provide high-level details for many common error messages. By default, the traces are written to the /home/<user>/.iSeriesODBC directory. Second level help text for error messages in the history log are written in the /opt/ibm/iSeriesODBC/doc directory in HTML format.

2.8.4 QUSER

The host servers run under the QUSER user profile, so it must be enabled. From an iSeries command line, type:

```
DSPUSRPRF USRPRF(QUSER)
```

Press Enter to display the status for QUSER. Use the CHGUSRPRF command to change the profile if necessary.

2.9 National Language Support

To see the current locale character set to IBM CCSID mapping, use the command:

```
# /opt/ibm/iSeriesODBC/bin/cwbnltbl
```

You can also use this command to download new character conversion tables.

In C, use the setlocale function (for example, setlocale(LC_ALL,"ja_JP.eucjp")) and use nl_langinfo(CODESET) to retrieve the current locales character set. The iSeries ODBC driver will try to match the locale character set to a CCSID.

2.10 Additional references

For more information, refer to these Web sites:

- iSeries ODBC Driver for Linux: http://www.ibm.com/iseries/linux/odbc
- ODBC and the unixODBC Project: http://www.unixodbc.org
- Apache Software Foundation: http://www.apache.org
- PHP: http://www.php.net

3

AS/400 NetServer support for Samba

This chapter discusses the new support for Linux Samba client connectivity to AS/400 NetServer and gives practical examples of how it may be used in an iSeries/Linux environment.

This is not a comprehensive discussion of Samba since there is already a great deal of literature available on the subject. Rather, this chapter focuses on Samba as it relates to the AS/400 NetServer and how it can be used to access a Linux partition on the iSeries from Windows PCs.

This chapter discusses:

- The SMB protocol
- The new AS/400 NetServer Samba client support and how to use it
- Troubleshooting AS/400 NetServer Samba client support
- Using the iSeries for Linux Samba server from Windows PCs

3.1 Conventions

There are some conventions that we use when discussing Samba access to AS/400 NetServer. In Windows, a user *maps a network drive* to shared resources. But in Linux, a user *mounts* over existing directories. For example, you might map a network drive to the *I:* drive in Windows, whereas you would *mount* a remote AS/400 NetServer share over /mnt/myshare in Linux. The ability to use any directory as a local mount point lifts the restriction on how many *mapped drives* you can have on a Linux system since you are not limited to single drive letters.

Many Linux applications and utilities are available in both a graphical user interface (GUI) mode and a text-based mode. The standard Samba suite does not include any GUI utilities, but there have been many GUI and Web-based configuration tools released to the open-source community to ease configuration. This chapter uses the text-based tools only since they are standard across every installation. However, don't let that dissuade you from trying some of the great GUI Samba configuration tools available.

This chapter assumes that you have already correctly installed the Linux partition and you are using a name resolution technique to get the IP address of the AS/400 NetServer.

You also need the root user password for the Linux partition.

3.2 Introduction to the SMB protocol

SMB stands for Server Message Block. It is a file-sharing protocol that is in very wide use by Windows PCs. Generally, whenever a network drive is mapped from a Windows PC to another Windows PC, the SMB TCP/IP protocol is being used.

Samba is a project that implements the SMB/CIFS standard on UNIX-style operating systems enabling them to easily share files with any other SMB-enabled operating system, including AS/400 NetServer, an SMB server.

The name Samba was given by the project's creator, Andrew Tridgell. He simply searched for a dictionary word that contained the letters "S" "M" and "B" in a word in that order and came up with SaMBa.

Samba is key for Linux as it allows Linux PCs and servers to seamlessly interact with existing Windows PCs and file servers without requiring any additional software to be loaded on those machines. There continues to be constant development and enhancements made to the Samba server and client resulting in a very stable and fast file-sharing environment. AS/400 NetServer has supported Windows client network drive connections for some time and is well-used. Now, we extend support to Linux Samba clients.

3.3 AS/400 NetServer Samba client support

Starting with V5R1 of OS/400, Linux Samba client connectivity is supported to AS/400 NetServer. Therefore, Linux PCs can map network drives to AS/400 NetServer shares and move files back and forth just as Windows PCs do today. See Figure 3-1.

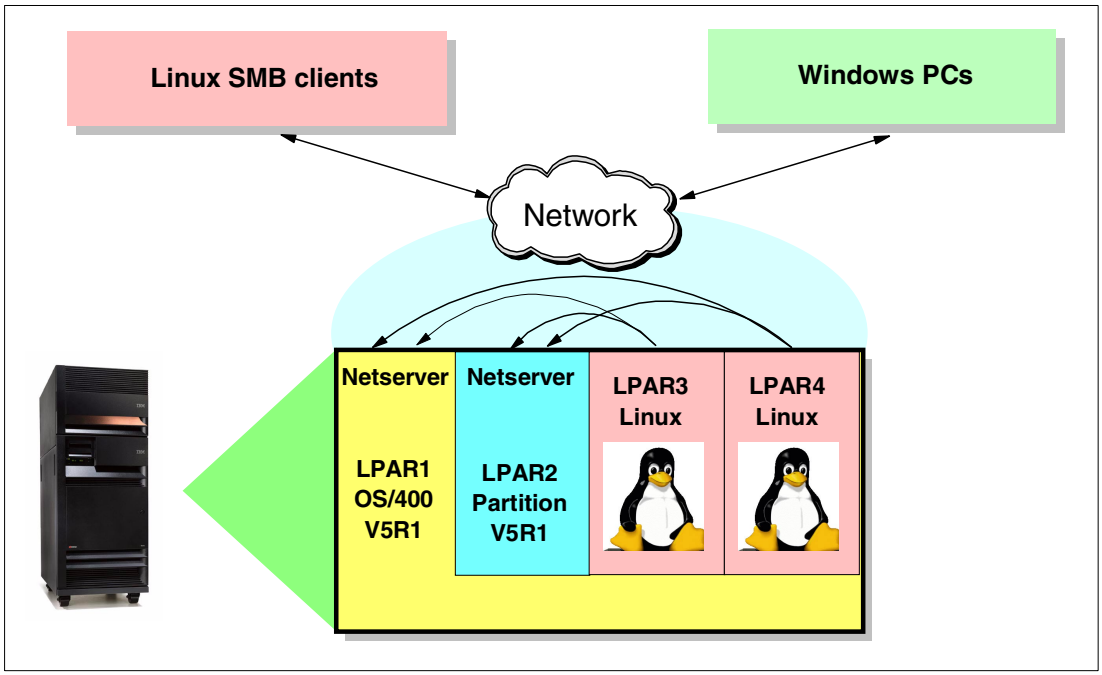

Figure 3-1 AS/400 NetServer Samba support

OS/400 has an SMB client in the form of the QNTC file system. It was designed solely for accessing Windows NT SMB servers and is not supported for use with Samba servers. Consider using NFS to access Linux servers from OS/400.

3.3.1 OS/400 requirements

V5R1 OS/400 requires the application of five PTFs to enable support for Linux Samba clients:

- MF27247 - General
- MF27248 - File
- MF27249 - Print
- MF27294 - Locking
- MF27295 - Security

We also recommend that you apply XPF SI02179 and all the PTFs concurrently.

The next release of OS/400 will have support for Linux Samba clients built-in.

3.3.2 Linux requirements

The following levels are required to be in a supported environment:

- Linux kernel version 2.4.4+
- Samba versions 2.0.7 or 2.2+
- Tested Distributions: TurboLinux, SuSE, and Red Hat

Testing with Red Hat 7.1 Kernel 2.4.2-2 has also been successful. All Linux for iSeries distributions ship with supported levels of both the Linux kernel and Samba.

To find information on the kernel and distribution you are running on a Linux machine, execute the command:

```
# cat /etc/issue
```

Samba installation

The three supported Linux distributions all use the Redhat Package Manager (RPM) system to manage installed software. Every program installed from binary code should have an entry in the RPM database on the system. For example, you can find what Samba components are installed on a Linux system with the command:

```
# rpm -qa | egrep "samba|smb"
samba-common-2.2.1a-4
samba-2.2.1a-4
samba-client-2.2.1a-4
```

If nothing is returned, you need to install at least the Samba client before mounting an AS/400 NetServer share. Some distributions package Samba into three separate rpms as shown above, but others package both the Samba client and server in the same rpm. The easiest way to install Samba is to install Samba during the installation of Linux. If that was not done, it is a simple task to install it using RPM tools.

The Samba .rpm files are part of the Linux for iSeries distributions, so you can use the distribution software installation tools to install Samba:

```
RedHat - gnorpm (GUI only)
SuSE - yast2 (GUI or text-based)
Turbolinux - TurboPkg (text-based)
```

If you are not familiar with these tools, you may find the Samba .rpm packages on the distribution CDs and install them manually using the commands:

```
# rpm -ivh samba.rpm
# rpm -ivh samba-client.rpm
```

Notes:

- The Samba file names may be different from the names listed above depending on the Linux distribution you are using and the release level of Samba. The SuSE SLES 7 CD contains samba.rpm and samba-client.rpm
- “Dependencies” are prerequisite programs. If you get dependency errors indicating that packages are missing when running “rpm -ivh samba.rpm”, you have to install those packages first.
- For details on the rpm command, use:

  ```
  # man rpm
  ```

- To uninstall a package, use:

  ```
  # rpm -e <package>
  ```

3.4 Using AS/400 NetServer Samba support

This section provides an introduction to use AS/400 NetServer support for Linux Samba clients. It assumes you’ve met the requirements listed in 3.3, “AS/400 NetServer Samba client support” on page 28.

AS/400 NetServer is a large topic and is covered in the redbook *The AS/400 NetServer Advantage*, SG24-5196. Up-to-date details on AS/400 NetServer and its support for Linux Samba clients are also available at: http://www.ibm.com/eserver/iseriest/netserver

3.4.1 AS/400 NetServer configuration

As with Microsoft Windows SMB clients, Linux SMB clients must have an AS/400 NetServer share available on the iSeries available to connect to. This is accomplished via Client Access Operations Navigator by following these steps:

1. Open Operations Navigator.
2. Expand the sections **<iSeries system name>-> File Systems-> Integrated File System-> Root-> home**. See Figure 3-2.

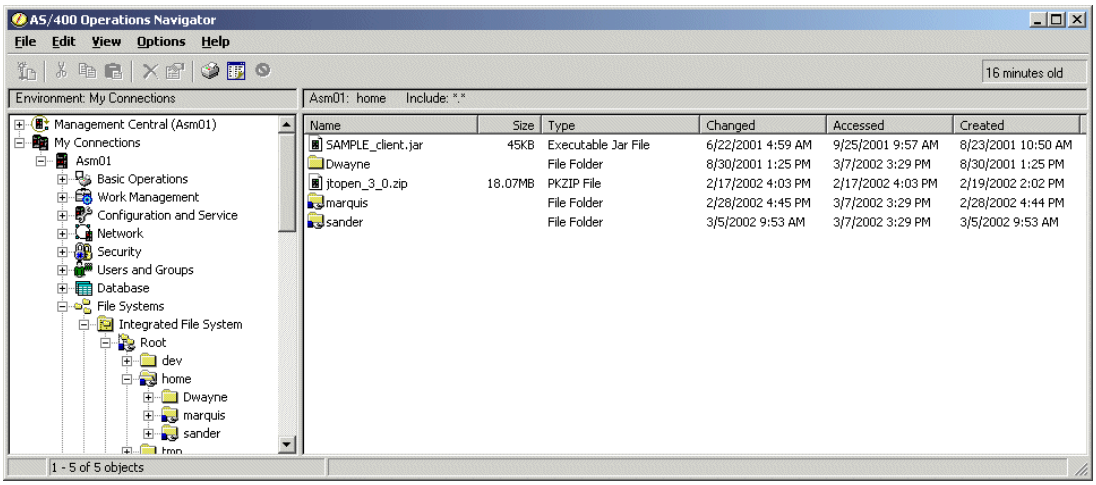

Figure 3-2 Operations Navigator

3. Create a new directory under the home directory and make it available for AS/400 NetServer clients.

   Right-click **home** and select **New Folder**.
4. Name the folder and click **OK** (for the purposes of this walkthrough, the folder is named *sander*). This folder appears in the right-hand window (Figure 3-2).
5. Right-click the folder and select **Sharing**-> **New Share**.
6. Change the access to **Read/Write**. You may also provide a description for the AS/400 NetServer share. Click **OK**.

> **Note:** The small hand appears beneath the folder. This indicates that the directory has been shared.

Once you complete these steps, you are ready to access the share from a Samba client.

3.4.2 Linux Samba client configuration

There are two methods to access SMB resources from a Linux Samba client:

- smbmount
- smbclient

smbmount

`smbmount` is the equivalent of "mapping a drive" in Windows. Instead of drive letters as with Windows, directories are mounted over. These directories are called *mount points*. Any directory you have authorization to can be mounted over – even if that directory is not empty!

Executing the **smbmount** command starts a **smbmount** daemon process that manages the mount point. While the **smbmount** daemon runs, the mount point is active.

To mount arbitrary file systems, you must either be signed in as the root user or the **smbmount** and **smbumount** comands must allow non-root user execution. See the **smbmount** command details in the section below.

The type of file system mounted by smbmount is called **smbfs**.

smbclient

smbclient is an FTP-like interface that can be used to access remote SMB shares. It is generally used when a user only requires a short connection to a remote share or wants to see a list of the available shares. It is also more mature and provides faster data transfer rates than smbmount. This is likely to change as smbmount updates are brought into the latest Linux kernel versions.

Using smbmount to map a network drive

Since we just created a new folder and shared it with Operations Navigator, we are now ready to mount to that remote AS/400 NetServer share. First, we need a local mount point on the Linux partition. Most distributions have a directory off the root ('/') named /mnt. Generally, its purpose is to place mount directories within it. A good place to create a mount point would be within /mnt:

1. Run the following command:

   ```
   # mkdir /mnt/smbtest
   ```

   The directory name does not need to match the share name on the iSeries.

2. If you are signed on as root, you can mount the remote share:

   ```
   # smbmount //<systemname>/sander /mnt/smbtest -o username=marquis
   ```

   Use a valid iSeries user profile name. You are prompted for a password. Enter a valid iSeries password. You may now treat the mounted directory as through it were a local file system.

3. To test it, run:

   ```
   # ls /mnt/smbtest
   ```

smbmount command details

The first argument of the smbmount command is the remote share being accessed in the form of //servername/sharename. The servername must be an IP address or be resolvable to an IP address (via DNS, /etc/hosts, or WINS-style lookup such as /etc/samba/lmhosts). The second argument is the local mount point and can be located anywhere. The *-o* switch indicates "options" and precedes the options to be sent to the mount daemon. In this case, the only option we are sending is the iSeries user profile that we want to sign on with. Other options include: password; fmask; dmask; guest; ro; etc. For additional **smbmount** command options, execute:

```
# smbmount --help
```

The **smbmount**-related commands are:

- smbmount
- smbumount

Generally, they are located in /usr/bin and owned by the root user. The **smbmount** command is actually a wrapper around the **smbmnt** command, so **smbmnt** must also be accessible by the user attempting the mount. Since these commands access the smbfs file system, they can only be run by root or *as* root using the Switch User ID (**suid**) file option. This means that if a regular user wants to use smbmount, smbmount must allow regular users to temporarily "switch" to root to run the command. To enable smbmount and smbumount to be run as the root user by any user on the system, use the **chmod** command, for example:

```
# chmod +s /usr/bin/smbmount /usr/bin/smbumount
```

Various Samba client add-on tools, such as xSMBrowser or LinNeighborhood, provide GUI alternatives to use the **smbmount** command.

smbumount

smbumount is the equivalent to unmapping a network drive in Windows. It unmounts a remote share from the local file system. The only parameter for this command is the local mount point that was mounted over. Using the example above, the command would be:

```
# smbumount /mnt/smbtest
```

This command fails if the actual connection no longer exists. In that case, the root user has to execute the more general **umount** command.

Also, TurboLinux distributions have a problem with the **smbumount** command. Even if /usr/sbin/smbumount and /usr/sbin/smbmnt have the suid bit turned on (-rwsr-sr-x), non-root users receive an error when attempting to unmount an smbfs mount point that they mounted. The result is that only the root user can unmount smbfs-type file systems on TurboLinux systems.

Using smbclient to copy a file from AS/400 NetServer to Linux

As mentioned earlier, smbclient is an FTP-like utility that can be used to copy files to and from an SMB share on a remote system. Follow these steps to use the utility to copy a file from AS/400 NetServer to Linux:

1. Verify what shares are available on the remote system:

   ```
   # smbclient -L //<systemname>
   ```

   This provides you with a list of shares that are available on the remote SMB server (AS/400 NetServer) as shown in Figure 3-3.

```
[marquis@sandr ~]$ smbclient -L //m01
added interface ip=10.5.213.95 bcast=10.5.213.255 nmask=255.255.255.0
session request to M01 failed (Called name not present)
Password:
Domain=[] OS=[S40VRM] Server=[]

        Sharename      Type      Comment
        ---------      ----      -------
        QIBM           Disk      IBM Product Directories
        ADMIN$         IPC       AS400 Resource
        PHILA          Disk
        QCA400         Disk      CA/400
        QNTAP          Disk      NT Service Directory
        QDIRSRV        Disk      OS/400 -- Directory Services
        SANDER         Disk
        PRTNP17        Printer   ITSO MR network printer
        TOMCAT         Disk
        QSYS LIB       Disk
        ASPRT          Printer   Device created for M01.
        HOME           Disk      Home directories
        IBMLAN$        IPC       AS400 Resource
        QNTC           Disk      Share for Int xSeries Server for iSeries
        IPC$           IPC       AS400 Resource
        MARQUIS        Disk      Share creation walkthrough
        ROOT           Disk
        LARRYPRT       Printer
        DUNNL          Disk      Test Samba share

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
[marquis@sandr ~]$
```

Figure 3-3 smbclient listing of shares

2. Establish an smbclient connection to the AS/400 NetServer:

   `# smbclient //<systemname>/<sharename> -U <valid iSeries user profile>`

   Enter your password when you are prompted. You then see an smb prompt:

   `smb: \>`

3. Type `help`

   You are presented with the list of available commands. Notice they are very similar to FTP.

4. Use `ls` to see a list of available files in the share you have connected to.

5. Use **`get`** or **`put`** to move a file from or to the iSeries. To copy an entire directory, you must turn on directory recursion and disable prompting (which would prompt for every file you want to copy).

6. To do this, execute:

   `smb: \> recurse`

   Then run:

   `smb: \> prompt`

You may now use the **mget** and **mput** commands to copy directories to and from the AS/400 NetServer share.

3.5 Troubleshooting SMB connectivity to AS/400 NetServer

Here is a list of problem instances that can occur:

- `4436: session request to <sysname> failed (Called name not present)`

  This error message occurs when the system name or IP address used to make the smbmount connection is different from the AS/400 NetServer name (usually 'Q'<serial #>). It can be ignored. smbmount and smbclient only require the IP address of the remote system or the ability to resolve the system name to an IP address.

- `cannot mount on /<mountpoint>: Operation not permitted`

  This error message occurs because the user executing smbmount does not have authority to the mount point.

- If you get "Input/output error" when trying to do anything with a mount point or when you try to unmount (smbumount) the mount point, the SMB connection has probably been lost. This can be caused by the forced ending of the QZLSFILE job that was managing the connection, a restart of the AS/400 NetServer server, a network issue, etc. To get the Linux system to know the mount point has been lost, the root user needs to execute:

  ```
  # umount /<mountpoint>
  ```

- Can't unmount a file system because it is "in use".

  A running process that you are unaware of may be "using" the file system you are trying to unmount. The first thing to check is whether your current working directory is inside the smbfs file system. If you did a 'cd' into it, then it will be in use. To find users of a file system, Linux provides the very helpful command, **fuser** (usually located in /sbin). See Figure 3-4.

```
[root@sandr /mnt/m01]# /sbin/fuser -mauv /mnt/m01

                     USER        PID ACCESS COMMAND
/mnt/m01             root       3374 ..c..  bash
[root@sandr /mnt/m01]#
```

Figure 3-4 /sbin/fuser command

You could then use the following command to end that process:

```
# kill 3374
```

Note: To kill all the PIDs accessing the file system, add the -k option to the **fuser** command. Use it with caution!

- `4403: tree connect failed: ERRSRV - ERRinvnetname (Invalid network name in tree connect.)`

  This error is received when an invalid remote share is specified on the **smbmount** command. Run:

  ```
  # smbclient -L //<systemame> -U <valid iSeries user profile>
  ```

  This generates a list of shares available on the SMB server specified.

smbfs permissions

You may notice that when doing an extended listing of an SMB share (`ls -l`), the permissions and owner of all objects are the same (Figure 3-5). The `rwx` (read, write, execute) and ownership properties are determined by the smbfs file system when **`smbmount`** is executed. It is possible to change the ownership and permission bits, but the change is for every object in the mount point.

```
marquis@SuSE641:/mnt> ls -l smbtest
total 33099
-rwxr-xr-x    1 marquis  users      130944 Dec 20 15:00 db2sql.sav
drwxr-xr-x    1 marquis  users        4096 Dec 21 12:34 delme
-rwxr-xr-x    1 marquis  users        1317 Jan 23 17:01 dleme.txt
-rwxr-xr-x    1 marquis  users        1476 Jan 23 17:07 listR
-rwxr-xr-x    1 marquis  users          95 Jan 23 17:17 listls
-rwxr-xr-x    1 marquis  users         105 Jan 23 17:17 lists.txt
-rwxr-xr-x    1 marquis  users          14 Dec 20 14:31 newbug
-rwxr-xr-x    1 marquis  users         104 May 31  2001 newimpf
-rwxr-xr-x    1 marquis  users          99 May 31  2001 newimpf.txt
drwxr-xr-x    1 marquis  users        4096 Oct 22 17:01 smblinux1
drwxr-xr-x    1 marquis  users        4096 Sep 13 12:13 smblinux2
-rwxr-xr-x    1 marquis  users    33743992 Dec  7 11:13 support
marquis@SuSE641:~/slh>
```

Figure 3-5 Directory permissions

Note that this does not change anything on the remote share. Only the local Linux system enforces these permissions. iSeries permissions always use object-level security independent of the security mechanisms on the Linux system.

iSeries-side troubleshooting

AS/400 NetServer allocates a QZLSFILE job in the QSERVER subsystem for each mount point mounted from the Linux system to AS/400 NetServer. This is in contrast to Windows clients that can double-up multiple network drives on a single QZLSFILE session. To see what QZLSFILE jobs are allocated to particular users, you can use the OS/400 command:

```
WRKOBJLCK OBJ(MARQUIS) OBJTYPE(*USRPRF)
```

Figure 3-6 shows a list of all jobs in use by user MARQUIS. If we work with the QZLSFILE job, we can get information such as the IP address of the remote SMB client and other typical OS/400 job information.

```
                          Work with Object Locks
                                                              System:   M01
Object:   MARQUIS        Library:   QSYS          Type:   *USRPRF

Type options, press Enter.
  4=End job   5=Work with job   8=Work with job locks

Opt   Job          User         Lock      Status          Scope     Thread
      ALEXITSO     MARQUIS      *SHRRD     HELD           *JOB
                                *SHRRD     HELD           *JOB
                                *SHRRD     HELD           *JOB
      QPWFSERVSO   QUSER        *SHRRD     HELD           *JOB
                                *SHRRD     HELD           *JOB
                                *SHRRD     HELD           *JOB
                                *SHRRD     HELD           *THREAD   00000004
      QZLSFILE     QUSER        *SHRRD     HELD           *JOB
                                *SHRRD     HELD           *JOB
                                *SHRRD     HELD           *JOB
                                *SHRRD     HELD           *THREAD   00000009
      QZRCSRVS     QUSER        *SHRRD     HELD           *JOB
                                *SHRRD     HELD           *JOB
                                                                    More...
F3=Exit   F5=Refresh   F12=Cancel
```

Figure 3-6 Object locks for user MARQUIS

Similar information is available in the Operations Navigator AS/400 NetServer configuration:

Open Operations Navigator and expand the sections ***<iSeries system name>*-> Network-> Servers-> TCP/IP**. Double-click **AS/400 NetServer**.

Here you can gather a great deal of information regarding AS/400 NetServer shares and all the active connections to those shares. Figure 3-7 displays the properties for an active session.

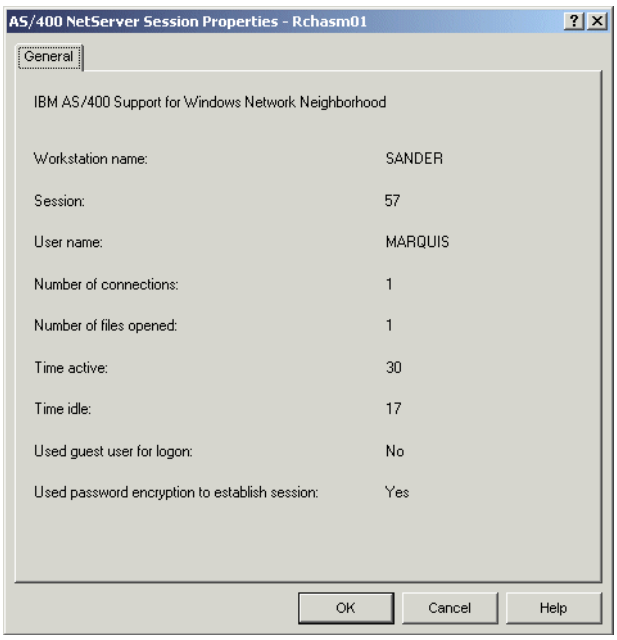

Figure 3-7 AS/400 NetServer Session Properties

3.6 Setting up the iSeries for Linux Samba server from PCs

Once Linux is installed on a partition of the iSeries, there may be instances where you want to move files from Windows PCs to the Linux partition or use the Linux partition as a file server. This section provides a simple walkthrough on setting up a network drive from a Windows PC to a Linux Samba server.

3.6.1 Linux setup

First you need to make sure that the Samba daemon programs are running (**smbd** and **nmbd**). **nmbd** is the "NetBIOS name server to provide NetBIOS over IP naming services to clients". This is important for browsing and host name lookup from Windows SMB clients, but, for the purposes of this walkthrough, is not necessary to be running.

smbd, on the other hand, is the daemon that manages incoming SMB client requests. Verify it is running using the command:

```
# ps -eH | grep smbd
```

If it is not listed, execute:

```
# /usr/sbin/smbd
```

Check to see that it is now started with the **ps** command above. This is the "dirty" way to start Samba. Generally, it is a good idea to use your Linux distributions' Sysvinit level editor to set the **smbd** daemon to always start on boot of the Linux partition. It is also possible to start it immediately from the editor. The command to access this GUI editor (on Red Hat systems) is **serviceconf**. SuSE systems can access it from the path **yast2-> System-> Runlevel Editor-> Runlevel Properties**. See your distribution documentation for more information on starting Samba.

Creating a Linux user ID

Next you need to have a user ID on the Linux system. You can use the command **/usr/sbin/useradd** to quickly create Linux user accounts:

```
# /usr/sbin/useradd baila -m
```

This creates user "baila". The "-m" at the end of the above command creates the directory /home/baila. Note that this use does not have a Linux system password unless you assign one with the **passwd <user id>** command. For the purpose of this demonstration, this is not necessary.

Setting up the Linux user ID for Samba

Microsoft Windows SMB clients use encrypted passwords for session authentication. This is a tricky issue for SMB servers since it requires the storage of an encrypted password in a text file on the Linux system. The default setup of a Samba server is to allow encrypted passwords (A manual change to the smb.conf file used to be necessary). For a user to sign on with an encrypted password, the root user must first create the /etc/samba/smbpasswd password file. This is done using the command:

```
# smbpasswd -a baila
```

It prompts for a password and then prompts you to verify that password.

The *-a* switch adds an entry for user *baila* to the /etc/samba/smbpasswd file and creates that file if it does not already exist. The **smbpasswd** command without any options allows any user running the command to change their Samba server password (but the root user must first have created the smbpasswd file since root must own it).

Note that the password stored in the /etc/samba/smbpasswd file *can* be different from the password that the user uses to sign onto the Linux system. To prevent confusion, it may be a good idea to keep the user's main Linux password the same as the Samba password.

SMB server shares in a Samba server

By default, all Samba configuration files are stored in the directory /etc/samba. The main configuration file is smb.conf. This file is mainly for Samba server configuration but does contain some parameters that are relevant when using the Samba client to connect to another SMB server. Since we are interested in mapping a network drive from Windows to the Linux Samba server, we need to have a share defined to connect to. The default installation of Samba already provides a share definition in the smb.conf file:

```
[homes]
        comment = Home Directories
        read only = No
        create mask = 0640
        directory mask = 0750
        browseable = No
```

This share allows an SMB client to access the home directory of whatever user ID they signed onto the SMB server with. In our case, when user "baila" connects to the Linux SMB server with the correct user ID and password, they see their home directory.

The various options that can be set in an smb.conf file for the purpose of creating Samba server shares is a large topic and beyond the scope of this book. But since we already have a good sample share, we are ready to map a network drive.

3.6.2 PC setup

Different Microsoft Windows operating systems behave differently as SMB clients. Pre-Win98se versions (without the Winsock2 update from MS) require the remote SMB servers' IP address be resolved via the hosts file, DNS server or a WINS method. Win98se and later versions allow mapping network drives directly to the IP address of an SMB server. Since there are a wide variety of IP address resolution strategies in use, we use the simple IP address method. If you are running Windows 95, you need to have a resolution strategy in place and use a system name instead of an IP address when searching for an SMB server (or install the Winsock2 update).

Another difference in MS Windows Operating Systems is that only the Windows NT-based operating systems (WinNT, Win2000, WinXP) allow you to specify a user ID when making the AS/400 NetServer connection. This can be quite inconvenient since, if the user ID you've signed onto a Windows 98 PC with does not exist on the iSeries or Linux server, you cannot provide a valid user ID to the SMB server unless you log off the PC and log back on with a valid user id. Non-NT-based operating systems are only capable of prompting you for a different password. NT-based operating systems are much easier to deal with in this respect since they allow you to send whatever user ID and password you'd like to the SMB server. For the purpose of simplicity, let's assume you are using a Windows NT-based operating system.

From the desktop, right-click **Network Neighborhood** (NT) or **My Network Places** (Win2000/XP) and select **Map Network Drive...**

In the **Folder:** area, enter:

```
\\<IP address of iSeries>\baila
```

Click the option to **Connect using a different user name**. We need to do this because if the user ID/password used to sign on to the Windows PC is not a valid smbuser on the Linux SMB server, you are signed on as the guest user. Using the default smb.conf, signing on as the guest user only gives you access to the *printers* folder.

Give the user ID “baila” and whatever password you assigned when you ran:

```
smbpasswd -a baila
```

Click **OK** and then **Finish**.

You now see a view of the /home/baila directory on the Linux partition. Try copying some large files into it.

3.7 Sharing an iSeries printer using Samba

Linux has its own printer and spooling support through **lpr** and **lpd**. However, it may be beneficial for Linux users to send files requiring printing to an OS/400 configured printer, therefore using OS/400 and its spooling and printer management capabilities as the print server.

3.7.1 Configuring a printer share

The following example shows you how to configure a printer share enabling users on a Linux partition to share an OS/400 configured IBM Infoprint 1130.

Note: A similar configuration can be used for any Hewlett Packard Printer Control Language (HP PCL) or PostScript-compliant printer.

On OS/400, the printer may be set up as a TCP/IP Remote Output Queue (RMTOUTQ) or a 3812 *LAN DEVD using SNMP. For information on configuring a printer on OS/400, see *IBM AS/400 Printing V,* SG24-2160.

Restriction: Due to the limited printer driver support at present on Linux if you intend to set up a printer share to an OS/400 printer configured as *IPDS, Advanced Function Printing (AFP) *YES, then you need to use the IBM Infoprint Server for iSeries (5722-IP1) product to convert a PCL or Postscript data stream to AFP.

For further information, see *IBM iSeries Printing VI: Delivering the Output of e-business SG24-62500.*

In the example, we will use the **kde** Linux graphical user interface (GUI) from a Linux partition running SuSE to both set up the Samba printer share and from which to print a Linux file. If you are using a different Linux GUI the steps and screen displays may be different.

Initially select a suitable OUTQ on OS/400 or create one using the **CRTOUTQ** command.

From Operations Navigator share the OS/400 OUTQ

To share the OS/400 OUTQ perform the following steps from the Operations Navigator interface:

1. Select **Network**, **Servers**, **TCP/IP** and **AS/400 NetServer**.
2. Expand **AS/400 NetServer** and right-click **Shared Objects**. Select **New** and then **Printer**.

3. In the following screen (see Figure 3-8) fill in the **Share name** and **Output queue** and **library** fields.

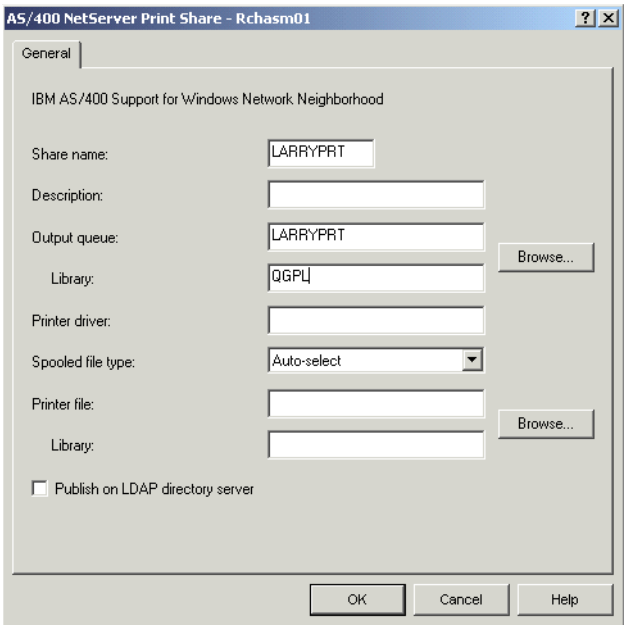

Figure 3-8 AS/400 NetServer Print Share screen

4. Select **OK**.
5. By selecting **Shared Objects** check that this printer share appears in the list.

Configure Samba printer from Linux

Now that an OS/400 OUTQ has been shared a Samba printer can be configured from the Linux partition.

1. From the KDE screen select the **Start Application** (or 'Kicker'), followed by **System**, **Configuration** and **Printer**. See Figure 3-9.

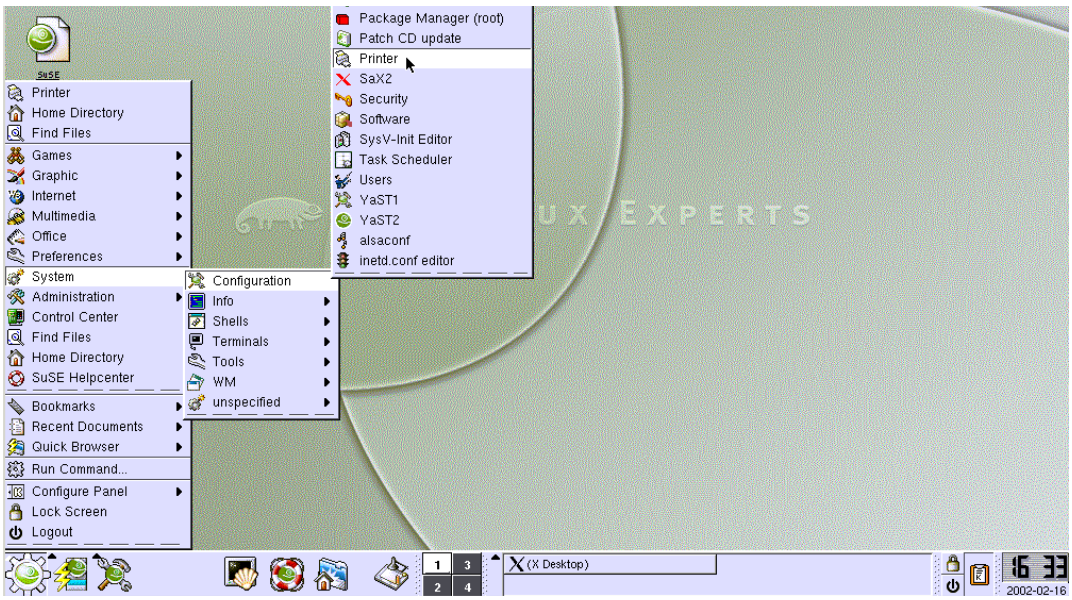

Figure 3-9 Selecting Printer from KDE

2. A *Run as root* screen appears if you are not signed on as the root user. Root privileges are required to set up a printer. Therefore, if this screen appears, enter the root user's password.
3. YaST2 then initializes the printer configuration tool. When the printer setup window appears, select **Configure**.
4. At the Connection for printer window, select **Samba/Windows printer** (see Figure 3-10) and then select **Next**.

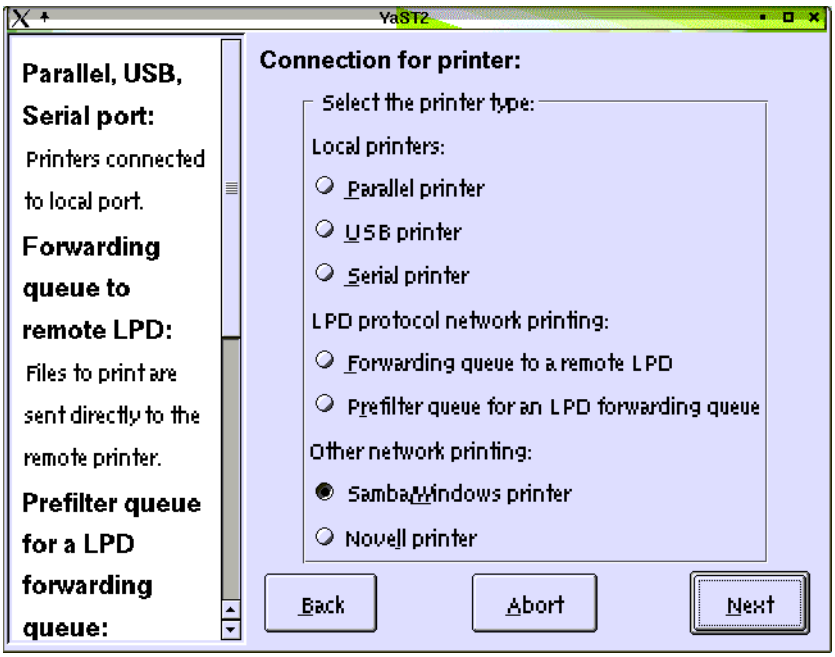

Figure 3-10 Connection for printer window

5. The Samba/Windows printer screen appears (Figure 3-11). Enter the following information and select **Next**:
   - **Host name**: Enter the host name or IP address for the OS/400 partition.
   - **Remote queue**: Define the OS/400 OUTQ name.
   - **User**: Enter a valid OS/400 user ID.
   - **Password**: Enter the password for this user.

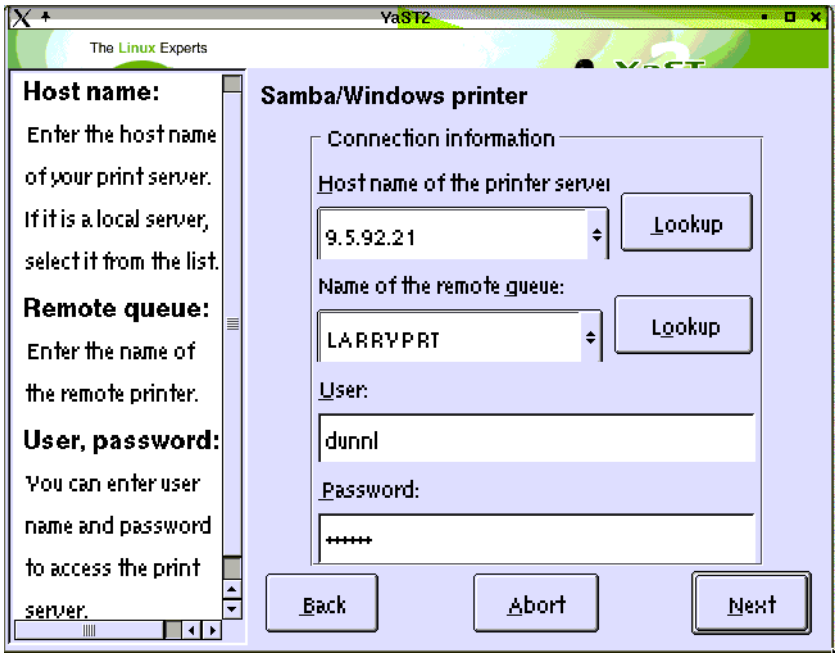

Figure 3-11 Samba/Windows printer window

6. From the Manufacturer and model of printer window, select the type of OS/400 printer on which you want the Linux files to eventually print on. Then select **Next**. This is similar to selecting a traditional printer driver from within a Microsoft Windows environment.

   Depending on the installed features of an Infoprint 1130 (or other printer model), you may have various options. If only the PCL engine is installed, try selecting **Generic printer** from the *Select manufacturer* list and **PCL 5e** from the *Select model* list. If the printer also has a PostScript engine then try the **PostScript level 2** from Select model. See Figure 3-12.

   The value selected in the Select model determines the exact ASCII printer data stream into which a Linux file is converted and then sent to the OS/400 OUTQ.

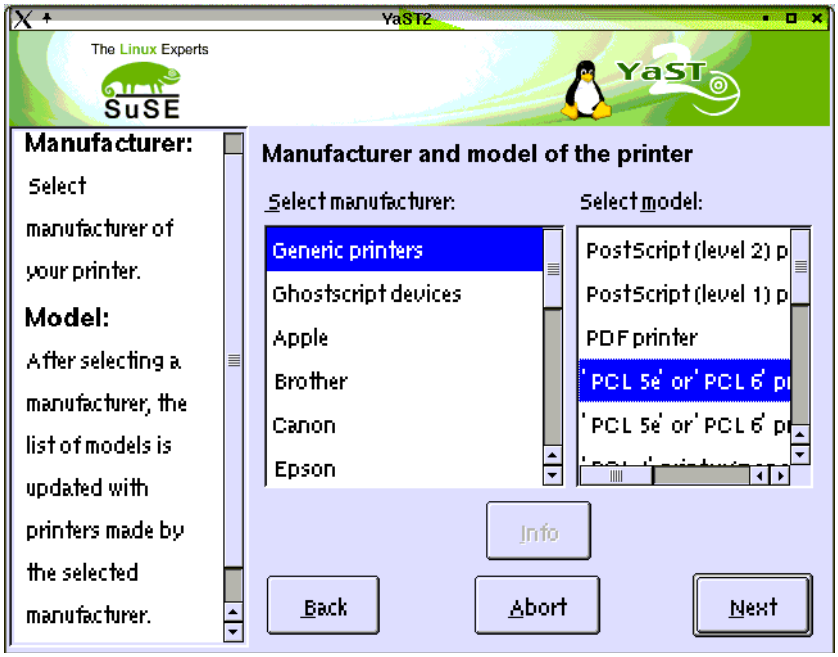

Figure 3-12 Manufacturer and model of printer window

7. On the next screen, Printer's name, give the printer a suitable name then select **Next**.
8. From the Printer settings window, you need to experiment with the settings. For example, if you are using A4 sized paper in the printer, select **a4** for Paper size. This prevents LOAD LETTER appearing on the printer's LED. Then click **Next**. See Figure 3-13.

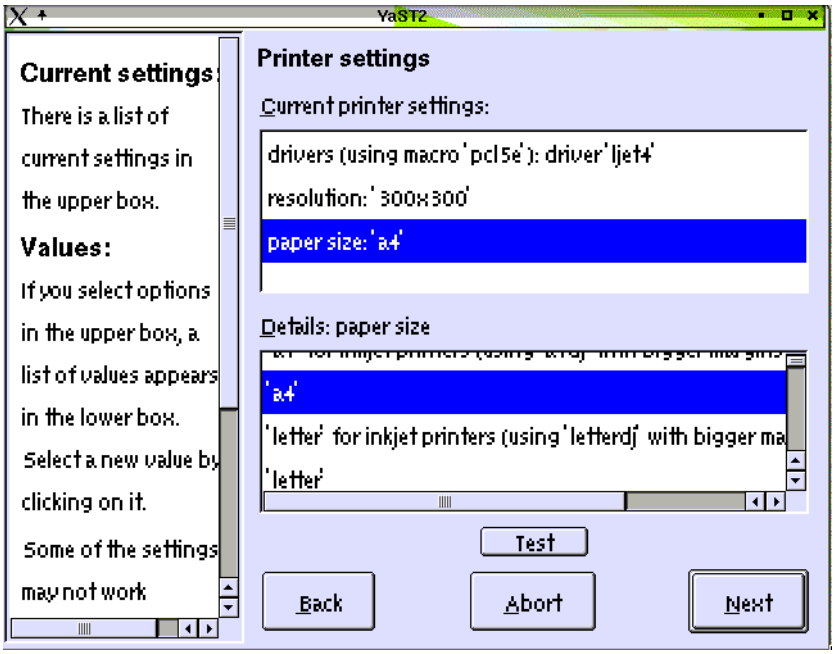

Figure 3-13 Printer settings window

9. On the Hardware-independent settings window, you can select other values, such as duplex, whether to print a cover sheet, or to force landscape orientation. Then select **Next**.

10. From the final window, select **Finish**. In SuSE, this updates the /etc/printcap file (see Figure 3-14). Using a GUI interface to configure a printer saves you having to edit this file manually.

```
### BEGIN apsfilter: ### /etc/gs.upp/y2prn_larryprt.upp larryprt ###
#   Warning: Configured for apsfilter, do not edit the labels!
#            apsfilter setup Sat Feb 16 12:31:37 CST 2002
#
larryprt-ascii|lp1|y2prn_larryprt.upp--ascii-larryprt|y2prn_larryprt.upp ascii:\
        :lp=/dev/null:\
        :sd=/var/spool/lpd/y2prn_larryprt.upp--ascii-larryprt:\
        :lf=/var/spool/lpd/y2prn_larryprt.upp--ascii-larryprt/log:\
        :af=/var/spool/lpd/y2prn_larryprt.upp--ascii-larryprt/acct:\
        :if=/var/lib/apsfilter/bin/y2prn_larryprt.upp--ascii-larryprt:\
        :la@:mx#0:\
        :tr=:cl:sh:
#
larryprt|lp2|y2prn_larryprt.upp--auto-larryprt|y2prn_larryprt.upp auto:\
        :lp=/dev/null:\
        :sd=/var/spool/lpd/y2prn_larryprt.upp--auto-larryprt:\
        :lf=/var/spool/lpd/y2prn_larryprt.upp--auto-larryprt/log:\
        :af=/var/spool/lpd/y2prn_larryprt.upp--auto-larryprt/acct:\
        :if=/var/lib/apsfilter/bin/y2prn_larryprt.upp--auto-larryprt:\
        :la@:mx#0:\
        :tr=:cl:sh:
#
larryprt-raw|lp3|y2prn_larryprt.upp--raw-larryprt|y2prn_larryprt.upp raw:\
"printcap" 32L, 1348C                                         1,1           Top
```

Figure 3-14 /etc/printcap file after configuring a Samba printer share from KDE

3.7.2 Printing to the Samba printer

Printing to the Samba printer from SuSE Linux can be accomplished in two ways:

Note: There may be slight differences in the other Linux distributions.

- From the KDE graphical user interface
- From the Linux command line using the **lpr** command

In both cases, make sure that the **lpd** is started. This can be achieved by the following command:

```
# /etc/rc.d/lpd start
```

This should result in a line similar to this:

```
Starting lpd                                                    done
```

Printing from the KDE graphical user interface

1. Select **Start Application** and select **Home Directory**. Open the file that you want to print.
2. From the menu bar, select **Location** and then **Print**. Ensure **Name** is defined with the correct printer and that the **Properties** values are correct. Click **OK**. See Figure 3-15.

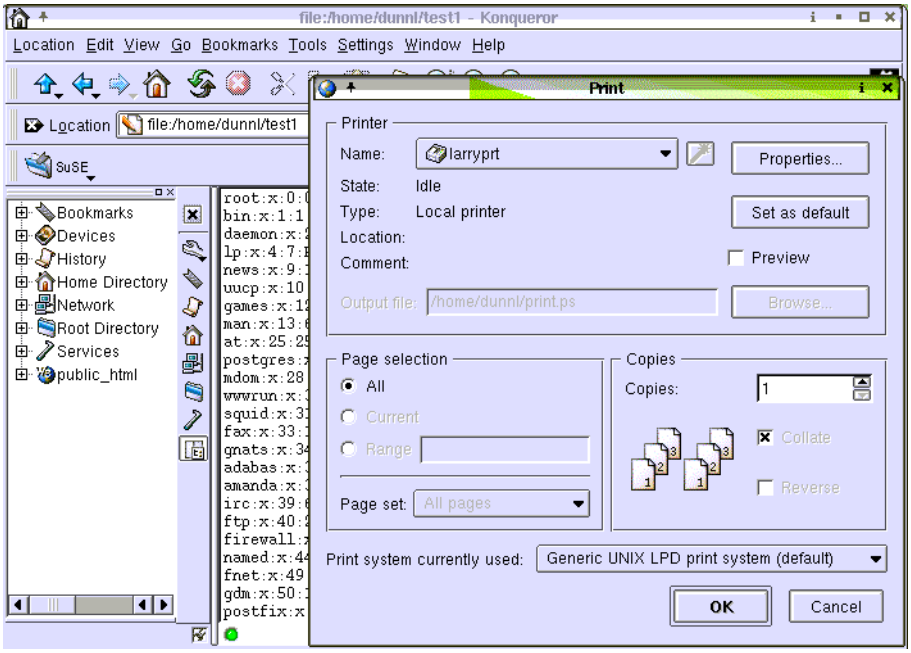

Figure 3-15 Printer selection from KDE

Printing from the Linux command line

For this example, the **lpr** command was used. The **lp** command could be used, but it requires the user to have similar authorities as the **root** user:

```
# lpr -P larryprt-ascii test1
```

To find the files on OS/400, type:

```
WRKOUTQ <outq_name>
```

The Work with Output display (Figure 3-16) should be displayed. You will not be able to display the spooled files (SPLFs) from a PC5250 emulation screen since they have already been converted to ASCII. The Printer Device Type is *USRASCII.

```
                          Work with Output Queue

Queue:    LARRYPRT       Library:    QGPL           Status:    RLS

Type options, press Enter.
  1=Send    2=Change    3=Hold    4=Delete    5=Display    6=Release    7=Messages
  8=Attributes         9=Work with printing status

Opt  File         User         User Data    Sts    Pages    Copies  Form Type    Pty
     QPZLSPRTF    DUNNL                     RDY        1*       1    *STD          5
     QPZLSPRTF    DUNNL                     RDY        1*       1    *STD          5
     QPZLSPRTF    DUNNL                     RDY        1*       1    *STD          5
     QPZLSPRTF    DUNNL                     RDY        1*       1    *STD          5




                                                                              Bottom
Parameters for options 1, 2, 3 or command
===>
F3=Exit    F11=View 2    F12=Cancel    F20=Writers    F22=Printers
F24=More keys
```

Figure 3-16 WRKOUTQ showing Linux SPLFs on OS/400 OUTQ

3.7.3 Points to keep in mind

There are some limitations and restrictions that you may need to consider when setting up a Samba printer share for many Linux users:

- In order that the USER column of the OS/400 WRKOUTQ <outq_name> display reflects SPLF ownership correctly, ensure that each Linux user *also* has a valid OS/400 user profile and password. Create a separate Samba printer session for each user. In the Samba/Windows printer setup display, define the particular user profile name and password in the User and Password fields respectively.
- SPLFs may be moved from one OUTQ to another, or a different printer writer can be started against the initial OUTQ. However, if you do this, ensure that the physical printer, which will eventually be used to print the file, has an ASCII printer engine reflecting the same ASCII data stream type supported by the Linux printer driver. A *USERASCII SPLF, once on an OS/400 OUTQ, cannot be changed. For example, a *USERASCII SPLF built with the HP PCL 5 ASCII data stream should print successfully on any PCL 5 compliant printer. However, it would print incorrectly if sent to a printer supporting the IBM PPDS data stream.
- Some printer functions, such as correct drawer selection, staple or finisher support may not work with the generic drivers. However, there are Linux printer drivers for various IBM printers. See the following Web site for more details:
  http://www.printers.ibm.com/R5PSC.NSF/Web/driver

For further information on Linux printing in general, the following HOWTOs are useful:

- http://www.linuxdoc.org/HOWTO/Printing-HOWTO/index.html
- http://www.linuxdoc.org/HOWTO/Printing-Usage-HOWTO.html

4

Linux integration with JDBC

This chapter discusses how Linux can be integrated with OS/400 using JDBC. It covers the following topics:

- Introduction to JDBC
- Preparing to use JDBC
  - Downloading the Toolbox for Java
  - Creating the sample database
- Command line JDBC application
- Web serving with JDBC
  - Apache and Tomcat on Linux
  - Apache and Tomcat on OS/400
- Graphical JDBC application

4.1 Introduction to JDBC

The Java Database Connectivity (JDBC) API was designed to make coding common SQL statements in Java simple and easy. At the same time, more complex SQL statements are possible. The typical steps to using the JDBC API are:

1. Load the JDBC driver.
2. Create the JDBC connection.
3. Create and execute your SQL statements.
4. Close the JDBC connection.

The JDBC API is part of the standard Java Development Kit (JDK). Version 1.0 of JDBC runs on JDK 1.1.x and above. Version 2.0 of JDBC runs on JDK 1.2 and above.

4.2 Preparing to use JDBC

As you can see from the previous section, the first step in using JDBC is to load the JDBC driver. That leads to the question of which driver to use and where to get it.

For UDB for iSeries there are two different JDBC drivers that can be used. These two drivers are called the *native* driver and the *toolbox* driver. The native driver is shipped as part of licensed program Java Developer Kit (57xx-JV1). The toolbox driver is part of licensed program IBM Toolbox for Java (57xxJC1).

A simple comparison of the two drivers is that the native driver is faster but runs only on the OS/400 JVM while the toolbox driver runs on any JVM. A good discussion about these drivers can be found at: http://www.as400.ibm.com/developer/jdbc/Faqs/JDBCFAQ.html

We used the toolbox driver in our examples.

4.2.1 Downloading the Java Toolbox

The IBM Toolbox for Java is installed as product 57xx-JC1. Once installed, the Toolbox jar files are found in IFS in directory /QIBM/ProdData/HTTP/Public/jt400/lib. You can find information about each of the jar files and the classes included at http://www.ibm.com/servers/eserver/iseries/toolbox/ or in the downloadable html help files. To use the Toolbox classes, you need to download the appropriate jar file to your client and update the CLASSPATH to include the jar files. If you use FTP to download the jar files, make sure to use binary mode. See 5.3, “FTP” on page 69, for more information on FTP.

For JDBC and the simple command line JDBC application included here, only the base jt400.jar file was needed. See 5.5, “Toolbox for Java” on page 79, for additional examples of using the Toolbox.

4.2.2 Creating the sample database

All of the examples in this chapter use a sample database. The creation of the sample database was done using the stored procedure CREATE_SQL_SAMPLE(‘*library*’), where *library* is the name of the sample library you want to create. This stored procedure is new in V5R1 and shipped with the operating system. To create the sample database, we used these steps:

1. To get into interactive SQL, enter:

```
STRSQL
```

2. To create the library (or collection in SQL terminology) with the sample tables:

```
CALL QSYS/CREATE_SQL_SAMPLE('ITSOSAMPLE')
```

Creating the sample database takes a couple of minutes to complete. Once it is completed, you have 13 tables and numerous logical views. Sample data is also included in the tables.

There are two sets of database tables in the sample schema. One set includes tables ORG, STAFF, and SALES. This set is a very simple schema with limited interaction between the tables and less complex data types within the tables.

The remaining ten tables make up a more complicated database schema and use more complex data types such as BLOBS and CLOBS. These tables also interact with each other much more. You can find more information about the sample database schema in Appendix A of the *DB2 UDB for iSeries SQL Programming Concepts* manual, which is available at:
http://publib.boulder.ibm.com/pubs/html/as400/infocenter.html

Once you reach this site, follow these steps:

1. Select **V5R1** and the language of your choice and then click **GO!**
2. Select **Site map**.
3. Scroll down and select **DB2 UDB for iSeries**.
4. From *Quick Access Paths*, *DB2 manuals*, select **SQL Programming Concepts**.

4.3 Command line JDBC application

To start JDBC programming on Linux, you can start with a command line approach where all the output is sent to standard out and all input is from standard input. This is a great way to test your JDBC and SQL code without the worries of the graphical interfaces. Our sample command line JDBC application selects all the records out of the ORG table to get the list of departments. It then takes the department number and runs a prepared SQL statement to get the members of each department, their position, and the number of years with the company. This information is sent to standard out for each department, producing a departmental type listing.

To create the Java program on a Linux system, we entered:

```
# javac SimpleJDBC.java
```

This created the executable Java code which we ran using the following command:

```
# java SimpleJDBC
```

You may have to add the current directory to your CLASSPATH to get the Java program to run. To do this issue, the following command:

```
# export CLASSPATH=$CLASSPATH:.
```

This adds the current directory (identified by the '.') to the end of the current classpath. The output of this Java program is sent to standard out. If you want to send the data to a file, you can redirect standard out to a file using:

```
# java SimpleJDBC > jdbc.output
```

A bit more about the program

When making the JDBC connection, we embedded the user profile and password that will be used during the connection. If a user profile and password are not provided, a prompt appears asking for profile and password. When we did this on our Linux partition, some font errors were reported from Java, but the prompt appeared and worked.

The program flow is to execute a query to identify all the departments from the ORG table. For each resulting department, a prepared statement query is run to get all the members of the department from the STAFF table. The SQL statements could be simple as in the example or could be very complex with multiple joins and anything else you would need.

Simple JDBC code

Example 4-1 shows the JDBC code using the ORG, STAFF, and SALES tables.

Example 4-1 Simple JDBC code

```
import java.sql.*;

public class SimpleJDBC
{

    //! format a string to a set length
    private static String format(String s, int width)
    {
        String formattedString;

        if (s.length() < width) {
            StringBuffer buffer = new StringBuffer(s);
            for (int i = s.length(); i < width; ++i) {
                buffer.append(" ");
            }
            formattedString = buffer.toString();
        } else {
            formattedString = s.substring(0, width);
        }

        return formattedString;
    }

    public static void main(String argc[])
    {
        Connection connection = null;

        try {
            //! load the Toolbox JDBC driver
            Class.forName("com.ibm.as400.access.AS400JDBCDriver");

            //! get the connection to the iSeries database
            //! if no user or password is given a prompt will appear
            String system = "itsosys";
            String url = "jdbc:as400://" + system;
            String user = "ITSO";
            String password = "ITSOPWD";
            connection = DriverManager.getConnection(url, user, password);
            DatabaseMetaData meta = connection.getMetaData();

            //! create and execute a simple SQL select statement
            Statement orgSelect = connection.createStatement();
            ResultSet orgResults = orgSelect.executeQuery("SELECT * FROM ITSOSAMPLE"
                                                          + meta.getCatalogSeparator()
                                                          + "ORG");

            PreparedStatement deptSelect = connection.prepareStatement(
                                       "SELECT NAME, YEARS, JOB FROM ITSOSAMPLE"
                                       + meta.getCatalogSeparator()
                                       + "STAFF WHERE DEPT = ?");
```

```
            ResultSet deptResults = null;
            int deptNumb = 0;

            //! output info from each record
            while (orgResults.next()) {
                deptNumb = orgResults.getInt("DEPTNUMB");

                System.out.println("Department listing for "
                                   + orgResults.getString("DEPTNAME")
                                   + " in division "
                                   + orgResults.getString("DIVISION")
                                   + " located in "
                                   + orgResults.getString("LOCATION"));

                deptSelect.setInt(1, deptNumb);
                deptResults = deptSelect.executeQuery();

                while (deptResults.next()) {
                    System.out.println("  " + format(deptResults.getString("NAME"), 15)
                                       + " " + format(deptResults.getString("JOB"), 7)
                                       + " " + deptResults.getInt("YEARS"));
                }

            System.out.println();
            }
        }

        catch(Exception e) {
            System.out.println();
            System.out.println("Caught error " + e.getMessage());
            System.out.println(e);
        }

        finally {
            //! close the connection
            try {
                if (connection != null) {
                    connection.close();
                }
            }

            catch(SQLException e) {
            //! ignore since we are exiting anyway
            }
        }
    }
}
```

Sample output

Example 4-2 shows a sample of the output generated by the simple JDBC example.

Example 4-2 Sample simple JDBC output

```
Department listing for Plains in division Midwest located in Dallas
  Fraye           Mgr     6
  Williams        Sales   6
  Smith           Sales   7
  Lundquist       Clerk   3
  Wheeler         Clerk   6
```

```
Department listing for Pacific in division Western located in San Francisco
  Lea             Mgr     9
  Wilson          Sales   9
  Graham          Sales   13
  Gonzales        Sales   4
  Burke           Clerk   1

Department listing for Mountain in division Western located in Denver
  Quill           Mgr     10
  Davis           Sales   5
  Edwards         Sales   7
  Gafney          Clerk   5
```

4.4 Web serving with JDBC

One of the most common uses for JDBC is to access backend databases from within a servlet running inside a web page. In this way, a Web application can access existing data without having to redeploy the data to a different format. The database server can either be on the same system as the HTTP server for a 2-tier architecture or the database server can be on another system for a 3-tiered architecture. The other tier in this architecture is the client machine running a Web browser.

We set up a 3-tiered application as part of the demonstration program. The same environment exists on both OS/400 and Linux along with a sample servlet that uses JDBC to access an OS/400 database schema. The environment is the Apache HTTP server with the Jakarta Tomcat Web Application Server. Apache is the world's most popular HTTP server and is produced by the Apache Software Foundation as an open-source product. Tomcat is the Apache plug-in for doing dynamic Web serving and is produced as an Apache project.

We discuss the general steps used for installing and configuring Apache and Tomcat on both Linux and OS/400. For more detailed information and installation instructions, go to the Apache home page at: http://www.apache.org

4.4.1 Apache and Tomcat on Linux

Follow these general steps to load Apache onto Linux:

1. Download the Apache code from:

   http://www.apache.org

   This normally comes as a tarball. We downloaded the apache_1.3.22.tar.gz file.
2. Unzip and untar the tarball (file extension of .tar.gz) using the **tar -xvzf** command.
3. Run the configuration script to configure the server.
4. Run **make** followed by **make install** to compile the server and install it.
5. Start the HTTPD server.

At this point, you should have the Apache HTTP daemon active on your system.

To create and serve dynamic Web pages, you also want a Web application server such as the Jakarta Tomcat server. The Web application server allows you to have JSPs and servlets to generate and display dynamic web pages. The general steps to load Tomcat are:

1. Install the proper JDK for Tomcat. Since Tomcat is Java-based, you need to make sure the correct JDK is installed. The preferred JDK is IBMJava2-SDK for PowerPC architecture. This JVM should be provided with the Linux distribution CDs that you have.
2. Install the JTOpen package. This is also known as IBM Java Toolbox and should be included in the Linux distribution CDs. This is used in our demo JDBC Web application. If you will not use the Java Toolbox functions in your applications, you do not need this step.
3. Download the correct version of the Tomcat code from:

   `http://jakarta.apache.org/builds/jakarta-tomcat/release/v3.2.4/bin/`

   Note that we did not download the latest Tomcat release because we encountered some problems getting the newer version working with Apache. We downloaded the `jakarta-tomcat-3.2.4.tar.gz` file.
4. Unzip and untar the downloaded tarball (file extension of .tar.gz) using the **`tar -xvzf`** command.
5. Change the mode of the Tomcat shell to executable and start Tomcat.

At this point, the Tomcat server should start. Depending on the amount of processor and memory, this may take a few minutes.

The last step is to get Apache and Tomcat working together. This involves downloading, installing, and configuring the Apache plug-in module for Tomcat. The general steps for installing this adapter are:

1. Download the Apache-Tomcat module from:

   `http://jakarta.apache.org/builds/jakarta-tomcat/release/v3.2.4/src/`

   We downloaded the `jakarta-tomcat-3.2.4-src.tar.gz` file.
2. Untar and unzip the tarball (file extension .tar.gz) using the **`tar -xvzf`** command.
3. Change directory to the source directory just created and then to the appropriate Apache subdirectory depending on your level of Apache. For us, that was the `apache1.3` subdirectory.
4. Set the JAVA_HOME environment variable to the JDK path on your system. This is done using the command:

   ```
   export JAVA_HOME=/usr/lib/IBMJava2-ppc-13/
   ```

   Here, */usr/lib/IBMJava2-ppc-13* should be replaced with the location of the JDK on your system.
5. Compile the Apache module using the **`apxs`** script and then copy the created module to the `APACHE_HOME/libexec` directory.
6. Add the following line to your *APACHE_HOME/conf/httpd.conf* file inside the “<IfModule mod_alias.c>”:

   ```
   include TOMCAT_HOME/conf/mod_JK.conf-auto
   ```

   Here, *TOMCAT_HOME* is your base Tomcat directory.

That should be all that you need to run Apache and Tomcat on a Linux partition. If you have any problems or want to explore further, a good resource is:
`http://jakarta.apache.org/tomcat/tomcat-3.2-doc/tomcat-apache-howto.html`

Creating a dynamic Web page using JDBC

Now that we have Apache and Tomcat configured and running, you need a Web application to use the features available. Our sample Web application uses JDBC to create a simple database query on the sample department database tables. The JDBC code is similar to the command line JDBC example. First the driver is loaded, then the connection is made, then the SQL statements are created and executed, and finally the connection is closed.

The first display you see is a login screen asking for the system to connect to, the schema to read from, and the user and password to sign on as. Once connected and signed on, you can query the employee, department, projects, and activities tables. Figure 4-1 shows the initial page after you log in.

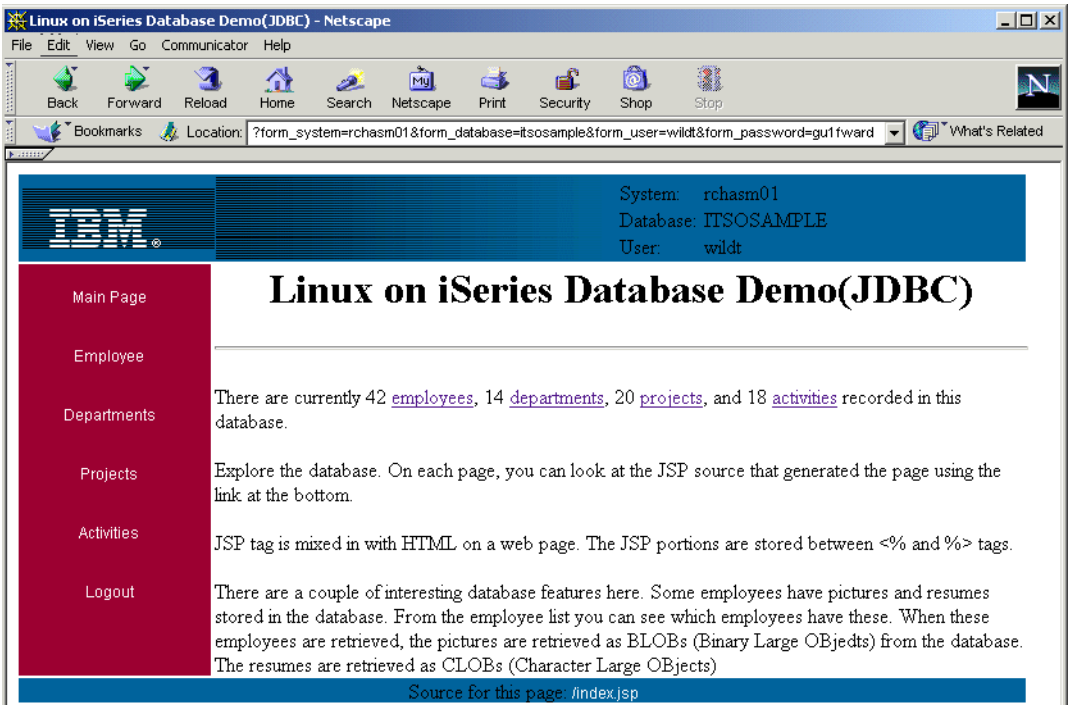

Figure 4-1 Web serving with JDBC from Linux

When you click either the left hand frame or the embedded links, you then see the list of employees, departments, projects, or activities. From there you can browse various links.

For example, you can select *employees* to see the list of employees. From there, you can select employee *Adamson, Bruce* to see all the information about Bruce Adamson. From Bruce's employee page, you can see which department he works in, who his manager is, and some personnel information.

If you select W L ROBOT DESIGN as one of the projects that Bruce is working on, you see the W L ROBOT DESIGN project page. From the W L ROBOT DESIGN project page, you can see all the activities that make up this project and all the employees working on this project.

If you select DESCRIBE LOGIC, you see information about the DESCRIBE LOGIC activity such as all the projects using this activity and all the employees performing this activity. You can access any of this information from a starting point. This makes for a nice interface since one user will start with an activity or project and another user will start with an employee or department. Both users can access all the information that they want without having to be trained on the order used to get the information.

4.4.2 Apache and Tomcat on OS/400

The Apache HTTP server and Tomcat Web server can also be installed, configured, and run on OS/400. This gives you the same setup as the Linux partition.

To install, configure, and setup both an Apache server and the Jakarta Tomcat adapter, you simply need to install the IBM HTTP Server for iSeries licensed program (57xxDG1). This licensed program contains two different HTTP servers. One is the original IBM HTTP Server that has existed for several releases. The other is the IBM HTTP Server (powered by Apache). This is a port of the Apache HTTP server running on OS/400.

Included in the HTTP Server (powered by Apache) is the Jakarta Tomcat web server or servlet engine. For more information about configuring the IBM HTTP Server (powered by Apache) and the included Tomcat servlet engine, see the iSeries Information Center Web site at: http://publib.boulder.ibm.com/pubs/html/as400/infocenter.html

Once you reach this site, follow these steps:

1. Select **V5R1** and the language of your choice and then click **GO!**
2. Select **Site map**.
3. Scroll down to *e-business and web serving* and select **Web serving**.
4. Select **HTTP Server (powered by Apache)**.

This page gives you an overview of the Apache HTTP server and links to such topics as configuration, management, and migration. The *Program* link takes you to additional help on programming to this HTTP server including information on JSPs and servlets. The *Java servlets and JSPs* link off the programming page takes you to a discussion on the implementation of the Jakarta Tomcat servlet engine on OS/400.

Creating a dynamic web page using JDBC

The strength of Java and its portability are shown in the fact that the same web application that runs on Linux runs on OS/400 with no changes. Review the steps in the demonstration guide for downloading the same JDBC example to the iSeries system. Then use the IBM HTTP Server (powered by Apache) with the Jakarta Tomcat Web server adapter to run the same browser based application to view employees, departments, projects, and activities.

4.5 Graphical JDBC application

JDBC can also be used in non-browser applications to access data from OS/400 and present that data in a graphical format. The following GUI program uses the same sample database as the command line JDBC application only presenting the data to the user graphically and allowing for interaction with the user.

A bit more about the program

The first thing the user is presented with is a drop-down or combo box from which to select the department they want information. By default, the first entry is selected and is blank. When this entry is selected, the table and all data is cleared out. Otherwise when a department is selected from the drop-down box, the server is queried and the information about that department is filled in. The division in which the department is located and the location of the department go in the text field above the table. Then each employee, their position, and their number of years with the company go in the table.

Graphical JDBC code

Example 4-3 shows the graphical JDBC code that was originally developed using VisualAge for Java from IBM. VisualAge for Java is an Integrated Development Environment (IDE) that provides a nice method of generating the graphical components and the non-graphical supporting code.

Example 4-3 Graphical JDBC example

```
import javax.swing.*;
import javax.swing.table.*;
import java.util.*;
import java.sql.*;

/**
 * Insert the type's description here.
 * Creation date: (3/2/2002 3:29:11 PM)
 * @author: WCS Developer
 */
public class GraphicalJDBC extends JDialog {
    private JPanel ivjJDialogContentPane = null;
    private JTextField ivjTFSelection = null;
    private JComboBox ivjCBDepartmentName = null;
    private JTextField ivjTFDepartmentLocation = null;
    private JScrollPane ivjSPTableScrollPane = null;
    private JTable ivjTBDepartmentTable = null;

    private Vector departments = new Vector();
    Connection connection = null;
    DatabaseMetaData meta = null;

    IvjEventHandler ivjEventHandler = new IvjEventHandler();

    class IvjEventHandler implements java.awt.event.ActionListener {
        public void actionPerformed(java.awt.event.ActionEvent e) {
            if (e.getSource() == GraphicalJDBC.this.getCBDepartmentName())
                connEtoC1(e);
        };
    };

    class DepartmentInfo extends Object{
        String name;
        String division;
        String location;
        int number;

        public DepartmentInfo() {
            super();
        }

        public String getName() {
            return name;
        }
        public String getDivision() {
            return division;
        }
        public String getLocation() {
            return location;
        }
        public int getNumber() {
            return number;
```

```
        }
        public void setName(String name) {
            this.name = name;
        }
        public void setDivision(String division) {
            this.division = division;
        }
        public void setLocation(String location) {
            this.location = location;
        }
        public void setNumber(int number) {
            this.number = number;
        }
    }

/**
 * GraphicalJDBC constructor comment.
 */
public GraphicalJDBC() {
    super();
    initialize();
}

/**
 * Return the JDialogContentPane property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJDialogContentPane() {
    if (ivjJDialogContentPane == null) {
        try {
            ivjJDialogContentPane = new javax.swing.JPanel();
            ivjJDialogContentPane.setName("JDialogContentPane");
            ivjJDialogContentPane.setLayout(null);
            getJDialogContentPane().add(getTFSelection(), getTFSelection().getName());
            getJDialogContentPane().add(getCBDepartmentName(),
getCBDepartmentName().getName());
            getJDialogContentPane().add(getTFDepartmentLocation(),
getTFDepartmentLocation().getName());
            getJDialogContentPane().add(getSPTableScrollPane(),
getSPTableScrollPane().getName());
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjJDialogContentPane;
}

/**
 * Return the TFSelection property value.
 * @return javax.swing.JTextField
 */
private javax.swing.JTextField getTFSelection() {
    if (ivjTFSelection == null) {
        try {
            ivjTFSelection = new javax.swing.JTextField();
            ivjTFSelection.setName("Selection");
```

```
            ivjTFSelection.setText("Select the department");
            ivjTFSelection.setBounds(16, 35, 189, 23);
            ivjTFSelection.setEditable(false);
            ivjTFSelection.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjTFSelection;
}

/**
 * Return the CBDepartmentName property value.
 * @return javax.swing.JComboBox
 */
private javax.swing.JComboBox getCBDepartmentName() {
    if (ivjCBDepartmentName == null) {
        try {
            ivjCBDepartmentName = new javax.swing.JComboBox();
            ivjCBDepartmentName.setName("CBDepartmentName");
            ivjCBDepartmentName.setBounds(221, 35, 189, 23);
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjCBDepartmentName;
}

/**
 * Return the TFDepartmentLocation property value.
 * @return javax.swing.JTextField
 */
private javax.swing.JTextField getTFDepartmentLocation() {
    if (ivjTFDepartmentLocation == null) {
        try {
            ivjTFDepartmentLocation = new javax.swing.JTextField();
            ivjTFDepartmentLocation.setName("TFDepartmentLocation");
            ivjTFDepartmentLocation.setBackground(new java.awt.Color(204,204,204));
            ivjTFDepartmentLocation.setBounds(13, 90, 400, 21);
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjTFDepartmentLocation;
}

/**
 * Return the SPTableScrollPane property value.
```

```
 * @return javax.swing.JScrollPane
 */
private javax.swing.JScrollPane getSPTableScrollPane() {
    if (ivjSPTableScrollPane == null) {
        try {
            ivjSPTableScrollPane = new javax.swing.JScrollPane();
            ivjSPTableScrollPane.setName("SPTableScrollPane");

ivjSPTableScrollPane.setVerticalScrollBarPolicy(javax.swing.JScrollPane.VERTICAL_SCROLLBAR_
ALWAYS);

ivjSPTableScrollPane.setHorizontalScrollBarPolicy(javax.swing.JScrollPane.HORIZONTAL_SCROLL
BAR_NEVER);
            ivjSPTableScrollPane.setBounds(13, 130, 400, 187);
            getSPTableScrollPane().setViewportView(getTBDepartmentTable());
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjSPTableScrollPane;
}

/**
 * Return the TBDepartmentTable property value.
 * @return javax.swing.JTable
 */
private javax.swing.JTable getTBDepartmentTable() {
    if (ivjTBDepartmentTable == null) {
        try {
            ivjTBDepartmentTable = new javax.swing.JTable();
            ivjTBDepartmentTable.setName("TBDepartmentTable");

getSPTableScrollPane().setColumnHeaderView(ivjTBDepartmentTable.getTableHeader());
            getSPTableScrollPane().getViewport().setBackingStoreEnabled(true);
            ivjTBDepartmentTable.setModel(new javax.swing.table.DefaultTableModel());
            ivjTBDepartmentTable.setBounds(0, 0, 200, 200);
            // user code begin {1}
            ivjTBDepartmentTable.getTableHeader().setReorderingAllowed(false);
            ivjTBDepartmentTable.getTableHeader().setResizingAllowed(false);
            ((DefaultTableModel)ivjTBDepartmentTable.getModel()).addColumn("Emloyee");
            ((DefaultTableModel)ivjTBDepartmentTable.getModel()).addColumn("Position");
            ((DefaultTableModel)ivjTBDepartmentTable.getModel()).addColumn("Years");
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjTBDepartmentTable;
}

/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
```

```
private void handleException(java.lang.Throwable exception) {

    /* Uncomment the following lines to print uncaught exceptions to stdout */
    // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
    // exception.printStackTrace(System.out);
}

/**
 * Initializes connections
 * @exception java.lang.Exception The exception description.
 */
private void initConnections() throws java.lang.Exception {
    // user code begin {1}
    // user code end
    getCBDepartmentName().addActionListener(ivjEventHandler);
}

/**
 * connEtoC1:  (CBDepartmentName.action.actionPerformed(java.awt.event.ActionEvent) -->
GraphicalJDBC.cBDepartmentName_ActionPerformed(Ljava.awt.event.ActionEvent;)V)
 * @param arg1 java.awt.event.ActionEvent
 */
private void connEtoC1(java.awt.event.ActionEvent arg1) {
    try {
        // user code begin {1}
        // user code end
        this.cBDepartmentName_ActionPerformed(arg1);
        // user code begin {2}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {3}
        // user code end
        handleException(ivjExc);
    }
}

/**
 * Department selected.
 */
public void cBDepartmentName_ActionPerformed(java.awt.event.ActionEvent actionEvent) {
    retrieveDepartmentInfo((String)getCBDepartmentName().getSelectedItem(),
getCBDepartmentName().getSelectedIndex());
    return;
}

/**
 * Initialize the class.
 */
private void initialize() {
    try {
        // user code begin {1}
        // user code end
        setName("GraphicalJDBC");
        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setSize(426, 366);
        setContentPane(getJDialogContentPane());
        initConnections();
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
```

```
    // user code begin {2}

    try {
        // load the Toolbox JDBC driver
        Class.forName("com.ibm.as400.access.AS400JDBCDriver");

        // get the connection to the iSeries database
        // if no user or password is given a prompt will appear
        String system = "itsosys";
        String url = "jdbc:as400://" + system;
        String user = "ITSO";
        String password = "ITSOPWD";
        connection = DriverManager.getConnection(url, user, password);
        meta = connection.getMetaData();

        // retrieve the departments
        retrieveDepartments();
    }
    catch(Exception e) {
    }

    // user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
    try {
        GraphicalJDBC aGraphicalJDBC;
        aGraphicalJDBC = new GraphicalJDBC();
        aGraphicalJDBC.setModal(true);
        aGraphicalJDBC.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
                System.exit(0);
            };
        });
        aGraphicalJDBC.show();
        java.awt.Insets insets = aGraphicalJDBC.getInsets();
        aGraphicalJDBC.setSize(aGraphicalJDBC.getWidth() + insets.left + insets.right,
aGraphicalJDBC.getHeight() + insets.top + insets.bottom);
        aGraphicalJDBC.setVisible(true);
    } catch (Throwable exception) {
        System.err.println("Exception occurred in main() of javax.swing.JDialog");
        exception.printStackTrace(System.out);
    }
}

/**
 * Retreive the information about a department.
 * @param args java.lang.String
 */
private void retrieveDepartmentInfo(String department, int selectedIndex) {
    DefaultTableModel tm = (DefaultTableModel)getTBDepartmentTable().getModel();

    // remove all the rows
    for (int x = tm.getRowCount(); x > 0; --x)
        tm.removeRow(x - 1);
```

```
    // if this is the first entry the clear everything out, else fill in the table
    if (selectedIndex == 0) {
        getTFDepartmentLocation().setText("");
    } else {
        try {
            DepartmentInfo dept = (DepartmentInfo)departments.elementAt(selectedIndex);
            getTFDepartmentLocation().setText(dept.getDivision()
                                              + " division located at "
                                              + dept.getLocation());

            PreparedStatement deptSelect = connection.prepareStatement(
                    "SELECT NAME, YEARS, JOB FROM ITSOSAMPLE"
                  + meta.getCatalogSeparator()
                  + "STAFF WHERE DEPT = ?");
            ResultSet deptResults = null;

            // add each record to the table
            deptSelect.setInt(1, dept.getNumber());
            deptResults = deptSelect.executeQuery();

            while (deptResults.next()) {
                Vector data = new Vector();
                data.add(deptResults.getString("NAME"));
                data.add(deptResults.getString("JOB"));
                data.add(new Integer(deptResults.getInt("YEARS")));
                tm.addRow(data);
            }
        }
        catch(Exception e) {
        }
    }
}

/**
 * Retrieve the list of departments.
 */
private void retrieveDepartments() {
    // create the default blank entry
    getCBDepartmentName().addItem("");
    DepartmentInfo defaultDepartment = new DepartmentInfo();
    defaultDepartment.setName("");
    defaultDepartment.setLocation("");
    defaultDepartment.setDivision("");
    defaultDepartment.setNumber(-1);
    departments.addElement(defaultDepartment);

    try {
        //! create and execute a simple SQL select statement
        Statement orgSelect = connection.createStatement();
        ResultSet orgResults = orgSelect.executeQuery("SELECT * FROM ITSOSAMPLE"
                                                      + meta.getCatalogSeparator()
                                                      + "ORG");

        while (orgResults.next()) {
            getCBDepartmentName().addItem(orgResults.getString("DEPTNAME"));
            DepartmentInfo dept = new DepartmentInfo();
            dept.setName(orgResults.getString("DEPTNAME"));
            dept.setLocation(orgResults.getString("LOCATION"));
            dept.setDivision(orgResults.getString("DIVISION"));
            dept.setNumber(orgResults.getInt("DEPTNUMB"));
```

```
                departments.addElement(dept);
            }
        }
        catch(Exception e) {
        }
}

}
```

Sample output

Figure 4-2 shows the output of the graphical JDBC code when run from the Linux platform using the ORG, STAFF, and SALES tables.

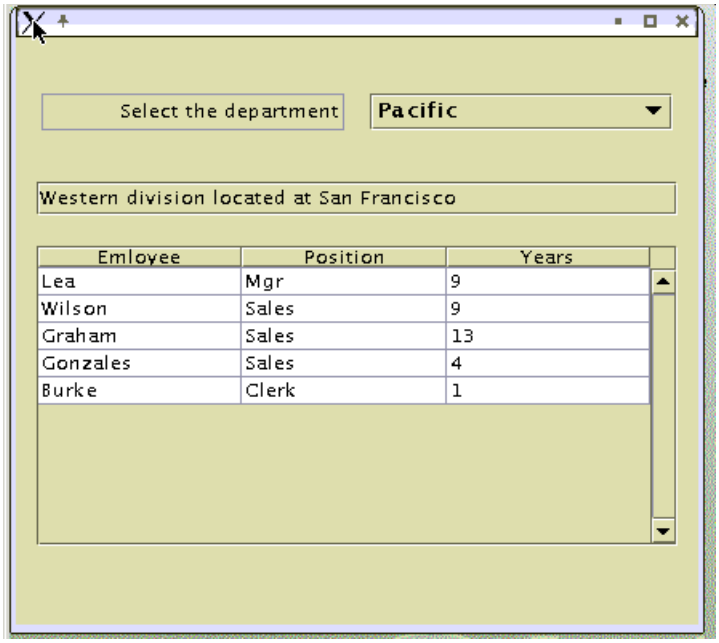

Figure 4-2 Graphical JDBC example

5

Other access methods

This chapter discusses other methods of accessing data stored in OS/400 from a Linux partition. It also discusses other methods for processes to communicate between the two platforms. It covers these topics:

- What other methods of access are available
- General considerations
  - ASCII and EBCDIC character sets
  - Endian byte order
- FTP
  - Transferring data from Linux
  - Transferring data from OS/400
  - Other FTP possibilities
- Sockets
  - Client example
  - Server example
- Toolbox for Java
- Remote login
  - Telnet
  - ssh

5.1 What other methods of access are available

There are many other methods of accessing data in OS/400 from Linux or even simply communicating between the OS/400 and Linux platforms. The methods that we discuss further are FTP, sockets, IBM Java Toolbox, and remote login. Other methods not discussed in this chapter include Remote Procedure Call (RPC) for C, Remote Method Invocation (RMI) for Java, and Samba/NFS for file systems.

5.2 General considerations

Before we go into some of the other methods of integration, there are a couple of items that you need to watch for:

- The character set used in the different platforms
- The byte order scheme used by the different architectures

5.2.1 ASCII and EBCDIC character sets

As you are probably aware, the iSeries runs natively in an EBCDIC character set while Windows and all flavors of UNIX (including Linux) run in an ASCII character set. This is most noticeable to you when you handle integration between the platforms at a lower level since higher levels of data movement usually handle the conversion automatically. The three mechanisms for handling the different character sets that are shown in this book are:

- Use the capabilities in UDB for iSeries. If the data you need and the integration method allows you to use database tables then you definitely should use them. The database capabilities in OS/400 will easily handle the conversion between ASCII and EBCDIC (as well as many other character set translations). This is mechanism is used by the ODBC and JDBC examples.
- Use Unicode. Unicode is a larger character set that is available on both the iSeries and Linux platforms. This is the approach that Java uses extensively.
- Do your own conversion. The conversion could be done on either the Linux or OS/400 side; however, it may be the easiest and best to do the conversion in OS/400. There are numerous tools available for character conversion. The socket example code demonstrates the use of the *iconv()* API to convert from one CCSID to another.

5.2.2 Endian byte order

The iSeries server works with PowerPC processors. One of the differences between the PowerPC architecture and Intel's x86 architecture is the byte ordering. A 32-bit binary integer is always represented as four bytes in the memory of a computer.

But, what part of the integer does the first byte get? The representation of the memory of a PowerPC processor is the way you would see it in a C source program. On a PowerPC processor, the line of code: *q = 0x0A0B0C0D;* is stored with the 0A byte at the first memory byte of *q*, 0B at the second and so on. This is called *big endian* because the high order ('big') byte is at the first memory offset.

However, on an x86 processor, the representation is the other way around. It is called *little endian* because the low order ('little') byte is stored at the first offset of *q* for the same statement *q = 0x0A0B0C0D;*. Therefore in little endian, the 0D is stored at the first byte position, 0C at the second and so on. There is a long history behind these differences; but,

what is important is that programmers understand the situation and code in a way to avoid depending on the exact order. This way, code written on an x86 Linux system can easily be ported to run on an iSeries Linux partition. Table 5-1 shows several examples and the differences.

Table 5-1 Bytes ordering (endian)

Decimal value, C source code value	Big endian representation	Little endian representation
125 0x7D	7D	7D
500 0x1F4	01 F4	F4 01
200,000,000 0xBEBC200	0B EB C2 00	00 C2 EB 0B
4,294,967,298 0x100000002	00 00 00 01 00 00 00 02	02 00 00 00 01 00 00 00

It is best to avoid byte ordering or endian problems by minimizing endian dependencies. You can use the following rules in C and C++ programs:

- Avoid casting smaller sized storage over larger sized ones (e.g. no (short *) over (int *)).
- Avoid casting larger sized storage over smaller sized ones (e.g. no (int *) over (short *)).
- Avoid casting one structure over another.
- Avoid unions with different sized items (this is almost the same as saying avoid unions).
- Avoid the use of bit variables.

If you cannot avoid these coding styles, then the use of suitable #ifdef statements should be employed with big endian and little endian versions of the code provided. This lets the compiler select the correct byte order dependent code.

5.3 FTP

FTP is the standard file transfer protocol for TCP/IP. FTP consists of both a client and a server application. A client application is included with most operating systems and is run normally in an interactive mode. The server application is also included in most operating systems. However, the server application runs in a background or batch mode and normally must explicitly be started.

There are two ways to transfer data in FTP, binary and non-binary mode. In binary mode, the bits of the file are copied exactly as is. No conversion is done to the data. Binary mode is used for transferring actual programs, image files, audio files, and other similar files. In non-binary mode, data may be converted between the two systems. Non-binary mode is used for text files, program source, and other files that are read by humans. By default, you start your FTP session in non-binary mode. To switch to binary mode, enter the FTP command **`binary`** or **`image`**.

To start an FTP session, enter **`FTP`** ***`server`***, where *server* is the host name or IP address of the system you want to transfer files with. Once the FTP session is started, you can use the **`get`** command to get files from the remote system to the local system and the **`put`** command to put files from the local system to the remote system.

Many **`get`** and **`put`** commands can be performed during a single FTP session. When you initially FTP into OS/400, you are in the QSYS file system. Files are accessed by entering library/file.member for member objects within a file or library/object for non-file objects. You can also access files in Integrated File System (IFS). To switch to IFS mode, you can enter the FTP command **`site namefmt 1`** switch to name format 1 or **`cd /`** to switch to name format

1 and change to the root directory. To access the same QSYS files, you would use /qsys.lib/library.lib/file.file/member.mbr or /qsys.lib/library.lib/object.type. To access files in IFS, you would use /directory1/directory2/file.name where file.name is the name of the file you want to transfer. To access files on Linux, use /directory1/directory2/file.name.

Other useful FTP commands are **cd** and **lcd** that change the directory location on the remote and local systems respectively. The **prompt** command is useful in conjunction with the **mget** or **mput** commands. The **prompt** command turns off interactive prompting that you want to do since **mget** or **mput** allows you to get or put multiple files with a single command.

5.3.1 Transferring data from Linux

To transfer data from Linux to the OS/400 while on the Linux partition, simply enter **FTP server** where *server* if the IP host name of the iSeries server or the IP address of the system. Once in the FTP session, you are first prompted for a user and then a password. This needs to be a valid OS/400 user profile and password. After you are logged on, you can use the **put** command to transfer a file from Linux to OS/400. By default, you will operate in the QSYS file system in OS/400. This is the traditional file system with libraries, files, and members. To move a program source file from Linux to OS/400, for example, you would issue the following command from within the FTP session:

```
put /home/itso/src/client.c /itso/qcsrc.client
```

This moves the client.c source file from /home/itso/src to member CLIENT in file QCSRC in library QCSRC. The data is converted from ASCII to EBCDIC by the FTP server. If the file was something like an image file that should not be translated, then you would first issue the **bin** FTP command. You can also get files from OS/400 to Linux using the **get** command. The following example shows how to get files from OS/400 to the Linux:

```
prompt
site namefmt 1
cd /itso/src
mget *.c
```

This sequence of commands retrieves all C source files out of the /itso/src IFS directory and places them in the current Linux directory. It uses the **prompt** command to turn off interactive prompting and then uses the **mget** command to get all files matching the wildcard.

5.3.2 Transferring data from OS/400

Transferring data from OS/400 is almost the same as transferring data from Linux. The FTP command is issued from an OS/400 command line and the FTP server needs to be started on the Linux partition. The only difference when starting the FTP session from OS/400 is how to get to IFS. When you initiate the FTP session from OS/400, enter command **namefmt 1** to get to IFS. The following sequence of commands accomplishes the same transfer of all C source files from OS/400 to Linux:

```
prompt
namefmt 1
lcd /itso/src
cd /home/itso/src
mput *.c
```

The **lcd** command changes the directory of the local system which in this case is the iSeries system. The **cd** command changes the directory on the remote or Linux system. Without the **prompt** command, before each file is transferred, you would be prompted whether to transfer that file.

5.3.3 Other FTP possibilities

So far we have shown the FTP commands only running interactively. While this is the predominate usage, it is possible to run FTP in a batch or program mode. Inside a CL program in OS/400 or a script file on the Linux partition, you can enter the **ftp** command, provide or be prompted for a user and password, and then run through a number of FTP commands. In this manner, one of the useful FTP commands is **rcmd**, which runs a remote command.

Example 5-1 contains a simple script file that when run, logs in as user ITSO, prompts for the password, creates an OS/400 library, creates a number of source physical files, puts a number of source files, and then finally creates one of the CL programs that was just downloaded and run the newly created program.

Example 5-1 Sample FTP script file

```
user itso

quote rcmd crtlib sysmgmt text('Systems Management Tools')

quote rcmd crtsrcpf sysmgmt/qcmdsrc text('CMD native source')
quote rcmd crtsrcpf sysmgmt/qclsrc text('CL Programs')
quote rcmd crtsrcpf sysmgmt/qcsrc text('C Programs')

put ~itso/sysmgmt/ctl.cmd sysmgmt/qcmdsrc.ctl
put ~itso/sysmgmt/dev.cmd sysmgmt/qcmdsrc.dev
put ~itso/sysmgmt/lin.cmd sysmgmt/qcmdsrc.lin
put ~itso/sysmgmt/pt.cmd sysmgmt/qcmdsrc.pt
put ~itso/sysmgmt/tn.cmd sysmgmt/qcmdsrc.tn
put ~itso/sysmgmt/chgcmddft.cl sysmgmt/qclsrc.chgcmddft
put ~itso/sysmgmt/startmsg.cl sysmgmt/qclsrc.startmsg
put ~itso/sysmgmt/startsys.cl sysmgmt/qclsrc.startsys
put ~itso/sysmgmt/startup.cl sysmgmt/qclsrc.startup
put ~itso/sysmgmt/mapkeys.cl sysmgmt/qclsrc.mapkeys
put ~itso/sysmgmt/create.cl sysmgmt/qclsrc.create
put ~itso/sysmgmt/snads.cl sysmgmt/qclsrc.snads
put ~itso/sysmgmt/ctl.cl sysmgmt/qclsrc.ctl
put ~itso/sysmgmt/dev.cl sysmgmt/qclsrc.dev
put ~itso/sysmgmt/lin.cl sysmgmt/qclsrc.lin
put ~itso/sysmgmt/pt.cl sysmgmt/qclsrc.pt
put ~itso/sysmgmt/tn.cl sysmgmt/qclsrc.tn
put ~itso/sysmgmt/snmpmgr.c sysmgmt/qcsrc.snmpmgr

quote rcmd crtclpgm sysmgmt/create sysmgmt/qclsrc

quote rcmd call sysmgmt/create

quit
```

To use this FTP script file on Linux, use the command:

```
ftp system < download.txt
```

5.4 Sockets

Sockets and the programs written on top of sockets will be able to handle almost any integration you want to provide between Linux and OS/400. In fact, most of the other methods discussed in this chapter use some flavor of sockets or another under the covers. The following example shows how to send a text message from one system to the other.

5.4.1 Client example

The client program takes as input up to two parameters. The first parameter is the host name or IP address of the server where the message will be sent. The second parameter is the text to send. If the second parameter is left off, then a default message is sent.

To create the client program on OS/400, issue the following commands in this order:

```
CRTCMOD MODULE(ITSO/CLIENT) SRCSTMF('/itso/src/client.c') TGTCCSID(37)
CRTPGM PGM(ITSO/CLIENT) MODULE(ITSO/CLIENT) BNDSRVPGM(QSYS/QTQICONV)
```

To create the client program on Linux, the following command is issued:

```
# gcc -o client client.c
```

To run the program on OS/400, you simply run the command:

```
CALL ITSO/CLIENT PARM(server 'This is my text message')
```

Here, *server* is the host name or IP address of the server you want to send your message to. On Linux, you run the program using:

```
client server 'This is my text message'
```

Here, *server* is again the host name or IP address of the server.

A bit more about the program

On OS/400 the program source code is stored in IFS under the directory /itso and is tagged with a CCSID of 1252. This makes for a very convenient way to edit the code from a PC. You only need to map a network drive to the iSeries server and then use your favorite editor to edit the file in IFS.

The client code is rather straightforward with #ifdef's around the OS/400 specific code. The same exact code will compile and run on OS/400 as well as Linux. To get the code to the Linux system, we simply sent the source over from the Linux system via FTP. When you FTP into OS/400, the standard location is into the QSYS file system.

Since our source is in IFS, we first needed to put FTP into the IFS mode for processing. There are two ways to do this:

- Issuing the **`site namefmt 1`** command
- Issuing the **`cd /`** command

Once in name format 1, you traverse IFS like a typical UNIX directory structure. The **`cd /itso`** command takes you into the /itso directory in IFS. From there, you can run the **`get client.c`** command to move the source code to your Linux system.

Client code

Example 5-2 lists the client code. The client code is a straightforward socket program that opens a socket on a specific port to a specific host system, sends data using the socket, and finally closes the socket. The only other code is the code to convert from EBCDIC to ASCII if on the iSeries server. Now lets look at the actual code.

Example 5-2 Client code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <errno.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define SERVER_PORT 30595
#define BUFFER_SIZE 250

#ifdef __OS400__
#include <iconv.h>

typedef struct {
  char name[8];
  char ccsid[5];
  char conversion[3];
  char substitute[1];
  char shift[1];
  char input[1];
  char option[1];
  char RESERVED[12];
} fromCode_t;

typedef struct {
  char name[8];
  char ccsid[5];
  char RESERVED[19];
} toCode_t;
#endif

int main(int argc, char *argv[])
{
  int rc;
  char buffer[BUFFER_SIZE],
       converted[BUFFER_SIZE],
       *inBuffer,
       *convertBuffer,
       server[255];

#ifdef __OS400__
  /*! iconv() variables ONLY ON OS/400 SERVER */
  size_t inBufferLength,
         convertBufferLength;
  fromCode_t fromCCSID;
  toCode_t toCCSID;
  iconv_t convertTable;
#endif


  /*! socket() variables */
  int descriptor,
      addressFamily,
      type,
      protocol;
```

```
  /*! sendto() variables */
  int bufferLength,
      flags,
      serverLength;
  struct sockaddr_in serverAddress;

  /*! gethostbyname() variables */
  struct hostent *host;

  /*! socket() */
  /*! sendto() */
  /*! close() */

  /*! read the text from the input */
  if (argc > 2) {
    bufferLength = (strlen(argv[2]) < BUFFER_SIZE) ? strlen(argv[2]) : BUFFER_SIZE;
    strncpy(buffer, argv[2], bufferLength);
  } else {
    strcpy(buffer, "Default data");
    bufferLength = strlen(buffer);
  }

  /*! open the socket descriptor */
  addressFamily = AF_INET;
  type = SOCK_DGRAM;
  protocol = IPPROTO_IP;
  if ((descriptor = socket(addressFamily, type, protocol)) < 0) {
    printf("Error getting socket descriptor\n");
    printf("errno is %i\n", errno);
    exit(-1);
  }

#ifdef __OS400__
  /*! setup the conversion table ONLY ON OS/400 SERVER*/
  memcpy(toCCSID.name, "IBMCCSID", 8);
  memcpy(toCCSID.ccsid, "01252", 5);
  memset(toCCSID.RESERVED, 0x00, 19);
  memcpy(fromCCSID.name, "IBMCCSID", 8);
  memcpy(fromCCSID.ccsid, "00037", 5);
  memcpy(fromCCSID.conversion, "000", 3);
  memcpy(fromCCSID.substitute, "0", 1);
  memcpy(fromCCSID.shift, "0", 1);
  memcpy(fromCCSID.input, "0", 1);
  memcpy(fromCCSID.option, "0", 1);
  memset(fromCCSID.RESERVED, 0x00, 12);
  convertTable = iconv_open((char *)&toCCSID, (char *)&fromCCSID);
  if (convertTable.return_value < 0) {
    printf("Error creating conversion table\n");
    printf("errno is %i\n", errno);
    close(descriptor);
    exit(-1);
  }

  /*! convert the data */
  convertBufferLength = inBufferLength = strlen(buffer);
  inBuffer = buffer;
  convertBuffer = converted;
  memset(converted, 0x00, sizeof(converted));
  iconv(convertTable, &inBuffer, &inBufferLength, &convertBuffer, &convertBufferLength);
```

```
#else
  memcpy(converted, buffer, bufferLength);
#endif

  /*! get the server's address */
  memset(&serverAddress, 0x00, sizeof(serverAddress));
  strcpy(server, argv[1]);
  if ((serverAddress.sin_addr.s_addr = inet_addr(server)) == (unsigned long)INADDR_NONE) {
    host = gethostbyname(server);
    if (host == (struct hostent *)NULL) {
      printf("Host not found\n");
      printf("h_errno is %d\n", h_errno);
      close(descriptor);
      exit(-1);
    }
    memcpy(&serverAddress.sin_addr, host->h_addr, sizeof(serverAddress.sin_addr));
  }

  /*! send to the server */
  flags = 0;
  serverAddress.sin_family = AF_INET;
  serverAddress.sin_port   = htons(SERVER_PORT);
  serverLength = sizeof(serverAddress);
  rc = sendto(descriptor, converted, bufferLength, flags,
              (struct sockaddr *)&serverAddress, serverLength);
  if (rc < 0) {
    printf("Error sending reqeusts\n");
    printf("errno is %i\n", errno);
    close(descriptor);
    exit(-1);
  }

#ifdef __OS400__
  /*! close the conversion table */
  iconv_close(convertTable);
#endif

  /*! close the socket */
  close(descriptor);

  exit(0);
} /*! ...end of main() */
```

5.4.2 Server example

The server program is the program that waits for incoming messages. It binds a socket to port 30595 (any free port will do) and then wait for incoming messages. Messages that it receives will be outputted to standard out. No parameters are required for the server program.

To create the server program on OS/400, issue the following commands in this order:

```
CRTCMOD MODULE(ITSO/SERVER) SRCSTMF('/itso/src/server.c') TGTCCSID(37)
CRTPGM PGM(ITSO/SERVER) MODULE(ITSO/SERVER) BNDSRVPGM(QSYS/QTQICONV)
```

To create the server program on Linux, issue the following command:

```
# gcc -o server server.c
```

Again, the source was simply sent by FTP from OS/400 IFS to the Linux system following the same process as we did for the client program.

You can run the server in either interactive mode or in batch mode. On both OS/400 and Linux, the messages sent to the server are output to standard out. To run the server interactively on OS/400, you run the CALL ITSO/SERVER command. To run the server interactively on Linux, you use **server**. To run the server code in a background or batch jobs, use the following commands:

On OS/400, enter:

```
SBMJOB CMD(CALL ITSO/SERVER)
```

This submits a batch job, and by default, standard out goes to a spooled file.

On Linux, enter:

```
# server > output.file &
```

This starts a background job and redirects standard out to the output.file file in the current directory.

A bit more about the program

Similar to the client program, the source code for the server is stored in IFS on the OS/400 system. It was also sent by FTP to the Linux system and compiled. The biggest design decision for either the server of the client code was how to handle the conversion from ASCII to EBCDIC and vice versa.

One way to handle this would be to add one additional byte to the front of the text message. All 0s could indicate that the incoming text is EBCDIC and all 1s could indicate that the message is ASCII. It would then be up to the server to handle all conversions. The down side to this approach is that conversion would be required on a Linux server and that is more difficult than conversion on OS/400.

The approach we took was to always send the message in ASCII format. This means that only the OS/400 server needs to convert incoming messages to EBCDIC. It also means that the OS/400 client needs to convert its message before sending.

Server code

Example 5-3 shows the server code. The server code waits on a specific port for a client to send a message. When the message is received, it is sent to standard output. The only difference between the Linux version and the OS/400 version is the need to convert the incoming message from ASCII to EBCDIC. Now for the server code.

Example 5-3 Server code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define SERVER_PORT 30595
#define BACK_LOG 20
#define BUFFER_SIZE 250
```

```
#ifdef __OS400__
#include <iconv.h>

typedef struct {
  char name[8];
  char ccsid[5];
  char conversion[3];
  char substitute[1];
  char shift[1];
  char input[1];
  char option[1];
  char RESERVED[12];
} fromCode_t;

typedef struct {
  char name[8];
  char ccsid[5];
  char RESERVED[19];
} toCode_t;
#endif

int main(int argc, char *argv[])
{
  int rc,
      bufferLength;
  char buffer[BUFFER_SIZE],
       converted[BUFFER_SIZE],
       *inBuffer,
       *convertBuffer;

#ifdef __OS400__
  /*! iconv() variables ONLY ON OS/400 SERVER */
  size_t inBufferLength,
         convertBufferLength;
  fromCode_t fromCCSID;
  toCode_t toCCSID;
  iconv_t convertTable;
#endif

  /*! socket() variables */
  int descriptor,
      addressFamily,
      type,
      protocol;

  /*! bind() variables */
  struct sockaddr_in localAddress;
  int addressLength;

  /*! recvfrom() variables */
  int flags,
      clientLength;
  struct sockaddr_in clientAddress;

  /*! socket() */
  /*! bind() */
  /*! recvfrom() */
  /*! close() */

  /*! open the socket descriptor */
```

```
  addressFamily = AF_INET;
  type = SOCK_DGRAM;
  protocol = IPPROTO_IP;
  if ((descriptor = socket(addressFamily, type, protocol)) < 0) {
    printf("Error getting socket descriptor\n");
    printf("errno is %i\n", errno);
    exit(-1);
  }

  /*! bind to the socket */
  memset(&localAddress, 0x00, sizeof(localAddress));
  localAddress.sin_family = AF_INET;
  localAddress.sin_port   = htons(SERVER_PORT);
  localAddress.sin_addr.s_addr = htonl(INADDR_ANY);
  addressLength = sizeof(localAddress);
  if ((rc = bind(descriptor, (struct sockaddr *)&localAddress, addressLength)) < 0) {
    printf("Error binding to the socket\n");
    printf("errno is %i\n", errno);
    close(descriptor);
    exit(-1);
  }

#ifdef __OS400__
  /*! setup the conversion table ONLY ON OS/400 SERVER*/
  memcpy(toCCSID.name, "IBMCCSID", 8);
  memcpy(toCCSID.ccsid, "00037", 5);
  memset(toCCSID.RESERVED, 0x00, 19);
  memcpy(fromCCSID.name, "IBMCCSID", 8);
  memcpy(fromCCSID.ccsid, "01252", 5);
  memcpy(fromCCSID.conversion, "000", 3);
  memcpy(fromCCSID.substitute, "0", 1);
  memcpy(fromCCSID.shift, "0", 1);
  memcpy(fromCCSID.input, "0", 1);
  memcpy(fromCCSID.option, "0", 1);
  memset(fromCCSID.RESERVED, 0x00, 12);
  convertTable = iconv_open((char *)&toCCSID, (char *)&fromCCSID);
  if (convertTable.return_value < 0) {
    printf("Error creating conversion table\n");
    printf("errno is %i\n", errno);
    close(descriptor);
    exit(-1);
  }
#endif

  do {
    /*! receive any incoming client requests */
    memset(buffer, 0x00, sizeof(buffer));
    bufferLength = sizeof(buffer);
    flags = 0;
    clientLength = sizeof(clientAddress);
    rc = recvfrom(descriptor, buffer, bufferLength, flags,
                  (struct sockaddr *)&clientAddress, &clientLength);
    if (rc < 0) {
      printf("Error receiving incoming client reqeusts\n");
      printf("errno is %i\n", errno);
      close(descriptor);
      exit(-1);
    }

#ifdef __OS400__
```

```
    /*! convert the data */
    convertBufferLength = inBufferLength = strlen(buffer);
    inBuffer = buffer;
    convertBuffer = converted;
    memset(converted, 0x00, sizeof(converted));
    iconv(convertTable, &inBuffer, &inBufferLength, &convertBuffer, &convertBufferLength);
#else
    memcpy(converted, buffer, sizeof(buffer));
#endif

    /*! output the data */
    printf("Received data from: %.i\n", clientAddress.sin_addr.s_addr);
    printf("'%.80s'\n\n", converted);

    /*! stop when 'STOP' is received as the first 4 characters */
  } while (strncmp(converted, "STOP", 4) != 0);

#ifdef __OS400__
  /*! close the conversion table */
  iconv_close(convertTable);
#endif

  /*! close the socket */
  close(descriptor);

  exit(0);
} /*! ...end of main() */
```

Sample output

Example 5-4 shows a sample of the output generated by a server running on a Linux partition.

Example 5-4 Sample server output

```
# server
Received data from: 151346265
'This is a sample message from OS/400'

Received data from: 151346266
'This is sample text from a Linux partition'

Received data from: 151346266
'STOP'
```

5.5 Toolbox for Java

IBM Toolbox for Java is where we found our JDBC driver in 4.3, “Command line JDBC application” on page 51. The Toolbox contains may other useful classes for access the iSeries system. You can use the Toolbox to do such tasks as run OS/400 commands, create OS/400 users and groups, call OS/400 programs and APIs, and access other OS/400 objects. Also included in the toolbox are graphical classes that make presenting OS/400 information and objects easy and consistent. The toolbox itself is Java based and runs on any

JVM 1.1.8 or above. The toolbox was also written as an open source project and the source code is available under the IBM Public license. You can find more information about the toolbox and the open source features at:
http://www.ibm.com/servers/eserver/iseries/toolbox/

Running an OS/400 command

Example 5-5 is Java program that shows just how easy it is to use the toolbox to run an OS/400 command. In this example, the user is prompted for a library name to create. The toolbox CommandCall object is then used to create the library. All messages generated by the command are then retrieved and shown to the user.

If a user and password are not given on the AS400 object constructor, the user is prompted for them. Again like the JDBC example, when the prompt for user and password is generated, Java font errors were generated on our Linux system even though the prompt appeared and worked.

Example 5-5 IBM Toolbox for Java command example

```
import java.io.*;
import java.util.*;
import com.ibm.as400.access.*;

public class ToolboxCommand
{
    public static void main(String[] argc) throws Exception
    {
    String systemName = "asm01.rchland.ibm.com";
    String user = "itso";
    String password = "itsopwd";
    AS400 system = new AS400(systemName, user, password);

        BufferedReader inputStream = new BufferedReader(new InputStreamReader(System.in),
1);
    String libraryName = null;
    System.out.println(" ");
    System.out.print("Library to create: ");
    libraryName = inputStream.readLine();

    System.out.println(" ");
    CommandCall command = new CommandCall(system);
    if (command.run("crtlib " + libraryName)) {
        System.out.println("Success");
    } else {
        System.out.println("Failure");
    }
    System.out.println(" ");

    AS400Message[] messageList = command.getMessageList();

    if (messageList.length > 0) {
        System.out.println("Messages returned are: ");
    }

    for (int i = 0; i < messageList.length; ++i) {
        System.out.print(messageList[i].getID());
        System.out.print(": ");
        System.out.println(messageList[i].getText());
    }
```

```
    system.disconnectService(AS400.COMMAND);

    System.exit(0);
    }
}
```

Sample output

Example 5-6 shows a successful library creation and an unsuccessful creation. In both cases, the OS/400 command is run under the user profile associated with the connection (in our case ITSO) and runs in a job under the QSERVER subsystem on OS/400. The same server jobs that are used for Client Access are used for the toolbox, but a license for Client Access is not required.

Example 5-6 Toolbox command sample output

```
# java ToolboxCommand

Library to create: itsotest

Success

Messages returned are:
CPC2102: Library ITSOTEST created.
#java ToolboxCommand

Library to create: itsotest

Failure

Messages returned are:
CPF2111: Library ITSOTEST already exists.
```

5.6 Remote login

So far, all the access methods that we have discussed have been program or batch-oriented methods. You can also access OS/400 and Linux partitions on OS/400 using interactive methods. In fact, since a Linux partition on OS/400 does not have an attached console, keyboard, or mouse all interactive use of the Linux partition requires some form of remote login. The two methods of interactive use that we discuss further are Telnet and ssh. There are other forms that are not discussed here, but most of them are a variation of Telnet or ssh.

Both Telnet and ssh are IP protocols and both require the remote system to have a server job running. Telnet uses IP port 23, while ssh uses IP port 22.

5.6.1 Telnet

Telnet is the traditional remote interactive access method. Telnet servers are available on most every operating system and there are many telnet clients available. A basic Telnet client usually takes you to a command prompt but may not handle formatting correctly. Therefore, the basic Telnet client should be used only in emergencies. To access the Linux partitions using Telnet, a good Windows client to use is PuTTY from:
http://www.chiark.greenend.org.uk/~sgtatham/putty/

PuTTY is a free Windows-based client that is widely used. The most best thing about PuTTY is that it also includes an ssh client. When you bring up PuTTY, you have your choice of opening your session using raw, telnet, rlogin, or ssh as the protocol.

To access OS/400 via a Windows telnet client, you can use the IBM Personal Communications client or PComm. A version of PComm is included with Client Access and Operations Navigator. When telnetting to OS/400 from a Linux partition, the preferred client is x5250.

The major drawback to Telnet is that it is insecure by design. All data transmissions are done in the clear, and there is no encrypting or masking of any data. All passwords and commands that you enter are sent in the clear. Therefore, basic Telnet should only be used when you are in a trusted LAN environment.

One notable exception to using Telnet is the Linux virtual console on iSeries. The Linux virtual console on iSeries runs via Telnet to port 2301 on the hosting partition. You should takes precautions to use the virtual console only on a trusted LAN network.

5.6.2 ssh

The ssh (secure shell) protocol is the preferred method for accessing Linux partitions on OS/400. In fact, the Linux partitions on OS/400 are configured by default to not allow Telnet sessions to come in but allow ssh sessions. The reason ssh is better is that ssh encrypts all data sent between the client and server, and Telnet does not. There are several Windows clients available with PuTTY again being the most widely used. The PuTTY client can be found at: http://www.chiark.greenend.org.uk/~sgtatham/putty/

OS/400 does not have an ssh server implemented. If you need to Telnet into OS/400 over an unsecured network, then you need to use an SSL connection for telnet. This is available on OS/400 and through the Personal Communications client.

6

Advantages of using virtual LAN and virtual DASD

This chapter discusses the advantages using the virtual LAN and virtual DASD with Linux logical partitions, as opposed to the native equivalents. It covers:

- The benefits of using the virtual LAN
- The benefits of using virtual DASD

6.1 Virtual LAN and virtual DASD

The integration of Linux with OS/400 on an iSeries logical partition was enhanced from the outset by the development of the concept of virtual LAN and virtual DASD. Although support also exists for other virtual I/O devices, the aim of this chapter is only to discuss the virtues of virtual LAN and virtual DASD.

From OS/400 the Network Server Storage Space (NWSSTG) provides the virtual DASD for the Linux partition and resides in the IFS.

A possible 16 separate virtual LANs can be configured, which emulate a gigabit Ethernet connection. This results in the potential for fast communication between Linux and OS/400 logical partitions, as well as with the primary OS/400 partition.

You can find specific details on setting up the virtual LAN and virtual DASD in *Linux on iSeries: An Implementation Guide* SG24-6232.

The decision of whether to configure one or more Linux partitions to use the virtual LAN and virtual DASD depends on various factors. Some of these factors are:

- Number of Linux partitions
- iSeries system model and available card positions and DASD slots
- Application type running on various Linux partitions
- Requirement for Linux partition to communicate outside of the iSeries server
- How disks should be protected
- Frequency of data retrieval from storage

6.1.1 The advantage of using virtual LAN

The virtual LAN, as mentioned earlier, provides a fast internal communication method between logical partitions on the iSeries. This mechanism was developed to allow partitions running Linux to communicate with each other. Traditionally, OS/400 partitions have used Virtual Opticonnect to communicate between each other. However, Linux does not know about Virtual Opticonnect.

There are many benefits to using the virtual LAN as a means of enhancing the performance and usability of Linux running on one or more iSeries logical partitions. Here are some of them:

- *Economical*: Potentially no extra networking hardware is required. Extra Linux partitions can be added to the iSeries server and still communicate with the outside world without needing to install extra physical LAN cards.
- *Extend the investment of an existing iSeries server*: If the current server has limited available card slots in which to install additional LAN cards, then using the virtual LAN offers the capability to operate LAN attached Linux partitions without the requirement to upgrade the whole iSeries server.
- *Flexibility*: It is possible to configure a maximum of 16 distinctive connections enabling the configuration of selective communication paths between partitions (see Figure 6-1). For added flexibility, the configuration model allows logical partitions to implement both a virtual LAN and native LAN connection. This is a desirable feature when using the Linux partition to host a firewall application.

```
                    Work with Virtual LAN Configuration
                                                          System:   S105ZY9M
Type options, press Enter.
  2=Change

    Par          -------------------Virtual LAN Identifiers-------------------
Opt ID  Name     0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
      0 PRIMARY  1   .   .   .   .   .   .   1   .   .   .   .   .   .   .   .
      1 LINUX1   1   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
      2 LINUX2   1   .   .   1   .   .   .   1   .   .   .   .   .   .   .   .
      3 LINUX3   1   .   .   1   .   .   .   .   .   .   .   .   .   .   .   .
```

Figure 6-1 Example virtual LAN configuration screen

- *Speed*: The virtual LAN emulates a 1 GB Ethernet connection and provides a fast and convenient communication method between partitions. This enhances the opportunity to integrate separate applications which run on different logical partitions.
- *Ease of use*: The virtual LAN is configured centrally from the primary OS/400 partition, using the System Services Tool (SST) or Operations Navigator interface. This means that traditional OS/400 users, possessing minimal Linux knowledge, should be able to easily configure network attached Linux partitions.
- *Reduction in LAN congestion*: Utilizing the virtual LAN for inter-partition communication prevents additional communication traffic from cluttering the external LAN. In the case of Ethernet, which is a collision-based standard, this will certainly help prevent a degradation of service for other LAN users.

In system configurations with a lot of inter-partition communication, use of the virtual LAN is an ideal solution. However, the choice to implement *only* the virtual LAN may depend on the number of configured Linux partitions and the type of applications running on them.

For example, in situations where Linux is being used to run Apache Web server applications, resulting in a high volume of communication to and from the Internet, then a native LAN may be advisable. If *only* the virtual LAN was configured, then the OS/400 LAN adaptor may become a “bottle-neck” for excessive Web traffic. However, in cases where the Web application also needs to send many ODBC requests to one or more DB2 databases running on the primary OS/400 partition, then the virtual LAN is an ideal medium for this type of traffic.

6.1.2 The advantage of using virtual DASD

Virtual DASD is assigned from OS/400 in the form of a Network Storage Space (NWSSTG) and is stored on the OS/400 Integrated File System in the form of a stream file. There are many benefits of allocating virtual DASD to Linux logical partitions, rather than the alternative of assigning native disks to each Linux system:

- *Flexibility*: From the outset, the minimum sized NWSSTG can be allocated to each Linux partition. If at a later date extra disk space is required by the Linux system, it can be easily added in the following ways:
  - To increase the size of the existing disk, the NWSSTG can be copied into a larger one.
  - Extra disks can be added by creating additional NWSSTG spaces and linking them to the one NWSD (they will need to be mounted from Linux).
- *Optimum data retrieval performance*: Installing Linux on virtual DASD utilizes the OS/400 concept of single level storage. Data is stored across multiple disk drives. Therefore, data retrieval performance is enhanced by the use of multiple disk arms.

- *Utilize existing OS/400 backup strategies*: Because virtual DASD is stored in the IFS existing backup strategies, using the SAV command ensures that data relating to *all* Linux partitions is saved at the same time. With native disks, the data needs to be saved from within each individual Linux system.
- *Disk protection*: Network Storage Spaces are stored in the Integrated File System within OS/400. Therefore, they can take advantage of OS/400 RAID and mirroring disk protection, and when required, the “hot replacement” of failed disks drives. If Linux uses a native disk, then disk protection needs to be implemented and controlled from Linux.
- *Ease of use*: Implementing a Linux logical partition using virtual DASD is configured from OS/400. Therefore, a Linux system can be initially installed and configured by OS/400 users possessing minimal knowledge of the Linux operating system.
- *Versatile Linux development*: Different levels of the Linux kernel can be installed on separate network storage spaces. This facilitates a convenient development and test environment, as when required, a different network storage space can be linked to the one network server description.
- *Preservation of data during Linux re-installation*: If Linux is re-installed on the OS/400 logical partition, care needs to be taken to avoid the loss of data. For example, a workstation or custom-class installation using automatic partitioning will remove all data on all Linux disk partitions within the virtual or physical disks associated with the current Linux logical partition. One strategy is to separate the Linux distribution from applications and application data on different virtual disks. The NWSSTG holding the application related data can be “unlinked” from the NWSD during an Linux upgrade or re-installation to preserve it.

7

NFS export of Linux files from OS/400 Integrated File System

This chapter discusses a Linux administration and management method whereby certain common Linux directories and files can be placed on the OS/400 Integrated File System. Using NFS these files can be accessed and shared between multiple Linux partitions.

The following topics are discussed:

- Introduction to this strategy
- Overview of a working example

7.1 Why install parts of Linux on the OS/400 IFS

The “i” in iSeries stands for integration. One major rationale supporting logical partitioning is the cost savings in not having to purchase extra hardware to run separate systems. The aim of this chapter is to demonstrate how further cost savings can be made by the careful administration of iSeries logical partitions running Linux, in particular, by storing certain Linux files separately on the OS/400 Integrated File system (IFS).

The nature and structure of the Linux file system lends itself to being stored, or partitioned, in different places. The Network File System (NFS) was developed to allow machines to mount disk partitions on remote systems as if they were on a local hard drive. This facilitates the seamless sharing of files across a network. Both Linux and OS/400 support NFS.

In situations where two or more Linux logical partitions (ideally of the same distribution) are being used to run applications and packages accessing common files, it is possible to store one copy of these files on the IFS, sharing them using NFS. This is an adaptation of the method used to support diskless Linux workstations.

The benefits for implementing this type of installation are:

- *Less storage space is required for each Linux partition*: For example, removing /usr from every individual logical partition would result in each respective Network Storage Space (NWSSTG) being as little as 500 MB in size. This will ultimately lead to an optimum amount DASD requirement in the iSeries server.
- *Standardized Linux kernel upgrade*: In certain situations, only one “footprint” of the kernel needs to be stored on the IFS. With enough Linux “intelligence” in each respective NWSSTG, this can be exported and shared with multiple Linux installations, therefore centralizing administration, ugrades and patching.
- *Use existing OS/400 save and restore procedures*: Files stored on the IFS are saved as part of the OS/400 SAV command.
- *Saving on other hardware*: Implementing the iSeries virtual LAN means that only one physical LAN IOP may be required.

Therefore, adopting such a strategy can enable an integrated iSeries and Linux system installation to further reduce the total cost of your IT infrastructure.

7.2 Implementation of this solution

There are various ways to approach this type of system management. However, when implementing this on an iSeries logical partitioned system, you need to consider the following points:

- The OS/400 NWSSTG, which holds the Linux operating system, when created, must be specified at a certain size. A NWSSTG is *not* a dynamic storage space so careful thought has to be taken in the planning phase when deciding on its initial size.
- The choice of which files to place on the IFS very much depends on the intended Linux application type running on each partition. For example, in cases where Linux users are likely to create many files, **`/home`** may be a candidate. In other scenarios, such as firewall implementations, in which little user data will be generated, placing the kernel and any log files onto the IFS may be feasible.
- One option is to initially configure a fully populated Linux partition containing all of the required Linux packages. This allows all identified files to then be copied to the IFS. When

done, the NWSSTG can be deleted and a new one created containing a minimal installation of Linux.

Note: If in the future more space is required, the existing NWSSTG can be copied into another NWSSTG. The size of the new storage space *must* be equal to or greater than the size of the original network server storage space.

- For any additional Linux logical partitions, a minimal fully functioning Linux operating system should be installed onto each of these partitions. After networking is configured and the appropriate NFS mounts are defined in /etc/fstab, any redundant files on each Linux system can be deleted.

7.2.1 A working example

This section demonstrates how to achieve a simple implementation of this solution.

Overview

An overview of the steps for this example is highlighted here:

1. Select a Linux distribution. In our test, we install Red Hat 7.1 for Power PC onto two iSeries logical partitions. Linux and all available packages are installed onto the first partition. The second partition contains a minimal installation. For this demonstration, we place /usr on the IFS and eventually share it between two minimal Linux installations.
2. Set up networking between OS/400 and the Linux partitions. We use the iSeries virtual LAN and proxy ARP.
3. Copy the Linux files to IFS. On the OS/400 IFS, create a directory to hold the Linux files. Using use one of the various methods copy the Linux files to this directory. This can be achieved with **ftp**. Alternatively, configure Samba or NFS and use the **smbmount** or **mount** respectively. Use the Linux copy command (**cp**) to move the files to the IFS. In the example, we copy the /usr Linux file system to directory /dunnas on the OS/400 IFS.
4. The partition can now be deleted.
5. Create a replacement, smaller NWSSTG and configure a minimal Linux installation.
6. Create a NFS mount. Do this from the exported /dunnas/usr directory in OS/400 IFS to the Linux mount-point /usr. Test and ensure that all functions still work.
7. Update `/etc/fstab` with the NFS mount and test. This ensures that the NFS mount is reestablished after a Linux reboot.
8. Configure `/etc/fstab` to automatically mount /usr on Linux with /dunnas/usr on the IFS. At this point, do not run the NFS **mount** command.
9. Rename /usr on Linux to something else, such as `/usr1`.
10. Create a new directory on the Linux system called `/usr`. This is the NFS mount point.
11. Delete **/usr1** from the Linux partition. This removes all of the files associated with the original /usr file system, freeing up the associated space to Linux. This *does not* reduce the size of the NWSSTG.
12. The first Linux partition is now configured to share the /usr file system from the IFS.
13. Repeat steps 5 through to 11 with the second Linux partition.
14. The result is two separate Linux partitions sharing the one copy of /usr from the IFS.

Detailed instructions

The assumption is that, at this point, you have created two guest partitions from OS/400. Linux has been installed onto one logical partition and TCP/IP networking has been correctly configured, enabling the Linux operating system to successfully communicate with OS/400 on the primary partition. The first partition should include Linux plus all of the packages, such as, GNOME, KDE etc.

In the following example, we use Red Hat Linux 7.1. *The Official Red Hat Linux iSeries Installation Guide*, which is included on the distribution CDs, states that a custom-class installation requires 2.4 GB of free space if every package is selected. Therefore, the NWSSTG should be large enough to reflect this size.

For a good source of information for the correct installation and configuration of Linux on an iSeries logical partition, see *Linux on the iSeries Server: An Implementation Guide,* SG24-6232.

You must have an understanding of how NFS runs on both OS/400 and Linux. You can learn about OS/400 NFS configuration in *Exploring NFS on AS/400*, SG24-2150. And, you can find more information on Linux NFS at:
`http://www.hijinks.com/~spade/linux/howto/NFS-HOWTO/index.html`

Setting up NFS on OS/400

In the following example, Operations Navigator is used to configure and manipulate the OS/400 side. Follow these steps:

1. Create a directory in the IFS to hold the Linux files.
2. On the iSeries, start the NFS server. Select **Network-> Servers-> TCP/IP**. Right-click **NFS** and then select **Start all** (see Figure 7-1).

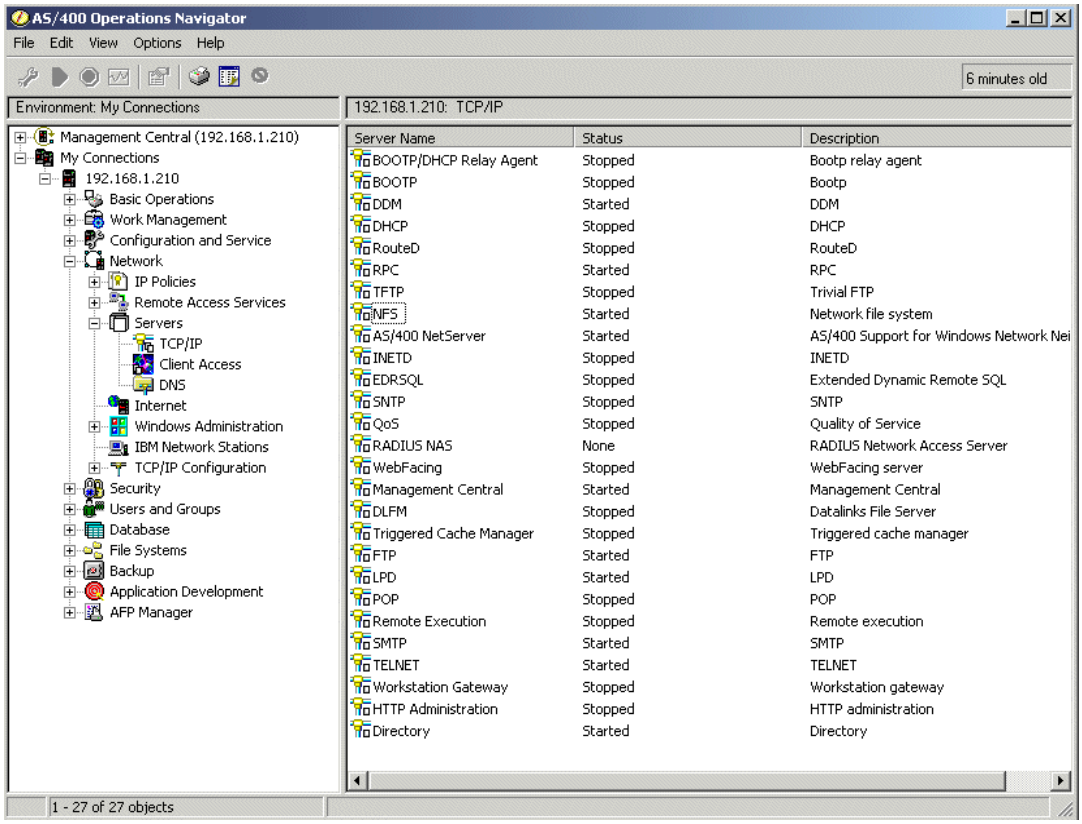

Figure 7-1 Selecting the NFS server using Operations Navigator

3. From the IFS, select the desired directory that will hold the Linux files. Right-click and select **NFS Export** and then **Properties** (see Figure 7-2).

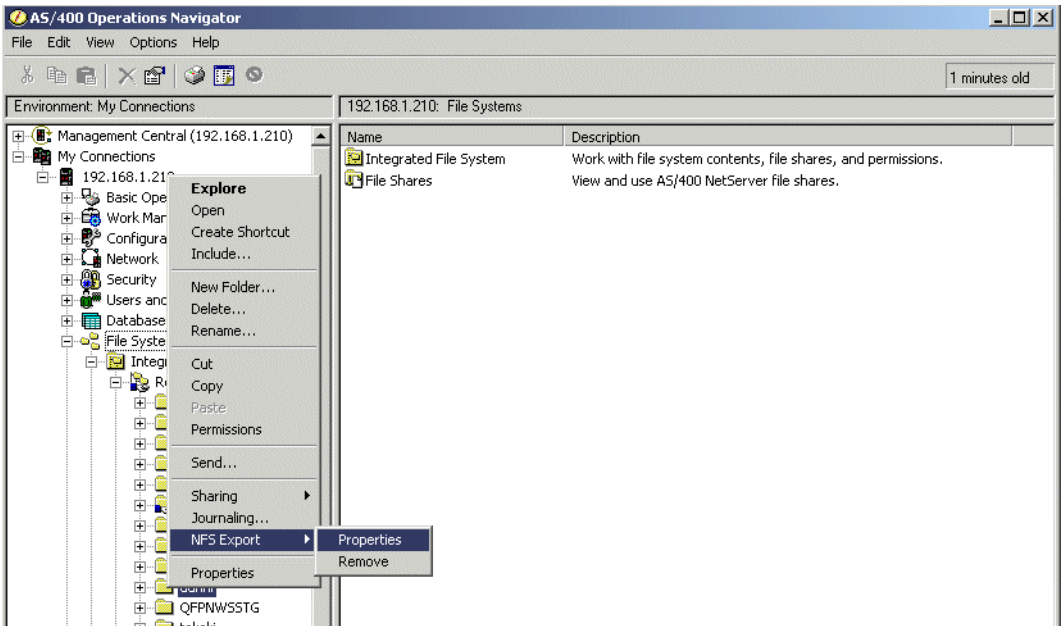

Figure 7-2 Select NFS Export IFS directory

4. Verify that the details are correct (by default, the OS/400 user profile QNFSANON is used). Then select **Export** (see Figure 7-3).

Note: If you select “Add to list of permanently defined exports”, an entry is added to the IFS /etc/EXPORTS file.

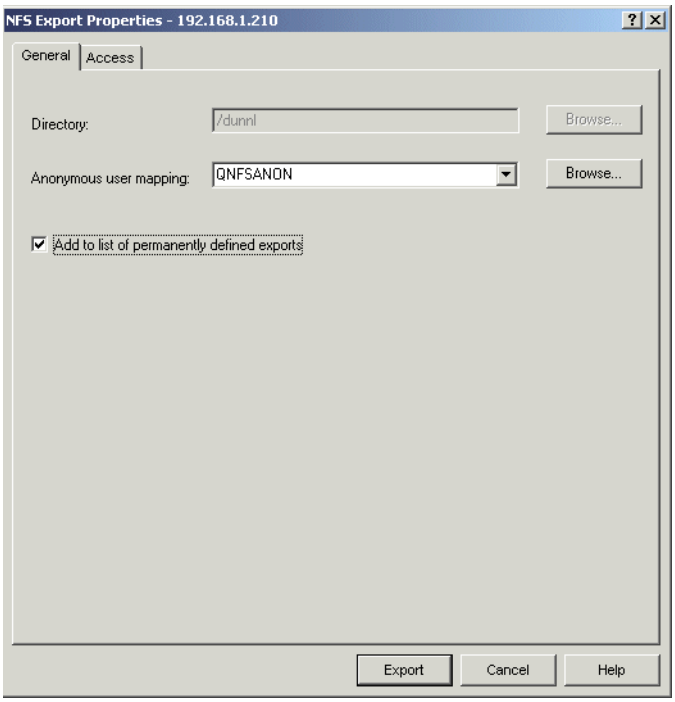

Figure 7-3 Exporting an IFS directory

NFS is now setup on OS/400.

Copying /usr from Linux file to OS/400

You can use various methods to copy files from Linux to the IFS. In this example, we mount the IFS directory from Linux using NFS and then use the Linux copy command (**cp**). Follow these steps:

1. Log on to the first Linux partition as root.
2. Create a NFS mount to /dunnas from a Linux mountpoint, for example `/mnt`:

   ```
   mount ASLN1:/dunnas /mnt
   ```

3. Use the **mount** command to check that the NFS mount is successful. You should see something similar to this:

   ```
   /dev/iSeries/vda1 on / type ext2 (rw)
   none on /proc type proc (rw)
   none on /dev/pts type devpts (rw,gid=5,mode=620)
   ASLN1:/dunnas on /mnt type nfs (rw,addr=192.168.1.210)
   ```

4. Make /mnt the home directory. This will actually access /dunnas on the IFS:

   ```
   cd /mnt
   ```

5. Copy /usr to /mnt. This may take some time:

   ```
   cp -a /usr
   ```

Changing the OS/400 NFS server file export options

After /usr is copied, it now appears as a directory within /dunnas. You should see something similar to the example in Figure 7-4.

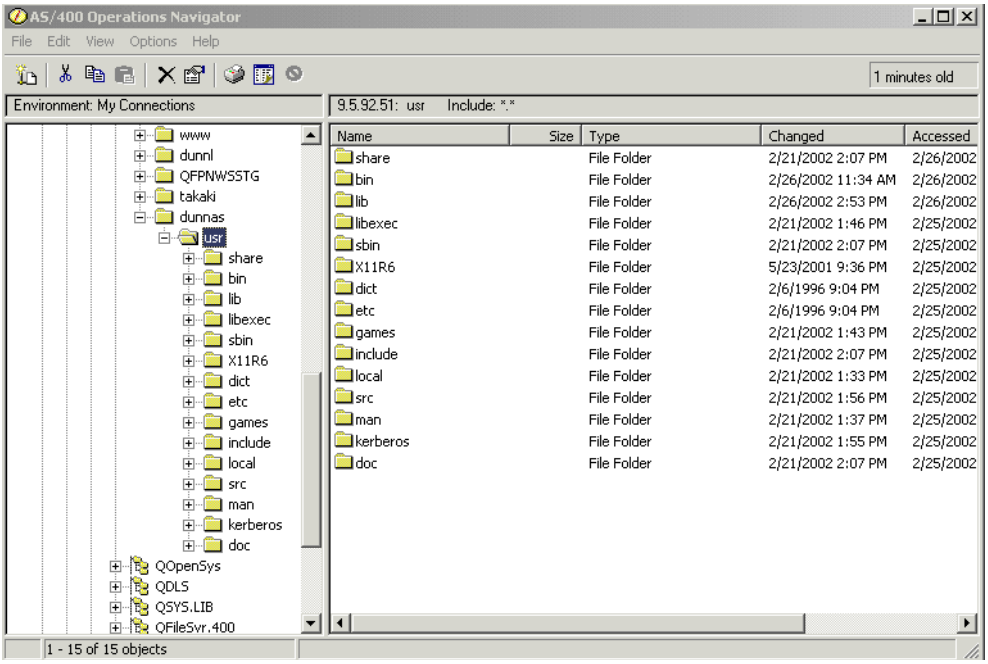

Figure 7-4 Result of copying /usr from Linux to OS/400

1. Remove the original NFS export to /dunnas and create a new one for /dunnas/usr.
2. Ensure that /dunnas/usr has the correct permissions. (Right click **/usr**, and then select **Permissions** (see Figure 7-5).

Important: In a live implementation, you should give careful thought to the permissions assigned to the files on the server. For example, if you do not want users to change any of the files, define the permissions as READ ONLY.

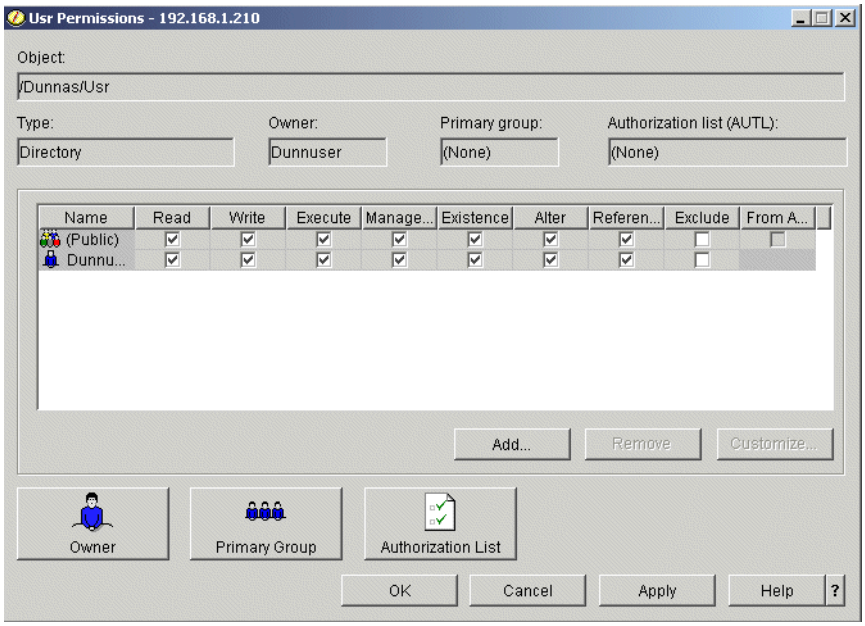

Figure 7-5 User permissions window

Mounting from Linux and testing

Now you are ready to mount /usr on Linux from /dunnas/usr on OS/400. Follow these steps:

1. Configure the /etc/hosts file with the fully qualified host name, the host name, and the IP address of both the Linux and OS/400 partitions. Use a text editor, such as **vi**, to edit the /etc/hosts file. It should look similar to this:

   ```
   127.0.0.1 LINUX.ROCHESTER.IBM.COM LINUX localhost.localdomain localhost
   192.168.1.202 LINUX1.ROCHESTER.IBM.COM LINUX1
   192.168.1.210 ASLN1.ROCHESTER.IBM.COM ASLN1
   ```

   Note: An optional step is to create a "test" Linux directory from which to mount to /usr in the IFS. For example, enter the following commands in the order shown:

   ```
   mkdir /fakeusr
   mount ASLN1:/dunnas/usr /fakeusr
   cd fakeusr
   ```

   Ensure that you can access and manipulate a selection of the files successfully.

2. Mount /usr on Linux from /dunnas/usr on the IFS:

   Note: Establishing a NFS **mount** renders the mount point, and all files underneath "invisible" to the local "client" system. The client can only see the files that are being mounted on the remote system. Therefore, in our example, after mounting /usr on the Linux system with /dunnas/usr on the IFS, the Linux system will access the */usr* files on the IFS.

   ```
   mount ASLN1:/dunnas/usr /usr
   ```

3. At this point, run the **ldconfig** command. This configures the shared library cache and restores links to the various shared libraries. This command is typically run at boot time. However, because Linux has not been rebooted at this time, it is likely that you will experience problems accessing some files.
4. Test that all Linux functions work. For example, ensure that the man pages display correctly.

   Tip: If there is a problem, **umount**, which is designed to *unmount* the previously mounted file system, may also fail. For example, you may see a message similar to this:

   ```
   umount: /usr: device is busy
   ```

   If this occurs, then rebooting the Linux partition will end the mount and the local copy of /usr can be accessed.

5. If the tests are successful, then you are ready to add an entry into the /etc/fstab file. *fstab* stands for File System TABle. It is where the system administrator can tell the Linux operating system about any file systems the machine may have access to. It also allows default parameters to be provided for each file system. Therefore, after rebooting the Linux system, the mount to /dunnas/usr on the OS/400 IFS is automatically re-established.

   For more information on the structure and format of the /etc/fstab entries, see the MAN pages for *fstab*. The following site, /etc/fstab under Linux, is also useful: http://www.humbug.org.au/talks/fstab/fstab.html

   We use the following entry:

   ```
   ASLN1:/dunnas/usr /usr nfs rsize=8192,timeo=14,intr
   ```

6. Reboot the Linux partition and test that all of the functions still work.

Creating a new NWSSTG and installing a minimal Linux installation

Now that you have successfully tested that the Linux system works correctly when mounting the /usr file system stored on the IFS, you are now in a position to replace the initial configuration with a minimal installation of Linux. The original NWSSTG, which contains Linux, plus all of the packages, can eventually be deleted.

Red Hat recommends 300 MB for a Custom-class minimal installation or 650 MB for a server-class minimal installation. However, when creating the NWSSTG, you should allow some extra space for any temporary files to reside.

1. Vary off the NWSD.
2. Remove the Network Server Storage Link using the RMVNWSSTGL command or by selecting option **11** from the WRKNWSSTG display (see Figure 7-6).

```
                    Remove Server Storage Link (RMVNWSSTGL)

Type choices, press Enter.

Network server storage space . . > LINUX1        Name
Network server description . . . > LINUX1        Name
```

Figure 7-6 OS/400 Remove Network Server storage Link command

3. From OS/400, create a new NWSSTG. Use the CRTNWSSTG command. In this example, NWSSIZ is defined as 500 MB. This is to provide extra space to hold any temporary files that may be created (see Figure 7-7).

```
                     Create NWS Storage Space (CRTNWSSTG)

Type choices, press Enter.

Network server storage space . . NWSSTG       > LINUX1MIN
Size . . . . . . . . . . . . . . NWSSIZE      > 500
From storage space . . . . . . . FROMNWSSTG     *NONE
Format . . . . . . . . . . . . . FORMAT       > *OPEN
Auxiliary storage pool ID  . . . ASP            1
Text 'description' . . . . . . . TEXT           NWSSTG for minimal Linux install
```

Figure 7-7 OS/400 Create Network Storage Space command

4. Insert the first Red Hat CD in the drive and change the NWSD for the installation of Red Hat (see Figure 7-8).

```
                    Change Network Server Desc (CHGNWSD)

Type choices, press Enter.

IPL source . . . . . .   *STMF         *SAME, *NWSSTG, *PANEL...
IPL stream file  . . . . . .   '/QOPT/RED_HAT/PPC/ISERIES/VMLINUX'

IPL parameters . . . . . .   *NONE



Text 'description' . . . . . .   'NWSD LINUX1 partition RED HAT 7.1'
```

Figure 7-8 OS/400 CHGNWSD command

5. Add a server storage link to the new NWSSTG. Use the ADDNWSSTGL, or option **10** from the WRKNWSSTG display (see Figure 7-9).

```
                    Add Server Storage Link (ADDNWSSTGL)

Type choices, press Enter.

Network server storage space . . > LINUX1MIN     Name
Network server description . . . > LINUX1        Name
Dynamic storage link . . . . . .   *NO           *NO, *YES
Drive sequence number  . . . . .   *CALC         3-18, *CALC
```

Figure 7-9 OS/400 ADDNWSSTGL command

6. Vary on the NWSD and install a Custom-class *minimal installation* of Red Hat Linux.

Deleting /usr from the Linux system

Once the minimal Linux installation is complete and the NFS mount from directories /dunnas/usr on the IFS onto /usr on Linux has been tested, you can delete /usr on the Linux system.

7. Add an entry for the NFS mount into /etc/fstab, and test it.
8. From Linux, rename the /usr file system:

   ```
   mv /usr /usr1
   ```

9. Create a new “empty” /usr directory that remains as the nfs mount point:

   ```
   mkdir /usr
   ```

10. Delete the renamed file system /usr1:

    ```
    rm -rf /usr1
    ```

    Attention: Be careful when using this command because it is very powerful and does not verify with you that the correct file or directory name has been entered. For details on all of the options, see the man pages for **rm**.

11. Reboot the Linux system. Linux is now mounting /usr from /dunnas/usr from the IFS.

The final step

The configuration for the first Linux partition is now complete. The final step is to configure a custom-class minimal installation of Red Hat Linux on the second OS/400 guest partition.

Perform the same actions, as detailed from steps 3 through to 11 from the previous section. After the final reboot of the second Linux partition, both Linux systems will share the same single copy of /usr, exported from the IFS on OS/400.

7.3 Summary

Linux is a very versatile and flexible system. When installed on an iSeries partition, Linux can further benefit from many features of the hosting OS/400 operating system.

In the above example, we demonstrated one particular Linux administration method. There are potentially numerous alternative configurations. Hosting files from the OS/400 IFS can be especially useful in the following situations:

- Sharing a standard set if files
- Hosting files that may grow in size

A

Linux system message logging

This appendix describes a mechanism for logging Linux system messages in OS/400 using syslog on the Linux partition and a syslogd server program on OS/400. All Linux system messages are logged to a file and select ones go as messages to QSYSOPR. This appendix covers:

- Introduction to syslogd
- Linux system configuration
- OS/400 system configuration

Introduction to syslogd

Syslogd is a Linux daemon responsible for logging system messages on Unix and Linux systems. Processing of the syslogd daemon is controlled by the */etc/syslog.conf* file and detailed information about this configuration file can be found using the following Linux command:

```
man syslog.conf
```

The important part of the man page is the section talking about remote logging. Built in to the syslogd program is the ability to send the system messages to a remote system where it will be logged. This allows you to effectively manage multiple Linux systems from a single system. Logged at the remote system will be a time stamp, originating system, and the original message.

Linux system configuration

The configuration of syslogd is controlled by the /etc/syslog.conf file on your Linux partition. There is extensive help for this configuration in the man pages. The configuration of this file allows an administrator to log certain messages (those coming from certain applications, those of a certain severity, etc.) to files or to reroute those messages to another system running the syslogd job. Normally the remote system is another Linux system, but with the OS/400 syslogd program, those messages can easily be rerouted to an OS/400 system.

Example A-1 shows a copy of the /etc/syslog.conf file from one of our Linux partitions.

Example: A-1 File /etc/syslog.conf

```
# /etc/syslog.conf - Configuration file for syslogd(8)
#
# For info about the format of this file, see "man syslog.conf".
#

#
#
#
*.emerg       *

# enable this, if you want that root is informed
# immediately, e.g. of logins
#*.alert      root


#
# all email-messages in one file
#
#mail.*      -/var/log/mail

#
# all news-messages
#
# these files are rotated and examined by "news.daily"
#news.crit-/var/log/news/news.crit
#news.err -/var/log/news/news.err
news.notice-/var/log/news/news.notice
# enable this, if you want to keep all news messages
# in one file
#news.*      -/var/log/news.all
```

```
#
# Warnings in one file
#
#*.=warn;*.=err-/var/log/warn
#*.crit       /var/log/warn

#
# save the rest in one file
#
#*.*;mail.none;news.none-/var/log/messages

#
# enable this, if you want to keep all messages
# in one file
#*.*         -/var/log/allmessages
*.*          @SYSTEM

#
# Some foreign boot scripts require local7
#
#local0,local1.*-/var/log/localmessages
#local2,local3.*-/var/log/localmessages
#local4,local5.*-/var/log/localmessages
#local6,local7.*-/var/log/localmessages

#kern.*-/var/log/firewall
```

The changes made to this file are:

```
#*.*         -/var/log/allmessages
*.*          @SYSTEM
```

The first line is the original line in the file that we commented out. The second line is the one we added to reroute all messages to the system designated by host name SYSTEM. The “@” character before the host names indicates that the messages will be rerouted to a remote system.

On our Linux system, we also need to add SYSTEM to the */etc/hosts* file. Example A-2 is a copy of the /etc/hosts file from the same Linux partition.

Example: A-2 File /etc/hosts

```
#
# hosts         This file describes a number of hostname-to-address
#               mappings for the TCP/IP subsystem.  It is mostly
#               used at boot time, when no name servers are running.
#               On small systems, this file can be used instead of a
#               "named" name server.
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#

127.0.0.1localhost

# special IPv6 addresses
::1             localhost ipv6-localhost ipv6-loopback

fe00::0         ipv6-localnet
```

```
ff00::0         ipv6-mcastprefix
ff02::1         ipv6-allnodes
ff02::2         ipv6-allrouters
ff02::3         ipv6-allhosts

10.10.5.4SYSTEM
10.10.5.9      linux.local linux
```

OS/400 system configuration

On the OS/400 side, the only configuration needed is to create a program that will act like a remote syslogd job and receive incoming messages. The first step is to determine how syslogd sends messages to remote systems. To do this, we look through the */etc/services* file and find an entry for syslog. That entry looks like this:

```
syslog          514/udp
```

This tells us that syslog uses IP port 514 and a protocol of udp. If you want to view the /etc/services file, we recommend that you pipe the output to a search function such as **grep**, since the /etc/services file is quite large. We use the following command to limit the results to only those lines containing the word “syslog”.

```
cat /etc/services | grep syslog
```

Once we know how syslogd is sending remote messages, it is fairly simple to take the socket server code from 5.4.2, “Server example” on page 75, and modify it to receive syslogd messages. Our initial step is just to change the port number so that we can see what the messages look like. Based on what the messages look like, we then further refine the server code until we achieve the exact behavior that we desire.

OS/400 syslogd server program

The syslogd server program differs from the socket server example in that the syslogd program is expected to only run on OS/400. To that extent, all the #ifdef statements are removed to clarify the code. Also, we need to get additional data such as the current time and TCP/IP host information. Finally, we need to open a file to store the incoming messages and send out selected messages to QSYSOPR. The program flow of the syslogd program is as follows:

1. Open the log file where all the messages are stored. If the file does not exist, it is created; otherwise, the file will be opened in append mode. This ensures that we do not lose previous messages that were logged.
2. Now open the socket descriptor.
3. Bind port 514 to the new socket descriptor. This is the port that syslogd on Linux uses. If it is changed on the Linux side, then the OS/400 server program also needs to be changed.
4. Create a conversion table. All incoming data will come from Linux machines and, therefore, have to be converted from ASCII to EBCDIC.
5. Start the main body of the program by looping until told to stop. Inside the loop, the following steps take place:
   a. Receive incoming requests from the socket.
   b. Convert the data to EBCDIC.
   c. Remove the newline character from the end of the message.
   d. Get the IP system information and add it to the front of the message.

e. Verify whether this message should go to QSYSOPR and send it there if it should.
f. Get the current time stamp.
g. Write the message to the log file.
h. Flush the write buffer every tenth write.

6. Before exiting the program, close the conversion table and socket descriptor.

To create the syslogd program run the following commands in the order shown:

```
CRTMOD MODULE(SYSLOGD/SYSLOGD) SRCSTMF('/itso/syslogd.c') SYSIFCOPT(*IFSIO) TGTCCSID(37)
CRTPGM PGM(SYSLOGD/SYSLOGD) BNDSRVPGM(QSYS/QTQICONV)
```

These commands create a program called SYSLOGD in library SYSLOGD and assume the source for the program is in file /itso/syslogd.c in IFS. You need to adjust the object names to fit your environment. The last step before the program can be run is to create the IFS directory structure. The program creates the file if it does not exist but cannot create the full directory structure that is needed. The directory needed is */var/log*, and you can create it by using the **mkdir** command on OS/400. Once the program is created, you can submit it to run in batch using the following command:

```
SBMJOB CMD(CALL SYSLOGD/SYSLOGD)
```

A batch job is now running and monitoring the 514 port for incoming messages. All incoming messages are logged to file /var/log/messages and any message containing “kernel:” are also sent to the QSYSOPR message queue. This is a good point to mention the other object that is needed for the syslogd program. That other object is a message file with a message description that will be used for the QSYSOPR messages. In our program, the message file is SYSLOGD/LINUX with a message ID of LNX0001. This can be changed to meet your needs as long as the program is also updated.

Message log on OS/400

Figure 7-10 shows the bottom of the /var/log/messages file on one of our OS/400 systems. As you can see from the sample, we are receiving messages from at least two Linux partitions. One partition is known by the IP host name SUSE641, while the other system is not known by any host name and therefore is tracked by its IP address.

```
 Browse : /var/log/messages 79
 Record :    1242   of    1255 by  14           Column :    1    123 by  79
 Control :

....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
Mar  8 14:04:40 SUSE641 <78>/USR/SBIN/CRON[1571]: (root) CMD ( rm -f /var/spool/
Mar  8 14:27:29 10.10.5.4 <6>-- MARK --
Mar  8 14:32:26 SUSE641 <6>-- MARK --
Mar  8 14:39:06 SUSE641 <37>su: (to root) wildt on /dev/pts/1
Mar  8 14:39:06 SUSE641 <87>PAM-unix2[1628]: session started for user root, serv
Mar  8 14:40:17 SUSE641 <87>PAM-unix2[1628]: session finished for user root, ser
Mar  8 14:47:29 10.10.5.4 <6>-- MARK --
Mar  8 14:52:26 SUSE641 <6>-- MARK --
Mar  8 14:58:53 SUSE641 <38>sshd[1655]: Accepted password for ROOT from ::ffff:9
Mar  8 14:59:17 10.10.5.4 <38>sshd[733]: Could not reverse map address 9.5.92.90
Mar  8 14:59:20 10.10.5.4 <38>sshd[733]: Accepted password for ROOT from ::ffff:
Mar  8 15:01:09 10.10.5.4 <4>kernel: read_super_block: can't find a reiserfs fil
Mar  8 15:01:09 10.10.5.4 <4>kernel: read_super_block: can't find a reiserfs fil
Mar  8 15:01:31 10.10.5.4 <4>kernel: reiserfs: checking transaction log (device
 ************End of Data********************

 F3=Exit   F10=Display Hex   F12=Exit  F15=Services  F16=Repeat find
 F19=Left   F20=Right
```

Figure 7-10 Bottom of /var/log/messages on OS/400

QSYSOPR message queue on OS/400

Figure 7-11 shows the QSYSOPR message queue containing Linux messages from the syslogd program.

```
                              Display Messages
                                                          System:    ITSO
Queue . . . . . :   QSYSOPR               Program . . . . :   *DSPMSG
  Library . . . :     QSYS                   Library . . . :
Severity  . . . :   90                     Delivery  . . . :   *HOLD

Type reply (if required), press Enter.
  All jobs at work station QPADEV0005 ended.
  Job not submitted for job schedule entry QEZBKMGFRI number 002466 because
    the entry is held.
  Adapter has inserted or left the ring on line TRNLINE.
  Adapter has inserted or left the ring on line TRNLINE.
  Kernel Message: 10.10.5.4 <4>kernel: read_super_block: can't find a
    reiserfs filesystem on (dev
  Kernel Message: 10.10.5.4 <4>kernel: read_super_block: can't find a
    reiserfs filesystem on (dev
  Kernel Message: 10.10.5.4 <4>kernel: reiserfs: checking transaction log
    (device 03:41) ...
  Kernel Message: 10.10.5.4 <4>kernel: Using r5 hash to sort names
  Kernel Message: 10.10.5.4 <4>kernel: ReiserFS version 3.6.25
                                                                      Bottom
F3=Exit              F11=Remove a message                F12=Cancel
F13=Remove all       F16=Remove all except unanswered    F24=More keys
```

Figure 7-11 QSYSOPR message queue with Linux messages

Syslogd code

Example A-3 is the actual syslogd code running on OS/400.

Example: A-3 Syslogd code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#include <iconv.h>
#include <QMHSNDM.H>

#define SERVER_PORT 514
#define BACK_LOG 20
#define BUFFER_SIZE 512

typedef struct {
  char name[8];
  char ccsid[5];
  char conversion[3];
  char substitute[1];
  char shift[1];
  char input[1];
  char option[1];
```

```
  char RESERVED[12];
} fromCode_t;

typedef struct {
  char name[8];
  char ccsid[5];
  char RESERVED[19];
} toCode_t;

FILE * openFile(void);
iconv_t createConversionTable(void);
void getDotted(char *dottedName, long ipAddress);
void logMessage(char *message);

int main(int argc, char *argv[])
{
  int rc,
      bufferLength,
      recordCount = 0;
  char buffer[BUFFER_SIZE],
       converted[BUFFER_SIZE],
       message[BUFFER_SIZE],
       *inBuffer,
       *convertBuffer,
       dotted[15];
  FILE *logFile;
  time_t currentTime;

  /*! socket() variables */
  int descriptor,
      addressFamily,
      type,
      protocol;

  /*! bind() variables */
  struct sockaddr_in localAddress;
  int addressLength;

  /*! recvfrom() variables */
  int flags,
      clientLength;
  struct sockaddr_in clientAddress;

  /*! gethostbyaddr() variables */
  struct hostent *system;

  /*! iconv() variable */
  size_t inBufferLength,
         convertBufferLength;
  iconv_t convertTable;

  /*! socket() */
  /*! bind() */
  /*! recvfrom() */
  /*! close() */

  /*! open the file */
  logFile = openFile();

  /*! open the socket descriptor */
```

```
addressFamily = AF_INET;
type = SOCK_DGRAM;
protocol = IPPROTO_IP;
if ((descriptor = socket(addressFamily, type, protocol)) < 0) {
  printf("Error getting socket descriptor\n");
  printf("errno is %i\n", errno);
  exit(-1);
}

/*! bind to the socket */
memset(&localAddress, 0x00, sizeof(localAddress));
localAddress.sin_family = AF_INET;
localAddress.sin_port   = htons(SERVER_PORT);
localAddress.sin_addr.s_addr = htonl(INADDR_ANY);
addressLength = sizeof(localAddress);
if ((rc = bind(descriptor, (struct sockaddr *)&localAddress, addressLength)) < 0) {
  printf("Error binding to the socket\n");
  printf("errno is %i\n", errno);
  close(descriptor);
  exit(-1);
}

/*! setup the conversion table ONLY ON OS/400 SERVER*/
convertTable = createConversionTable();

do {
  /*! receive any incoming client requests */
  memset(buffer, 0x00, sizeof(buffer));
  bufferLength = sizeof(buffer);
  flags = 0;
  clientLength = sizeof(clientAddress);
  rc = recvfrom(descriptor, buffer, bufferLength, flags,
                (struct sockaddr *)&clientAddress, &clientLength);
  if (rc < 0) {
    printf("Error receiving incoming client reqeusts\n");
    printf("errno is %i\n", errno);
    close(descriptor);
    exit(-1);
  }

  /*! convert the data */
  convertBufferLength = inBufferLength = strlen(buffer);
  inBuffer = buffer;
  convertBuffer = converted;
  memset(converted, 0x00, sizeof(converted));
  iconv(convertTable, &inBuffer, &inBufferLength, &convertBuffer, &convertBufferLength);

  /*! remove the newline from the end of the string */
  converted[strlen(converted) - 1] = '\0';

  /*! get the system information and add to the message*/
  system = gethostbyaddr((char *)&clientAddress.sin_addr, 4, AF_INET);
  if (NULL == system) {
    getDotted(dotted, clientAddress.sin_addr.s_addr);
    sprintf(message, "%.15s %.512s", dotted, converted);
  } else {
    sprintf(message, "%s %.512s", system->h_name, converted);
  }

  /*! check if this message needs to go to QSYSOPR */
```

```
    if (NULL != strstr(converted, "kernel:")) {
      logMessage(message);
    }

    /*! get the current time */
    currentTime = time(&currentTime);

    /*! output the data */
    fprintf(logFile, "%.15s %.512s\n", (ctime(&currentTime)) + 4, message);

    /*! flush the write every 10th write */
    ++recordCount;
    if (recordCount > 10) {
      recordCount = 0;
      fflush(logFile);
    }

    /*! stop when 'STOP' is received as the first 4 characters */
  } while (strncmp(converted, "STOP", 4) != 0);

  /*! close the conversion table */
  iconv_close(convertTable);

  /*! close the socket */
  close(descriptor);

  exit(0);
} /*! ...end of main() */

/*!
 * ********
 * openFile
 * ********
 *
 * Open the output log file.
 *
 * Returns: FILE *
 *
 */
FILE *openFile()
{
  FILE *temp;

  temp = fopen("/var/log/messages", "a, ccsid=37");
  fclose(temp);
  temp = fopen("/var/log/messages", "a");
  return temp;
}

/*!
 * *********************
 * createConversionTable
 * *********************
 *
 * Creates the conversion table for the iconv() API.
 *
 * Returns: iconv_t
 *
 */
iconv_t createConversionTable()
```

```
{
  size_t inBufferLength,
         convertBufferLength;
  fromCode_t fromCCSID;
  toCode_t toCCSID;
  iconv_t convertTable;

  memcpy(toCCSID.name, "IBMCCSID", 8);
  memcpy(toCCSID.ccsid, "00037", 5);
  memset(toCCSID.RESERVED, 0x00, 19);
  memcpy(fromCCSID.name, "IBMCCSID", 8);
  memcpy(fromCCSID.ccsid, "01252", 5);
  memcpy(fromCCSID.conversion, "000", 3);
  memcpy(fromCCSID.substitute, "0", 1);
  memcpy(fromCCSID.shift, "0", 1);
  memcpy(fromCCSID.input, "0", 1);
  memcpy(fromCCSID.option, "0", 1);
  memset(fromCCSID.RESERVED, 0x00, 12);
  convertTable = iconv_open((char *)&toCCSID, (char *)&fromCCSID);
  if (convertTable.return_value < 0) {
    printf("Error creating conversion table\n");
    printf("errno is %i\n", errno);
    exit(-1);
  }
  return convertTable;
}

/*!
 * *********
 * getDotted
 * *********
 *
 * Gets a string of the dotted IP address.
 *
 * Returns: void
 *
 */
void getDotted(char *dottedName, long ipAddress)
{
  long address = ipAddress;
  int fourth, third, second, first;

  fourth = address % 256;
  address /= 256;

  third = address % 256;
  address /= 256;

  second = address % 256;
  address /= 256;

  first = address % 256;

  sprintf(dottedName, "%i.%i.%i.%i", first, second, third, fourth);
}


/*!
 * **********
 * logMessage
```

```
 * **********
 *
 * Log the message to the QSYSOPR message queue.
 *
 * Returns: void
 *
 */
void logMessage(char *message)
{
  char messageId[] = "LNX0001",
       messageFile[] = "LINUX     SYSLOGD   ",
       messageData[592],
       messageType[] = "*INFO     ",
       messageQueue[] = "*SYSOPR   ",
       replyQueue[] = "                    ",
       messageKey[4];
  int lengthOfMessage = sizeof(messageData),
      numberOfQueues = 1,
      errorCode = 0,
      firstLevelLength,
      secondLevelLength;

  /*! set the message data */
  firstLevelLength = (strlen(message) < 80) ? strlen(message) : 80;
  secondLevelLength = (strlen(message) < 512) ? strlen(message) : 512;
  memset(messageData, ' ', sizeof(messageData));
  memcpy(messageData, message, firstLevelLength);
  memcpy(messageData + 80, message, secondLevelLength);

  /*! send the message */
  QMHSNDM(messageId, messageFile, messageData, lengthOfMessage, messageType, messageQueue,
          numberOfQueues, replyQueue, messageKey, (void *)&errorCode);
}
```

B

Squid: Customized Linux Solution

This appendix describes how a Customized Linux Solution like Squid can be built on the iSeries. By a Customized Linux Solution, we mean that user should be able to use a service running on Linux partition without seeing a Linux prompt.

From an OS/400 user point of view, they have to:

- Create LPAR partition and Network Server Description (NWSD)
- Restore the Linux partition from the tape or CD as a network storage space and link it to NWSD you just created
- From OS/400, pass the TCP/IP information like IP address, default route, DNS servers to the NWSD
- Vary on the NWSD

This makes it possible to access Internet through a Squid proxy running on Linux without typing any Linux commands.

What is Squid and why use Squid

Squid is a free, open source Internet-proxy caching program. Squid acts like an agent accepting connections from clients (browsers) and then passes it on to the appropriate servers. Before returning the data back to the clients, it stores a copy on an on-disk cache. Next time it gets a request to serve the same *object*, Squid just returns the data from its cache thus saving Internet access and bandwidth. The saved data are called objects because the data need not be always webpages. The data may consist of images, video files, or audio files.

The Squid proxy server can then be accessed from a browser by making changes in the proxy settings:

- The IP address of Linux partition
- Port number 3128

Squid is a very sophisticated service that offers a lot of flexibility to configure it depending upon the needs. For more information on Squid and its features, see:
http://www.squid-cache.org/

To learn about the benefits of using Squid, see:
http://squid-docs.sourceforge.net/latest/html/c28.htm#AEN56

Implementation details

1. Create an LPAR partition and NWSD with LAN card and install Linux.
2. Save the Linux partition from OS/400.
3. Restore the Linux partition on a different or same iSeries.
4. Pass the TCP/IP information to NWSD.
5. Vary on NWSD.

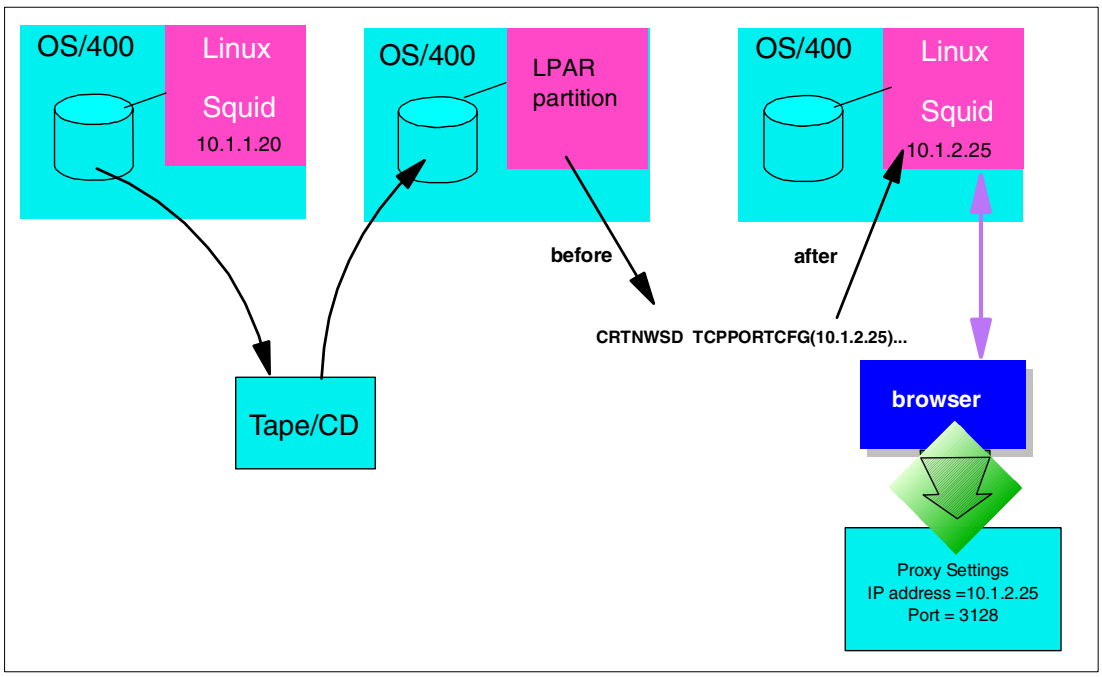

Figure B-1 Overview of Customized Linux Solution - Squid

Creating LPAR partition and NWSD with LAN card and installing Linux

You have to set up a logical partition and a NWSD with a native LAN card. In this case, we want to install the minimum Linux system less than 640 MB so that it can fit on a single CD if needed.

We assume that you know how to set up LPAR and NWSD and to install a Linux partition. It is better if the filesystem type of the Linux partition is *ReiserFS*. ReiserFS is an advanced journaling type of filesystem that is suitable for Squid as ReiserFS is very efficient in dealing with large numbers of small files that Squid generates. You can read about ReiserFS at the Web site: http://www.namesys.com

Note: ReiserFS filesystem is a good option to use but other filesystem types like *ext2* and *ext3* can also be used.

iSeries rpm installation

You should install a iSeries Linux rpm package for your Linux distribution. You can get it (source?????)

This package gets the TCP/IP networking information from /proc/iSeries/config file and makes appropriate changes in the Linux system network files during boot time.

If you have stored this package in /var, then:

```
linux:~ # cd /var
linux:/var # rpm -Uvh iSeries-netcfg-suse-1.0-1.ppc.rpm
iSeries-netcfg-suse         ##############################################
linux:/var #
```

Sometimes after installing this rpm and rebooting Linux, services like Apache and Squid are not able to start because they are unable to resolve the hostname to an IP address. If this happens, then you have to add the IP address and hostname in /etc/hosts. You can perform the following step to automate the task.

In the /etc/rc.d/iSeries-ifcfg script, add the following lines in the start() function at the end just before the last **fi**:

```
if [ -z $(cat /etc/hosts | grep $(hostname) | awk '{print $2}') ]; then
                echo "$IPADDR_0 $(hostname)" >>/etc/hosts;
fi
```

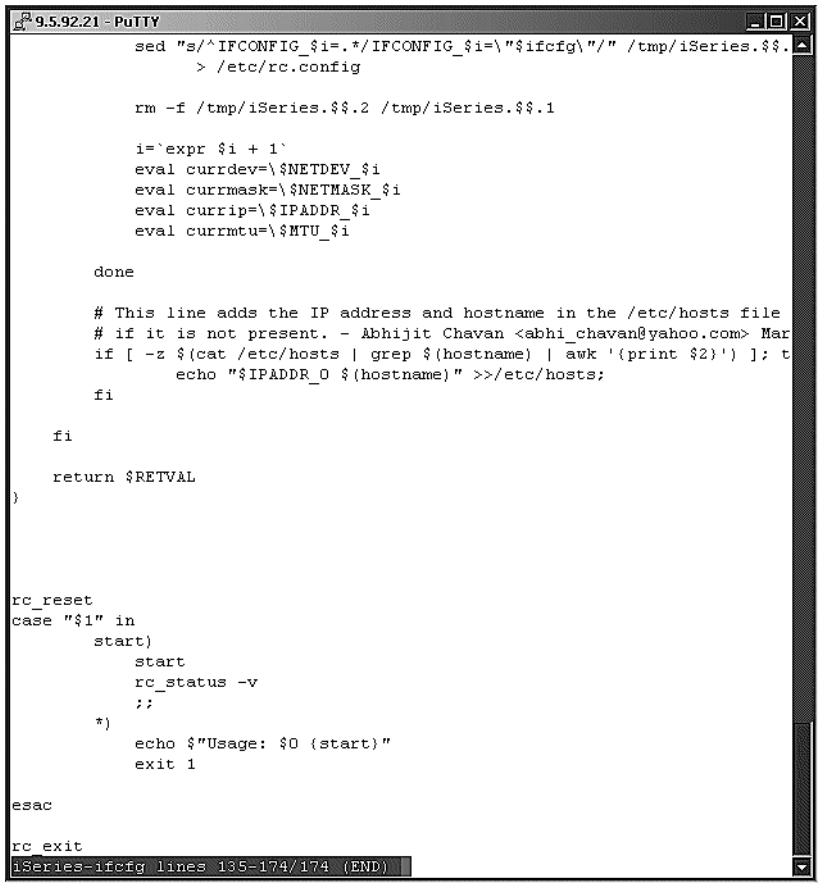

Figure B-2 A line to automatically add the IP address and hostname in the /etc/hosts file

Squid setup

You can install Squid from the rpms in your Linux distribution CDs. For example, the Squid rpm package is in CD number two of SuSE 64bit Enterprise Server edition. From the Linux prompt, you can run:

```
linux:~ # mount /media/cdrom
linux:~ # find /media/cdrom -name “squid*”
./full-names/ppc/squid-2.3.STABLE4-63.ppc.rpm
./full-names/ppc/squidgrd-1.1.4-131.ppc.rpm
./full-names/src/squid-2.3.STABLE4-63.src.rpm
./full-names/src/squidgrd-1.1.4-131.src.rpm
./suse/contents/squid
./suse/contents/squidgrd
./suse/n2/squid.rpm
./suse/n3/squidgrd.rpm
./suse/zq1/squid.spm
./suse/zq1/squidgrd.spm
linux:~ # cd /media/cdrom/full-names/ppc
linux:/media/cdrom/full-names/ppc # rpm -Uvh squid-2.3.STABLE4-63.ppc.rpm
squid                       ##################################################
Updating etc/rc.config...
linux:/media/cdrom/full-names/ppc #
```

For other Linux distributions, follow similar steps after mounting the appropriate CD.

You can install Squid by compiling the source code. Visit the Squid Web site and download the source from: http://www.squid-cache.org/

Then follow the steps mentioned in the README and INSTALL files.

Important: Please ensure that Squid is started every time the Linux system boots or IPLs. In the /etc/rc.config file, you should set the variable START_SQUID="yes".

Squid configuration file

The default settings in Squid do not allow any client to connect to Squid. You have to modify the /etc/squid.conf file.

Note: The ACL entry and http_access entry are not one after another. These entries are to be made in the appropriate blocks of squid.conf file. The ACL entry is in the "ACCESS CONTROLS" section and the http_access section is generally the next section.

Example: B-1

```
#defines a group (or Access Control List) that includes all IP
#addresses
acl all src 0.0.0.0/0.0.0.0
#allow all sites to use connect to us via HTTP
http_access allow all
```

Change the "src 0.0.0.0/0.0.0.0" to the network address and subnet mask of your network or add another *ACL* entry so that no one outside your network is able to access Squid.

You may also need to change the *cache_mem* default value of 8 MB. *cache_mem* is mainly used to store the *In-Transit* objects in memory, for example, the objects that are accessed in recent time. This needs to be set according to your system resources and needs. For Linux on iSeries, if you have say 64 MB RAM, then you can set *cache_mem* as 32 MB.

The cache directory (*cache_dir*) and its size are also set in the squid.conf file. Ensure that the user *squid* can write to the cache directory. The default size of cache directory is 100 MB. You may want to change this depending upon the disk space allocated to your Linux partition, but since we want the disk space to be less than 640 MB, you should keep it to 100 MB.

One thing that can be done is to attach a secondary hard disk and mount it on the squid cache directory. In this way, you can allocate as much disk space as the iSeries allows. But having large caches (up to 20 GB) does not necessarily mean high number of hits as webpages become stale too soon and are deleted.

Whenever you change the squid.conf file, you should restart Squid. For SuSE, enter:

```
linux:~ # /etc/rc.d/squid restart
```

To verify whether Squid is running, you can run:

```
linux:~ # ps -auwwwx | grep [s]quid
root      2797  0.0  0.2  2284 1072 tty1     S    15:18   0:00 less /etc/squid.conf
root      2951  0.0  0.2  4664 1220 ?        S    16:50   0:00 /usr/sbin/squid -sYD
squid     2952  0.0  1.0  9116 5048 ?        S    16:50   0:00 (squid) -sYD
squid     2953  0.0  0.0  1460  440 ?        S    16:50   0:00 (unlinkd)
```

If you see similar output, then squid is running.

You can now access the Internet via Squid from a browser by changing the proxy settings.

For Netscape

1. Go to **Edit-> Preferences**. The Preferences window appears (Figure B-3).

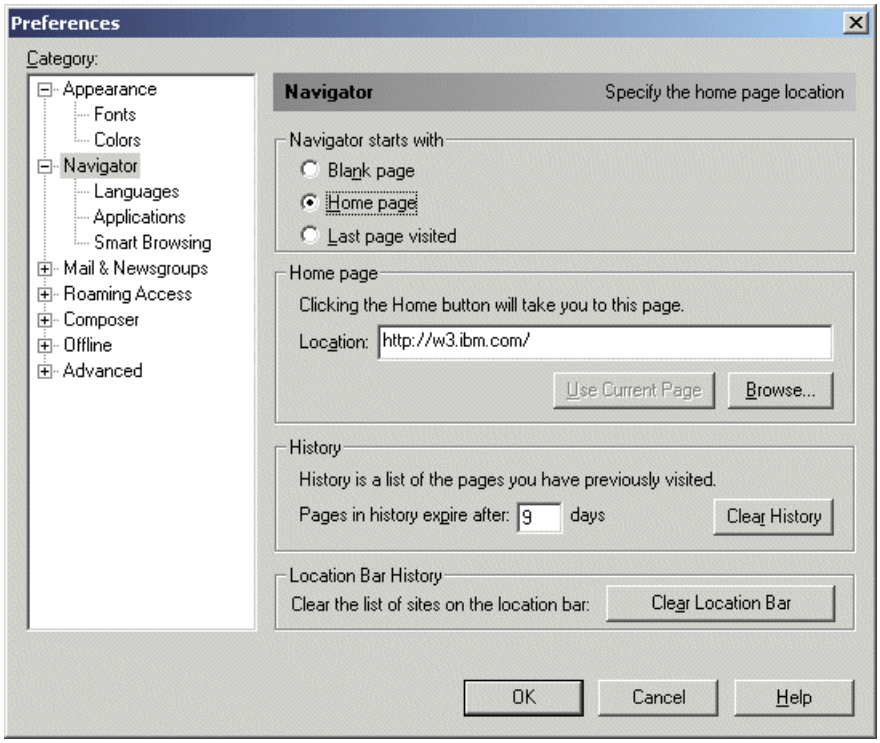

Figure B-3 Netscape Preferences

2. Click **Advanced-> Proxies**. See Figure B-4.

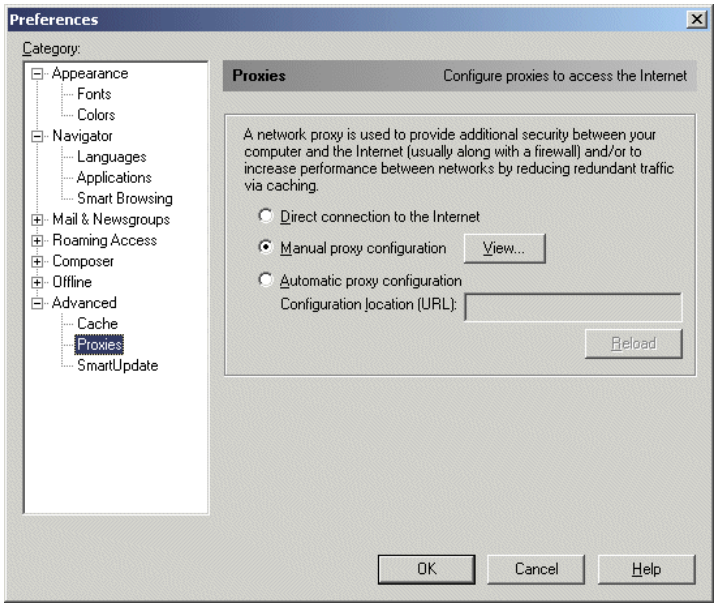

Figure B-4 Netscape Proxies option

3. Select **Manual proxy configuration** and **View**.

4. You should type the Linux partition's IP address and port number 3128 in the *HTTP*, *Security*, *FTP* and *Gopher* fields.

   Here 10.1.1.20 is the IP address of Linux system on which Squid is running.

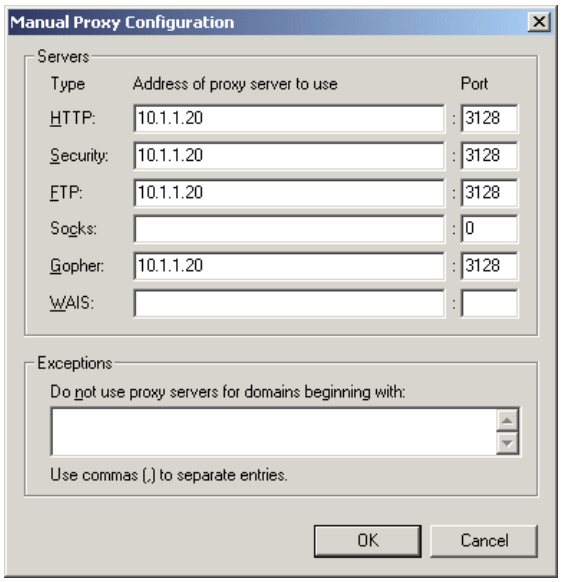

Figure B-5 Manual proxy settings in Netscape

For Internet Explorer

1. Go to Internet Explorer. Select **Tools -> Internet Options**.

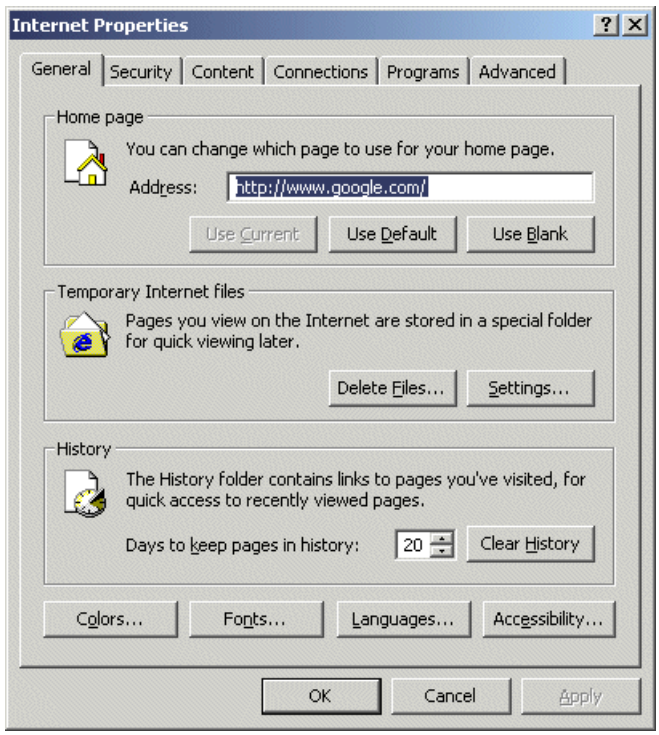

Figure B-6 Internet Explorer Properties

2. Select the **Connections** tab and click **LAN Settings**.
3. Type in Linux partition's IP and port 3128 as a *proxy server*.

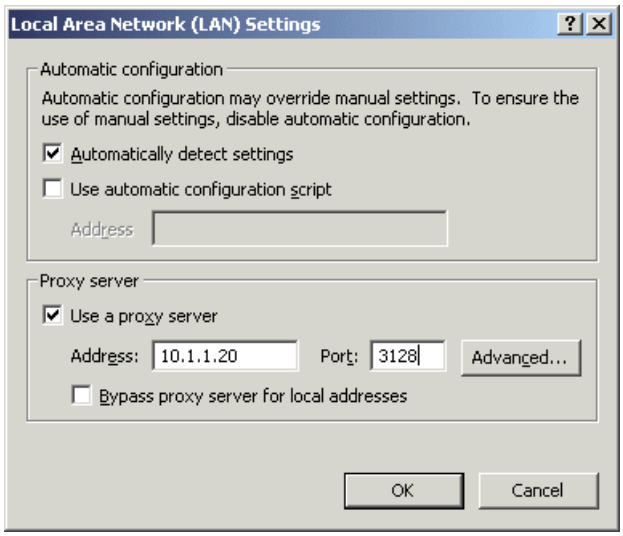

Figure B-7 Internet Explorer's LAN settings

After you change the browser to the required values, you can use Squid to access the Internet.

Saving the Linux partition from OS/400

1. Vary off the Linux partition from OS/400. On the OS/400 command line, type the `WRKCFGSTS` command. Then type **2** in front of the partition to vary off.
2. Save the partition:
   - If you want to save the Linux partition on a CD, then save it to a save file and then ftp it to a PC. You can then burn the data into a CD.
   - If you want to save it on a tape then, from OS/400 command line, type `SAV` and press F4. You then see the display shown in Figure B-8.

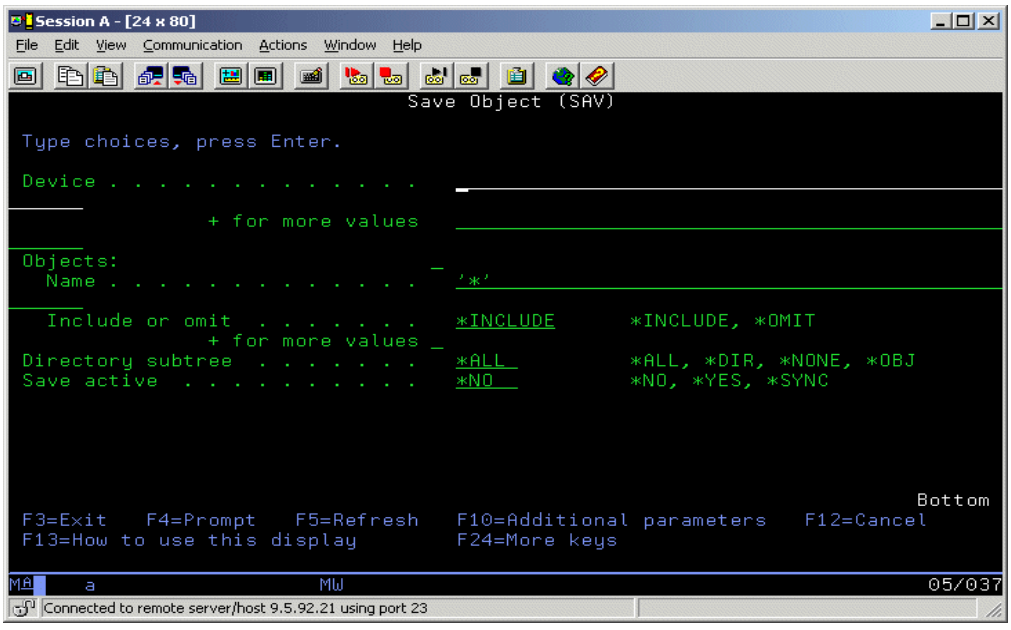

Figure B-8 Save Object display after SAV F4

3. Type in the Device file path. In case you don't know which tape is used you can run the `WRKDEVD` command and find out the tape number.

   Device should be `/qsys.lib/tap02.devd`

   Object name should be `/qfpnwsstg/susemin`

   Here *susemin* is the network storage space that we have to save.

Note: If you are using a new tape, then you should first initialize the tape by running the `INZTAP` and passing the appropriate parameters.

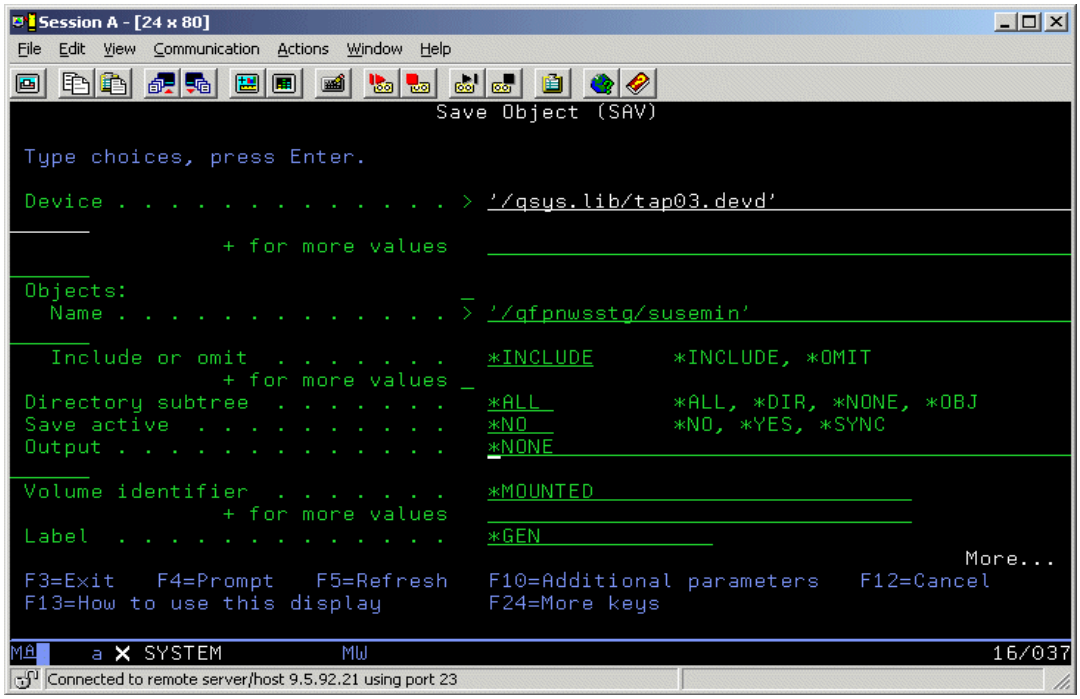

Figure B-9 OS/400 after starting the process to save the NWSSTG on tape

When the process is over, you see the Work with Configuration Status display in Figure B-10.

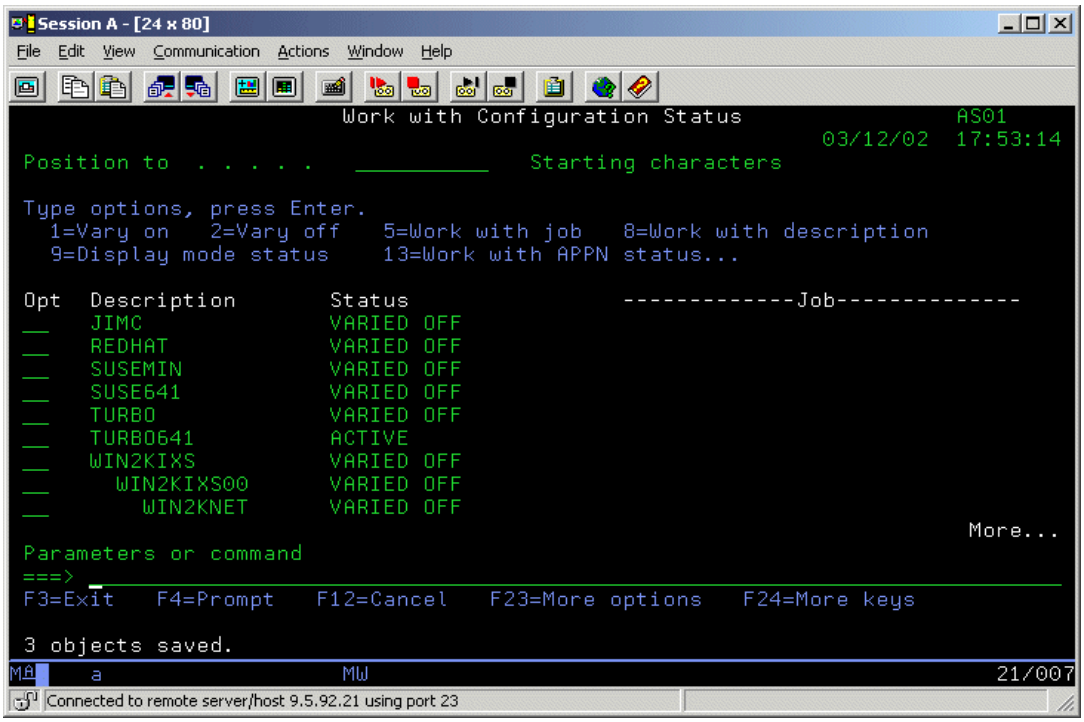

Figure B-10 Linux partition saved on tape

Restoring the Linux partition on the same or different iSeries servers

1. Place the tape or CD in the appropriate I/O device on the iSeries server for which you want to restore the Linux partition.
2. On the OS/400 command line, type the RST command and press F4. Then you see the Restore Object display as shown in Figure B-11.

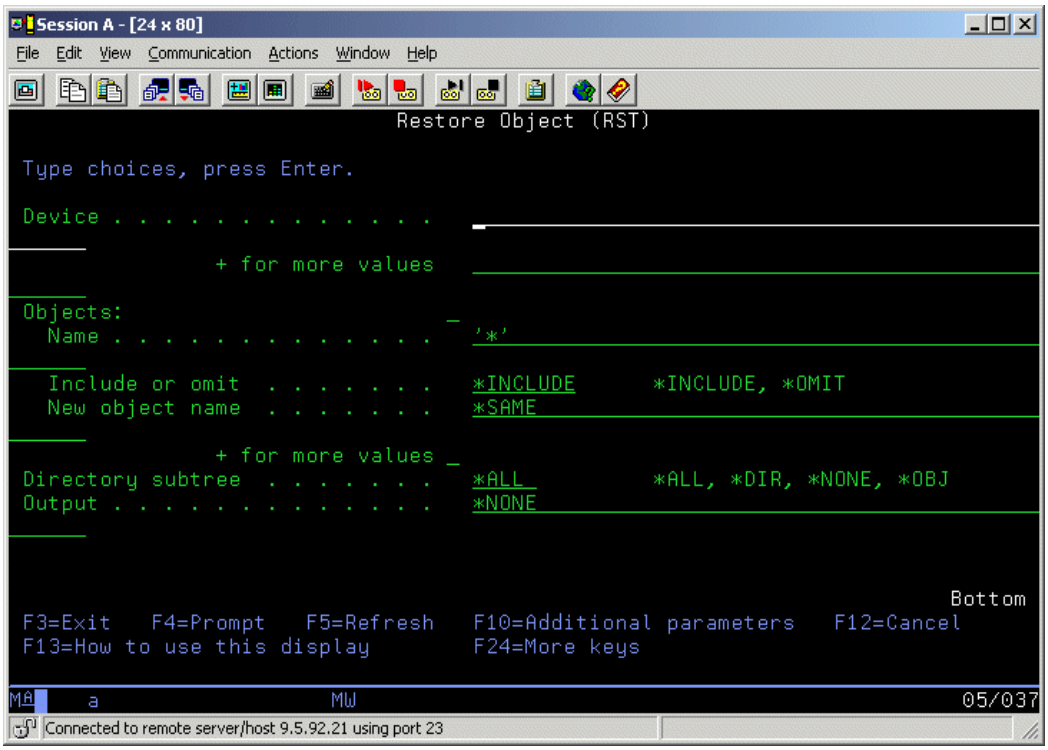

Figure B-11 Starting to restore from tape/CD

3. In the Device field, type `/qsys.lib/tap03.devd` for tape or type `/qsys.lib/opt01.devd` for CD.

   Attention: If you saved on a tape, then if applicable, enter the appropriate sequence number to select the proper NWSSTG.

   The Object should be /qfpnwsstg/susemin. susemin is the NWSD name and should be changed to your NWSD's name.

   Then after the restoration, you see the Work with Configuration Status display as shown in Figure B-12.

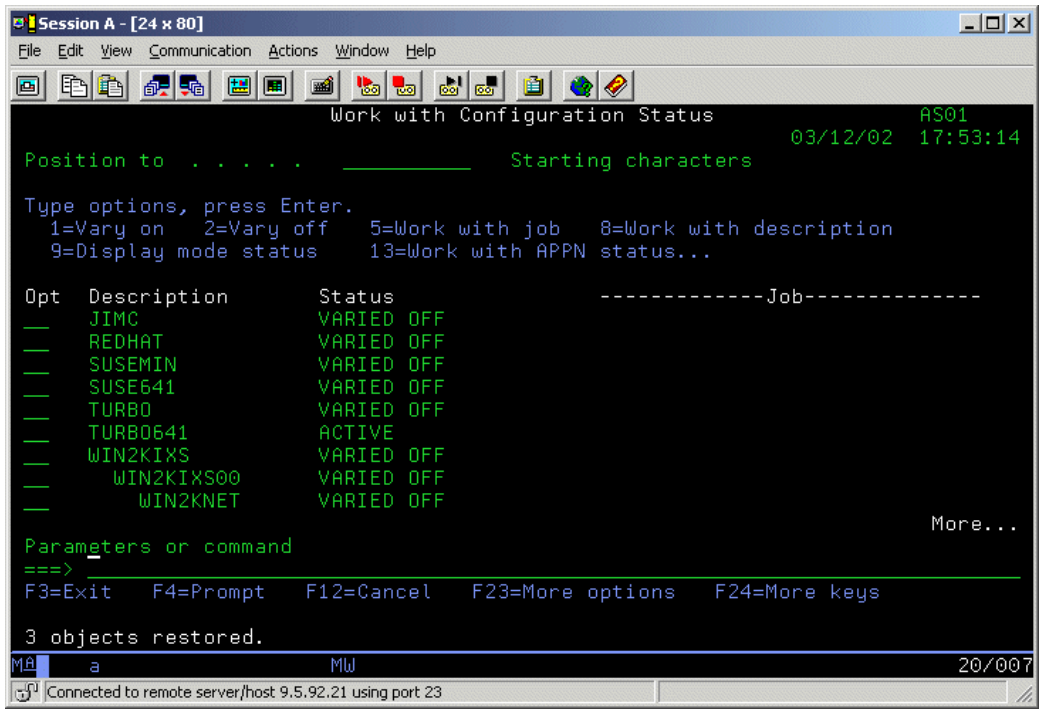

Figure B-12 Objects restored

Passing the TCP/IP information to NWSD

1. To configure TCP/IP information for the Linux partition, open a command line on the hosting OS/400 partition.
2. Type the `WRKNWSD` command to work with network server descriptions, and type **2** in front of the partition for which you want to configure TCP. Press F9 to view all available options.

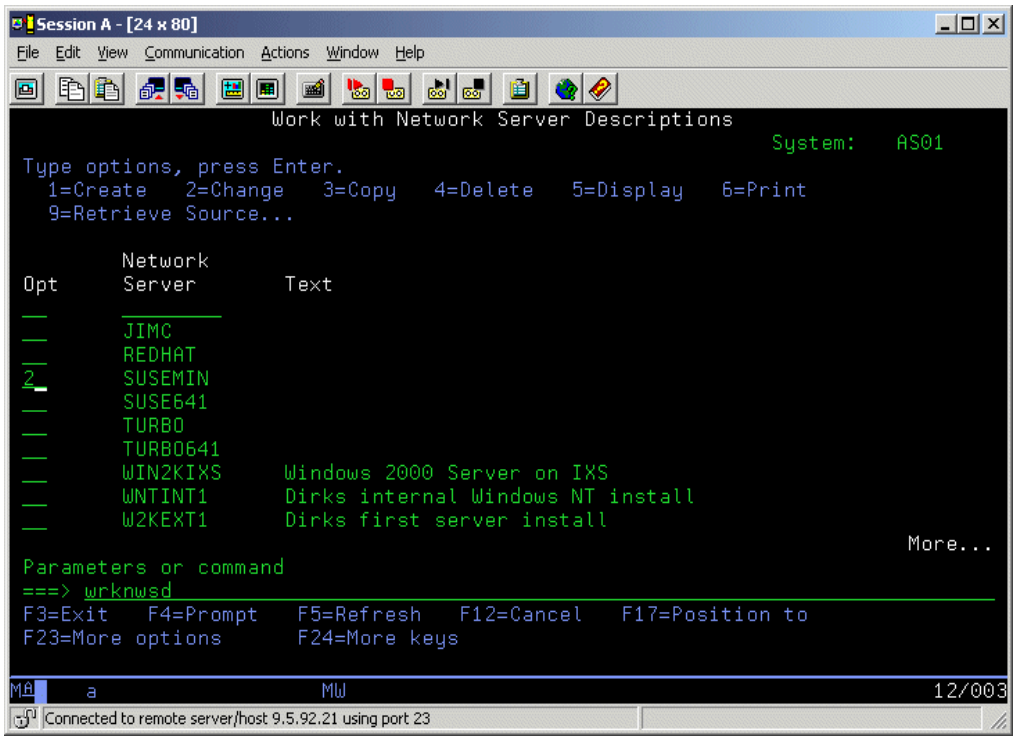

Figure B-13 Working with the NWSD

3. Press Enter.

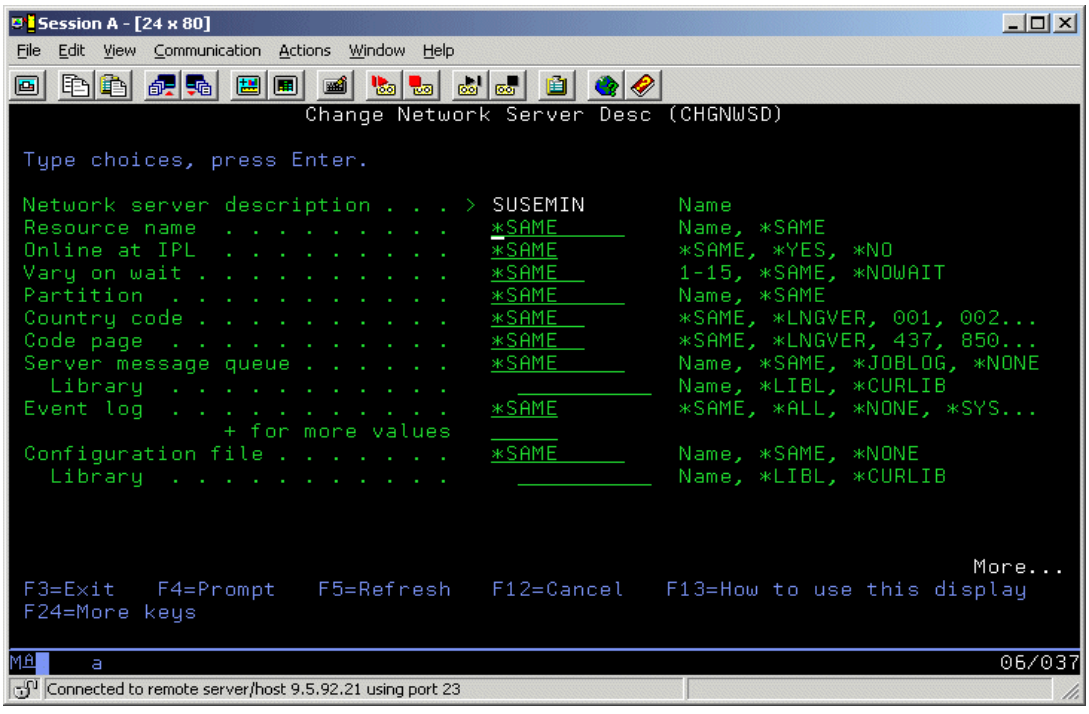

Figure B-14 Changing the NWSD

4. The first relevant section, *TCP/IP port configuration*, allows you to configure various network devices for the partition. Valid options for *Port* are numbers 1 through 3.

The following table describes which network devices are associated with each port number.

Table B-1

Network device	Port	Name in Linux
Virtual LAN	1	veth0
Native Ethernet	2	eth0
Native Token Ring	3	tr0

You can only configure one of each type of device for a partition. Additional network devices, such as vethX, ethX, trX (where X is greater than 0), cannot be configured in this way.

Internet address, *Subnet mask*, and *Maximum transmission unit* refer to standard TCP/IP network configuration information for the device. To configure more than one device, enter a + in the final field of the section.

TCP/IP route configuration is where different routes may be configured. Enter the destination network, subnet mask, and the next hop (or gateway) for the route. To change the default route, enter ***DFTROUTE** as the Route destination. To configure more than one route, enter a + in the final field of the section.

TCP/IP local host name and local domain name are simply the internet name for the partition.

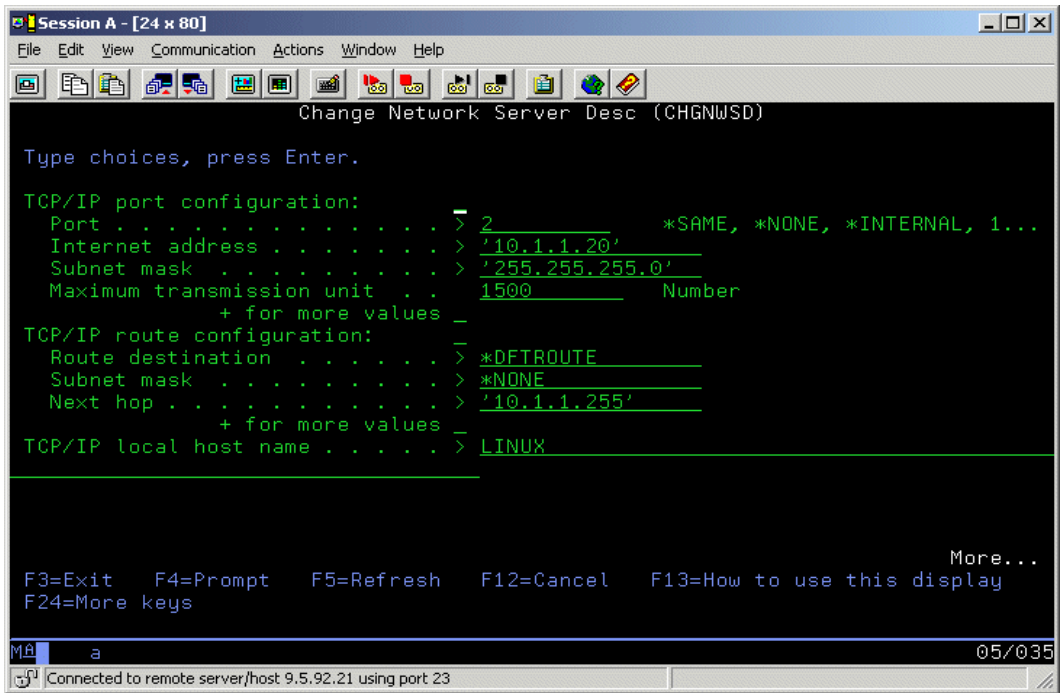

Figure B-15 TCP/IP information for NWSD

5. Press Page Down to arrive at the next screen.

   TCP/IP name server system is where you can enter a list of domain name servers (DNS servers) in order of priority. Simply enter the IP address of the DNS server or servers.

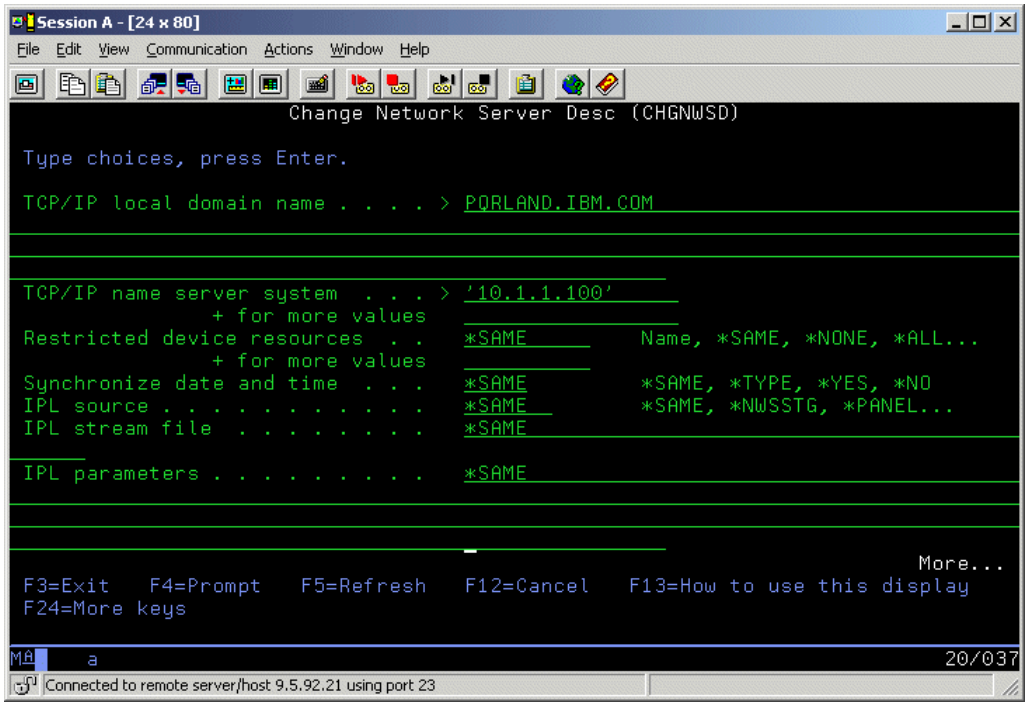

Figure B-16 TCP/IP information for NWSD

To enter more than one, enter a + in the field below *TCP/IP name server system*.

After the configuration is satisfactory, press Enter to save the settings.

Varying on the Linux partition

In order for the Linux partition to recognize the changes, it must be varied off and on. You can do this by typing the following command on the OS/400 command line:

```
WRKCFGSTS *NWS
```

Then enter **2** and then **1** in front of the appropriate partition. When the partition reboots, the changes take effect.

To verify that the configuration is correct, you can use the following commands in Linux:

- To verify that network devices are configured correctly, enter:

  ```
  $ /sbin/ifconfig
  ```

- To verify that network routes are configured correctly, enter:

  ```
  $ /sbin/route -n
  ```

- To verify that host/domain name are configured correctly, enter:

  ```
  $ hostname
  ```

Now you should access the Internet with browser settings as described in “For Netscape” on page 116 and “For Internet Explorer” on page 117. This time you should put in the IP address of the Linux partition you have just restored.

If all works out well, you now access the Internet via a Squid proxy on a Linux partition.

C

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG24????

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG24????.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
????????.zip	????Zipped Code Samples????
????????.zip	????Zipped HTML Documents????
????????.zip	????Zipped Presentations????

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	????MB minimum????
Operating System:	????Windows/UNIX????
Processor:	???? or higher????
Memory:	????MB????

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

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 129.

- *????full title???????,* xxxx-xxxx
- *????full title???????,* xxxx-xxxx
- *????full title???????,* xxxx-xxxx

Other resources

These publications are also relevant as further information sources:

- *????full title???????,* xxxx-xxxx
- *????full title???????,* xxxx-xxxx
- *????full title???????,* xxxx-xxxx

Referenced Web sites

These Web sites are also relevant as further information sources:

- Description1

  `http://????????.???.???/`
- Description2

  `http://????????.???.???/`
- Description3

  `http://????????.???.???/`

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

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, as well as updates and formats.

Index

Linux Integration with OS/400 on the IBM @server iSeries Server

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5” spine. Now select the Spine width for the book and hide the others: Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set

Linux Integration with OS/400

on the IBM @server iSeries Server

Learn the advantages of integrating Linux with OS/400

Access DB2 UDB for iSeries data from Linux applications

Install Linux for the intranet using NetServer

The IBM @server iSeries server offers many points of integration that support Linux applications leveraging OS/400 applications and data. This includes iSeries ODBC Driver for Linux, OS/400 NetServer with Samba support, IBM Java Virtual Machine, and JTOpen support.

iSeries ODBC Driver for Linux provides access from a Linux application running on iSeries to the DB2 UDB for iSeries database. OS/400 NetServer supports the exchange of files between OS/400 and Linux, and Linux applications can print to OS/400 print queues. IBM Java Virtual Machine is available for iSeries Linux. This JVM in combination with the iSeries Toolbox for Java or its OpenSource version, JTOpen, allows developers to access DB2 UDB for iSeries data via JDBC as well as leverage OS/400 programs and services.

This IBM Redbook covers each integration technique with sample code. It also shows you the methods that you can use to integrate Linux applications with OS/400. This redbook is intended to help beginner and intermediate Linux users, with an OS/400 background, to integrate the Linux application with OS/400 application and data.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6551-00 ISBN 0738425389