SUSE Linux Enterprise Server 10 Advanced Administration

COURSE 3073

Novell Training Services

www.novell.com

AUTHORIZED COURSEWARE

Part # 100-005062-001
Version 1

Novell®

Contents

SECTION 3 Configure and Use Samba

SECTION 4 Enable Fundamental Network Services

SECTION 5 Create Shell Scripts

SECTION 7 Perform a Health Check and Performance Tuning

SECTION 8 Manage Hardware

Introduction

In the *SUSE Linux Enterprise Server 10 Advanced Administration* (3073) course you learn the SUSE Linux Enterprise Server 10 administration skills necessary to complete your basic SUSE Linux Enterprise Server 10 skill set.

These skills, along with those taught in the *SUSE Linux Enterprise Server 10 Fundamentals* (3071) and *SUSE Linux Enterprise Server 10 Administration* (3072) courses, prepare you to take the Novell® Certified Linux® Professional (Novell CLP) certification practicum test.

Your student kit includes the following:

- *SUSE Linux Enterprise Server 10 Advanced Administration* Manual
- *SUSE Linux Enterprise Server 10 Advanced Administration* Workbook
- *SUSE Linux Enterprise Server 10 Advanced Administration* Course DVD
- *SUSE Linux Enterprise Server 10* Product DVD
- *SUSE Linux Enterprise Desktop 10* Product DVD

The *Course DVD* contains a preinstalled VMware image of SUSE Linux Enterprise Server 10 that you can use with the *SUSE Linux Enterprise Server 10 Advanced Administration Workbook* to practice the skills you need to take the Novell CLP practicum.

Instructions for setting up a self-study environment are in the setup directory on the Course DVD.

Course Objectives

This course teaches you how to perform the following SUSE Linux administrative tasks for SUSE Linux Enterprise Server 10:

1. Manage Virtualization with XEN
2. Configure a Web Application Server
3. Configure Samba
4. Enable Fundamental Network Services
5. Create Shell Scripts
6. Compile Software from Source
7. Perform a Health Check and Performance Tuning
8. Manage Hardware

These are tasks common to an experienced SUSE Linux administrator in an enterprise environment.

The final day of class is reserved for a “LiveFire” exercise that provides a set of scenarios to test your SUSE Linux Enterprise Server 10 administration skills and prepare you to take the Novell CLP 10 Practicum.

Audience

This course is designed for those who already have experience with Linux, including general system configuration and command line work. This course is also ideal for those seeking advanced administration skills on SUSE Linux Enterprise Server 10, those who have completed the previous two courses in the Novell CLP10 curriculum and those preparing to take the Novell CLP10 Practicum exam.

Certification and Prerequisites

This course helps you prepare for the Novell Certified Linux Professional 10 (Novell CLP 10) Practical Test, called a *practicum*. The Novell CLP 10 is an entry-level certification for people interested in becoming SUSE Linux administrators.

As with all Novell certifications, course work is recommended. To achieve the certification, you are required to pass the Novell CLP 10 Practicum (050-697).

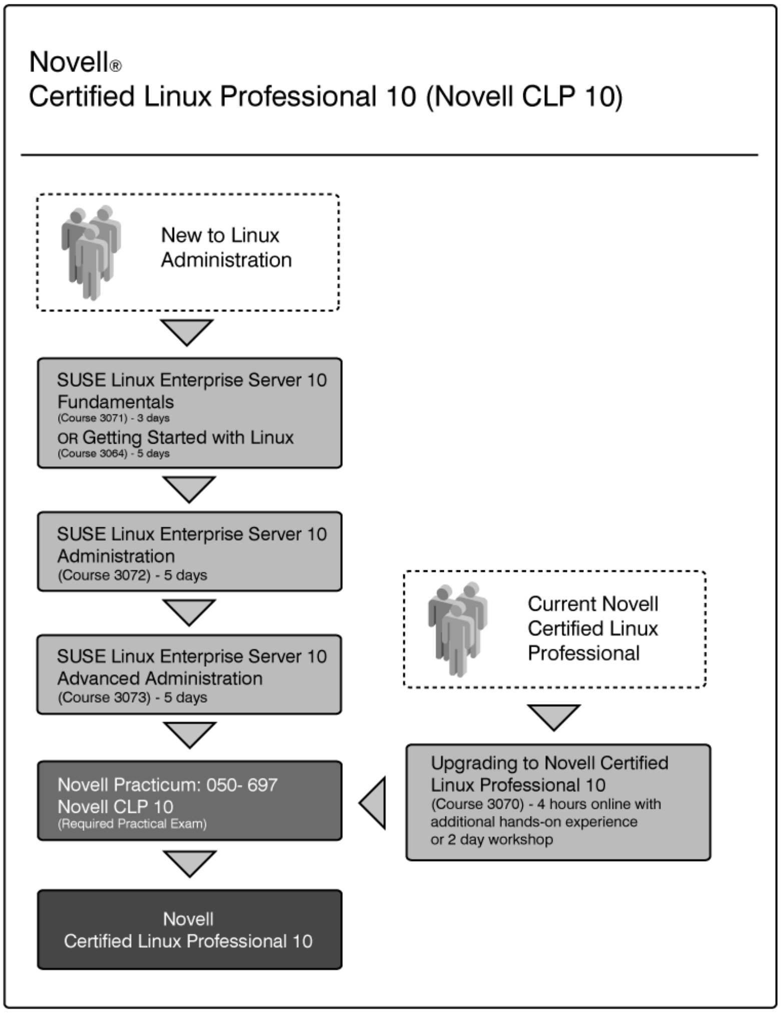
The Novell CLP 10 Practicum is a hands-on, scenario-based exam where you apply the knowledge you have learned to solve real-life problems—demonstrating that you know what to do and how to do it.

The practicum tests you on objectives in this course (*SUSE Linux Enterprise Server 10 Advanced Administration* - Course 3073) and the skills outlined in the following Novell CLP 10 courses:

- *SUSE Linux Enterprise Server 10 Fundamentals* - Course 3071
- *SUSE Linux Enterprise Server 10 Administration* - Course 3072

The following illustrates the training/testing path for Novell CLP 10:

Figure Intro-1


Novell®
Certified Linux Professional 10 (Novell CLP 10)
New to Linux Administration
SUSE Linux Enterprise Server 10 Fundamentals (Course 3071) - 3 days
OR Getting Started with Linux (Course 3064) - 5 days
SUSE Linux Enterprise Server 10 Administration (Course 3072) - 5 days
SUSE Linux Enterprise Server 10 Advanced Administration (Course 3073) - 5 days
Novell Practicum: 050- 697 Novell CLP 10 (Required Practical Exam)
Novell Certified Linux Professional 10
Current Novell Certified Linux Professional
Upgrading to Novell Certified Linux Professional 10 (Course 3070) - 4 hours online with additional hands-on experience or 2 day workshop

This course is designed for those who have intermediate level knowledge of Linux. They should be able to:

- Understand what Linux is and know about the Open Source concept
- Perform a basic installation of SUSE Linux Enterprise Server 10
- Perform a basic system configuration including network setup
- Manage software packages
- Work on the command line including file management and text editing.

This knowledge can also be gained though the *SUSE Linux Enterprise Server 10 Fundamentals* (Course 3071) and S*USE Linux Enterprise Server 10 Administration* (Course 3072).

For more information about Novell certification programs and taking the Novell CLP Practicum, see http://www.novell.com/education/certinfo.

SUSE Linux Enterprise Server 10 Support and Maintenance

The copy of SUSE Linux Enterprise Server 10 you receive in your student kit is a fully functioning copy of the SUSE Linux Enterprise Server 10 product.

However, to receive official support and maintenance updates, you need to do one of the following:

- Register for a free registration/serial code that provides you with 30 days of support and maintenance.
- Purchase a copy of SUSE Linux Enterprise Server 10 from Novell (or an authorized dealer).

You can obtain your free 30-day support and maintenance code at http://www.novell.com/products/linuxenterpriseserver/eval.html.

You will need to have or create a Novell login account to access the 30-day evaluation.

Novell Customer Center

Novell Customer Center is an intuitive, web-based interface that helps you to manage your business and technical interactions with Novell. Novell Customer Center consolidates access to information, tools and services such as:

- Automated registration for new SUSE Linux Enterprise products
- Patches and updates for all shipping Linux products from Novell
- Order history for all Novell products, subscriptions and services
- Entitlement visibility for new SUSE Linux Enterprise products
- Linux subscription-renewal status
- Subscription renewals via partners or Novell

For example, a company might have an administrator who needs to download SUSE Linux Enterprise software updates, a purchaser who wants to review the order history and an IT manager who has to reconcile licensing. With Novell Customer Center, the company can meet all these needs in one location and can give each user access rights appropriate to their roles.

You can access the Novell Customer Center at http://www.novell.com/center.

SUSE Linux Enterprise Server 10 Online Resources

Novell provides a variety of online resources to help you configure and implement SUSE Linux Enterprise Server 10.

These include the following:

- http://www.novell.com/products/linuxenterpriseserver/

  This is the Novell home page for SUSE Linux Enterprise Server 10.

- http://www.novell.com/documentation/sles10/index.html

  This is the Novell Documentation web site for SUSE Linux Enterprise Server 10.

- http://support.novell.com/linux/

  This is the home page for all Novell Linux support, and includes links to support options such as Knowledgebase, downloads, and FAQs.

- http://www.novell.com/coolsolutions

  This Novell web site provides the latest implementation guidelines and suggestions from Novell on a variety of products, including SUSE Linux.

Agenda

The following is the agenda for this 5-day course:

Table Intro-1

	Section	Duration
Day 1	Introduction	00:45
	Section 1: Manage Virtualization with XEN	04:00
	Section 2: Configure a Web Application Server	01:45
Day 2	**Section 2:** Configure a Web Application Server (continued)	3:30
	Section 3: Configure Samba	03:00
Day 3	**Section 4:** Enable Fundamental Network Services	02:30
	Section 5: Create Shell Scripts	04:00
Day 4	**Section 5:** Create Shell Scripts (continued)	02:00
	Section 6: Compile Software from Source	01:30
	Section 7: Perform a Health Check and Performance Tuning	03:00
Day 5	**Section 8:** Manage Hardware and Component Changes	02:00
	Section 9: Prepare for the Novell CLP Practicum	04:30

Scenario

The Digital Airlines management has made the decision to migrate several back-end services to Linux servers running SUSE Linux Enterprise Server 10. The services should run on one host in separate virtual machines.

You have already installed SUSE Linux Enterprise Server 10 before and are familiar with administering SUSE Linux Enterprise Server 10 from YaST and from the command line.

To be able to implement the migration plan, you need additional experience in the following areas:

- Virtualization with Xen.
- System settings on the configuration file level
- Configuration of important services from the command line
- Creating basic shell scripts and compiling software from source packages

You decide to set up a test server in the lab to enhance your skills in these areas.

Exercise Conventions

When working through an exercise, you will see conventions that indicate information you need to enter that is specific to your server.

The following describes the most common conventions:

- ***italicized/bolded text***. This is a reference to your unique situation, such as the host name of your server.

  For example, if the host name of your server is DA50, and you see the following

  ***hostname*.digitalairlines.com**

  you would enter

  DA50.digitalairlines.com

- **10.0.0.*xx***. This is the IP address that is assigned to your SUSE Linux Enterprise Server 10.

  For example, if your IP address is 10.0.0.50, and you see the following

  10.0.0.*xx*

  you would enter

  10.0.0.50

- **Select.** The word *select* is used in exercise steps to indicate a variety of actions including clicking a button on the interface and selecting a menu item.

- **Enter and Type.** The words *enter* and *type* have distinct meanings.

  The word *enter* means to type text in a field or at a command line and press the Enter key when necessary. The word *type* means to type text without pressing the Enter key.

  If you are directed to type a value, make sure you do not press the Enter key or you might activate a process that you are not ready to start.

SECTION 1 Manage Virtualization with Xen

In this section you learn about the Xen virtualization technology in SUSE Linux Enterprise Server 10.

Objectives

1. Understand the Concept of Virtualization
2. Understand How Xen Works
3. Install Xen
4. Manage Xen Domains with YaST
5. Manage Xen Domains at the Command Line
6. Understand Xen Networking
7. Migrate a Guest Domain

Introduction

Virtualization is one of the hottest topics in the industrie at the moment. However, the idea of virtualization is not new at all. Hardware platforms like IBMs pSeries or zSeries support virtualization since a long time and software like VMware Workstation for x86 based systems has been available for many years.

Now virtualization moves to mainstream, because affordable Intel or AMD based x86 systems, provide enough resources to run more than one virtual machine at the same time.

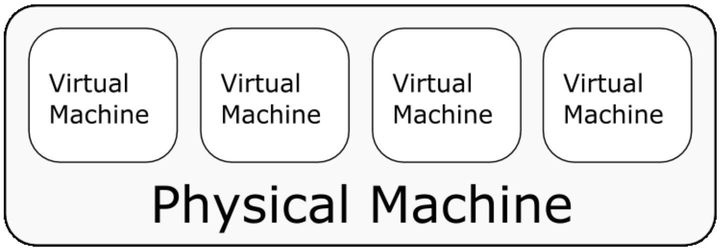
SUSE Linux Enterprise Server 10 comes with build-in virtualization support through the Xen virtual machine monitor. In the following section you'll learn how to use this powerful feature.

Objective 1 Understand the Concept of Virtualization

Virtualization technology separates a running instance of an operating system from the physical hardware. Instead of a physical machine, the operating system runs in a so-called virtual machine. Multiple virtual machines share the resources of the underlying hardware.

Virtualization allows you to run multiple virtual systems on one single physical machine.

Figure 1-1



The following are the main advantages of virtualization, in comparison with non-virtualized physical hardware:

1. **Efficient Hardware Utilization**. Often systems are not using the full potential of their hardware. By running multiple virtual machines on the same hardware, the resources are used more efficiently.
2. **Reduced Downtime**. Virtual machines can be easily migrated to to a new physical host system. This reduces the downtime in case of a hardware failure.
3. **Flexible Resource Allocation.** Hardware resources can be allocated on demand. When the resource requirements of a virtual machine change, resource allocation can be adjusted or the machine can be migrated to a different physical host.

Objective 2 Understand How Xen Works

The idea of virtualization is not new. Platforms like IBM zSeries or pSeries offer built-in virtualization and Intel x86 based systems can be virtualized using third-party software like VMware.

SUSE Linux Enterprise Server 10 comes with a virtualization technology called Xen, which allows you to run multiple virtual machines on a single piece of Intel x86 based hardware.

At the moment, the operating systems that run in a Xen virtual machine need to be modified. Therefore only open source operating systems like Linux or BSD can be installed. One exception is Netware, which has been adjusted by Novell to run in a Xen virtual machine.

Intel and AMD are developing extensions (Intel Vanderpool and AMD Pacifica) to the x86 Standard to support virtualization. Once these extensions are available, Xen will be able to run unmodified operating systems including Microsoft Windows.

You can find updated information about Xen, including an instruction how to run unmodified operating systems on the OpenSUSE Xen page at: **http://en.opensuse.org/Xen**

To understand how Xen works, you need to do the following:

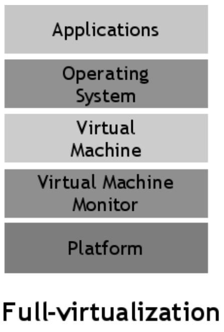
- Understand Virtualization Methods
- Understand the Xen Architecture

Understand Virtualization Methods

Before we talk in detail about the Xen technology, you should understand the following two different virtualization methods.

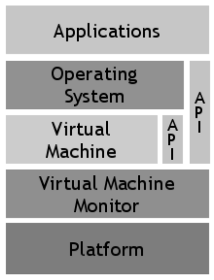
- **Full Virtualization.** In this case the virtualization software emulates a full virtual machine including all hardware resources. The operating system running in the virtual machine (guest OS) communicates with these resources as if they were physical hardware. VMware Workstation is a popular full virtualization software.

Figure 1-2



- **Para Virtualization.** Instead of emulating a full virtual machine, para-virtualization software provides an Application Programming Interface (API) which is used by the guest OS to access hardware resources. This requires that the guest OS is aware that it runs in a virtual machine and needs to know how to access the API. Xen is a para-virtualization software.

Figure 1-3



Para-virtualization

Para virtualization provides better performance because it does not emulate all hardware details. The drawback is, that the guest OS needs to be modified to run with para-virtualization. Full-virtualization works with an unmodified guest OS but generates more overhead resulting in a weaker performance.

Another advantage of para-virtualization is the fexible resource allocation. As the guest OS is aware of the virtual environment, Xen can, for example, change the memory allocation of a virtual machine on the fly without any reboot.

Understand the Xen Architecture

Xen consists of the following two major components:

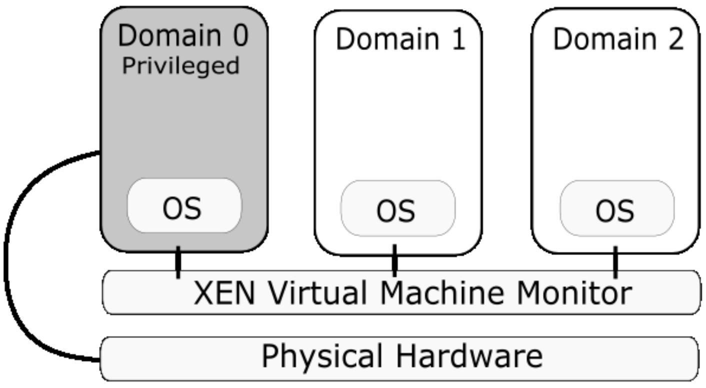
- **Virtual Machine Monitor.** The virtual machine monitor forms a layer between physical hardware and virtual machines. In general this kind of software is called a **Hypervisor**.
- **Xen tools.** The Xen tools are a set of command line applications that are used to administer virtual machines.

The virtual machine monitor must be loaded before any of the virtual machines are started. When working with Xen, virtual machines are called **domains**.

The Xen virtual machine monitor neither includes any drivers to access the physical hardware of the host machine nor an interface to communicate directly with an administrator. These tasks are performed by an operating system running in the privileged **domain0**.

The following is an overview of a Xen system with three domains.

Figure 1-4



A process called **xend** runs in the domain0 Linux installation. This process is used to manage all Xen domains running on a system and to provide access to their consoles.

A unprivileged domain is also called domainU in the Xen terminology.

SUSE Linux Enterprise Server 10 can be used for privileged (domain0) and unprivileged (domainU) Xen domains.

Objective 3 Install Xen

To setup a Xen system, your start from a normal SUSE Linux Enterprise 10 installation, which is going to run in domain0.

The other Xen domains can later be installed in physical partitions or file system images. When you plan to use physical partitions, you have to make sure that the initial SUSE Linux Enterprise Server 10 installation is not using all of the available disc space.

For maximum flexibility it makes sense to use the logical volume manager LVM or EVMS for a Xen system.

The following packages have to be installed in the initial SUSE Linux Enterprise Server 10 installation:

- **xen.** This package contains the Xen virtual machine monitor (Hypervisor).
- **xen-tools.** Contains xend and a collection of command line tools to administer a Xen system.
- **kernel-xen.** This package contains a modified Linux kernel that runs in a Xen domain.
- **xen-doc-*** (optional)**.** Xen documentation in various formats.

The installation of the Xen package automatically adds an entry like the following into the bootloader configuration file **/boot/grub/menu.lst**.

```
title Xen
root (hd0,3)
kernel /boot/xen.gz
module /boot/vmlinuz-Xen root=/dev/hda3 selinux=0
module /boot/initrd-Xen
```

On some Xen systems you might see the parameter **dom0_mem** in the kernel module line. This parameter assigns a certain amount of initial main memory to domain0 at boot time. However in Xen version 3, this parameter is not required anymore.

Initially all available memory is used by domain0. When you start additional domainUs, the required amount of memory is reduced in domain0 and used for the new domainU.

The entry in menu.lst adds a new option to the boot menu of your system. When selecting this entry, the Xen virtual machine monitor is loaded (**kernel /boot/xen.gz**) which starts SUSE Linux Enterprise Server 10 in domain0 (see the lines starting with **module**).

Before rebooting your system with the Xen option, you should check if the automatically generated entry is correct. Make sure that ...

- ... the line **root (hd0,3)** points to the filesystem which contains the Xen Virtual Machine Monitor and the Kernel of the Linux installation for domain0. In our example **hd0,3** means the fourth partition on the first hard drive in the system. Also check if the parameter **root** in the first module line points to the root partition of the domain0 installation.
- ... the Xen version of the Linux kernel and the initrd are loaded in the module line. The names of the image files should end in **-xen**.

After checking the bootloader configuration file, you can reboot your system and select the Xen option at the bootloader menu. In the early stages of the boot process, you will see some messages of the Xen virtual machine monitor on the screen. Then the domain0 Linux installation is started.

In case the system is not booting properly, you can switch back to a non-virtualized system by selecting the regular SUSE Linux Enterprise Server 10 boot option.

When running Xen, the network setup is done by the xend management process. This can interfere with the native network configuration scripts of the domains. Especially SuSEfirewall2 is known to cause problems. It´s therefore recommended to stop SuSEfirewall2 with **rcSuSEfirewall2** and to remove the firewall scripts from the init process:

insserv -r SuSEfirewall2_setup
insserv -r SuSEfirewall2_init
insserv -r SuSEfirewall2_final (conditional)

Exercise 1-1 Install Xen

In this exercise, you learn how to install Xen and configure domain0.

You can find this exercise in the workbook.

(End of Exercise)

Objective 4 Manage Xen Domains with YaST

After you have installed Xen and the Xen tools, you can start to create more Xen domains. Before we go into the details of the domain configuration, we will introduce the YaST module **Virtual Machine Management (Xen).**

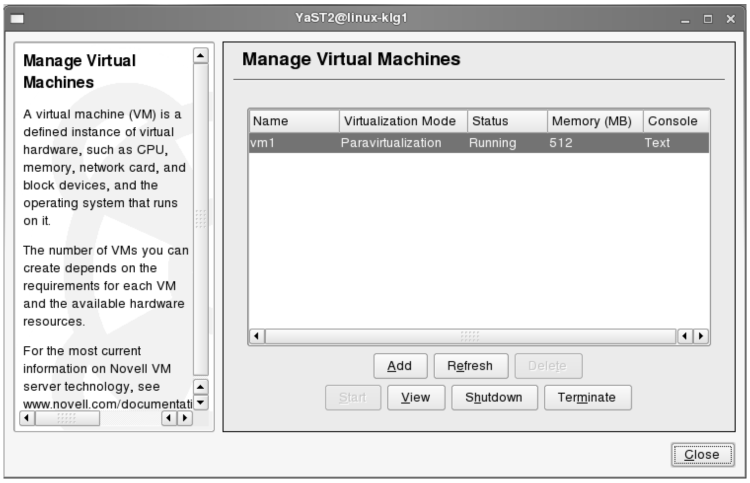
This module provides a convenient way to create and control the Xen domains on your system. The module can be started from the **System** section in the YaST Control Center, and has to run on the Linux system running in domain0.

Not every detail of the Xen domain configuration is described in the following. More in-depth information follow in the next objective.

The following is a step by step description of how to create and boot a new Xen domain with this YaST module.

After you have started the module, the following dialog appears on the screen:

Figure 1-5



YaST2@linux-klg1

Manage Virtual Machines

A virtual machine (VM) is a defined instance of virtual hardware, such as CPU, memory, network card, and block devices, and the operating system that runs on it.

The number of VMs you can create depends on the requirements for each VM and the available hardware resources.

For the most current information on Novell VM server technology, see www.novell.com/documentati

Manage Virtual Machines

Name	Virtualization Mode	Status	Memory (MB)	Console
vm1	Paravirtualization	Running	512	Text

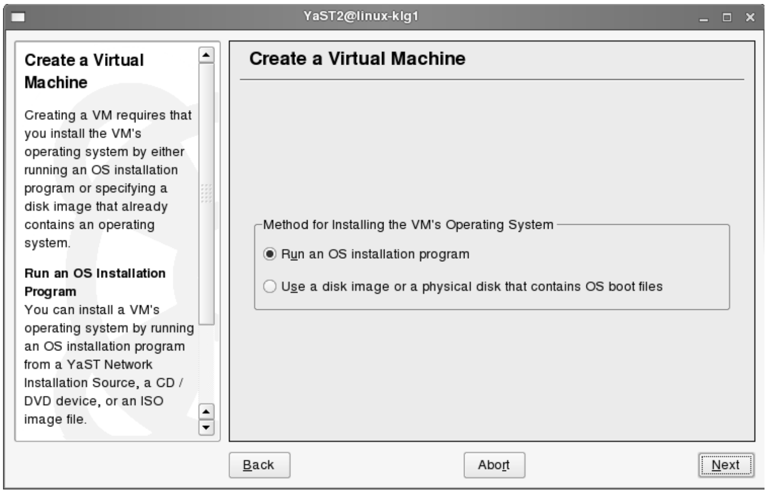
In our example there is already one guest domain running on the system, which is listed in the upper part of the dialog. The columns of the table display various information about the domain including the name, the status and the memory allocation.

The following buttons are in the lower part of the dialog:

- **Add**. Select this button to create a new domain.
- **Refresh**. This button refreshes the information about the domains.
- **Delete.** Deletes a domain completely.
- **Start**. Starts a domain.
- **View**. Opens a terminal window to access the console of a domain.
- **Shutdown**. Performs a regular shutdown of the guest OS.
- **Terminate**. Terminates the domain immediately without waiting for the guest OS to shutdown.

After selecting **Add**, the following dialog appears:

Figure 1-6



Create a Virtual Machine

Creating a VM requires that you install the VM's operating system by either running an OS installation program or specifying a disk image that already contains an operating system.

Run an OS Installation Program
You can install a VM's operating system by running an OS installation program from a YaST Network Installation Source, a CD / DVD device, or an ISO image file.

Create a Virtual Machine

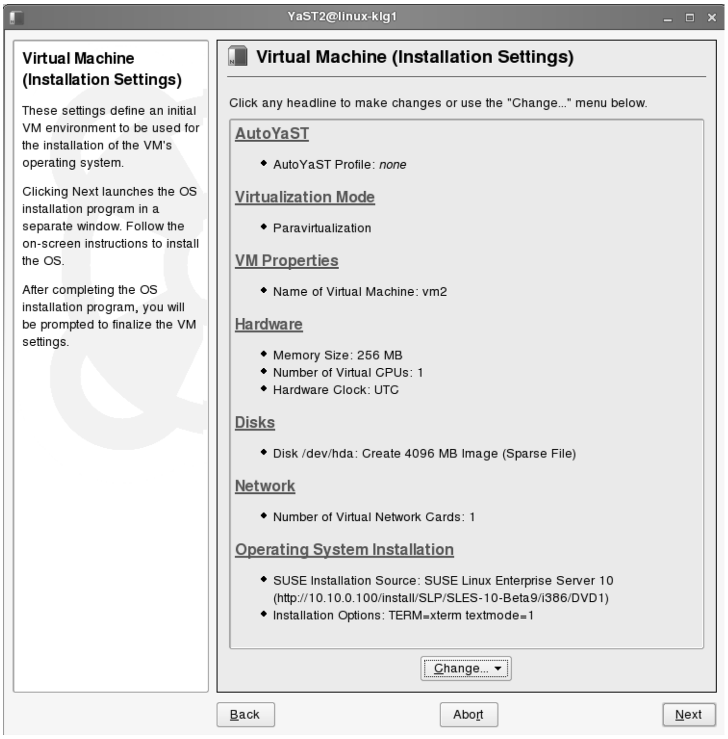
Method for Installing the VM's Operating System
[● Run an OS installation program]
○ Use a disk image or a physical disk that contains OS boot files

The dialog gives you two choices:

- **Run an OS installation program**. This allows you to run a SUSE Linux Enterprise Server installation from an installation source that is registered in the system.
- **Use a disk image or a physical disk that contains OS boot files.** This option lets you create a Xen domain from an existing installation in a physical disc or disc image.

For the following example we select the **Run an OS installation program** option. The following dialog appears:

Figure 1-7



Virtual Machine (Installation Settings)

These settings define an initial VM environment to be used for the installation of the VM's operating system.

Clicking Next launches the OS installation program in a separate window. Follow the on-screen instructions to install the OS.

After completing the OS installation program, you will be prompted to finalize the VM settings.

Virtual Machine (Installation Settings)

Click any headline to make changes or use the "Change..." menu below.

AutoYaST
- AutoYaST Profile: *none*

Virtualization Mode
- Paravirtualization

VM Properties
- Name of Virtual Machine: vm2

Hardware
- Memory Size: 256 MB
- Number of Virtual CPUs: 1
- Hardware Clock: UTC

Disks
- Disk /dev/hda: Create 4096 MB Image (Sparse File)

Network
- Number of Virtual Network Cards: 1

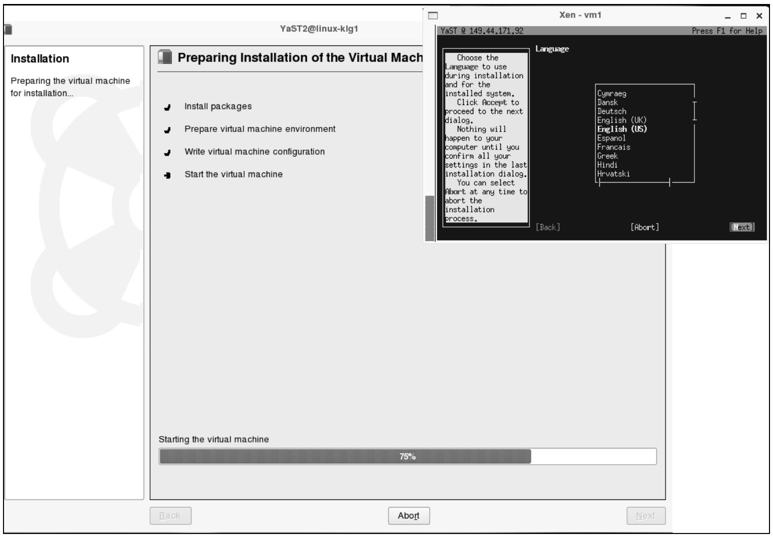
Operating System Installation
- SUSE Installation Source: SUSE Linux Enterprise Server 10 (http://10.10.0.100/install/SLP/SLES-10-Beta9/i386/DVD1)
- Installation Options: TERM=xterm textmode=1

The following options can be adjusted by selecting their headlines:

- **AutoYaST.** In this option you can specify an AutoYaST profile that should be used for the installation. When there is no AutoYaST profile, a manual installation is started.
- **Virtualization.** You can switch between para virtualization and full virtualization. Full-virtualization is only available on supported hardware with Intel or AMD virtualization extension.
- **VM Properties.** Here you can change the name of the new domain.
- **Hardware.** In this option you can configure the hardware configuration of the domain. (Memory, Number of CPUs, ...)
- **Disks.** Configure the Disks here. These can either be physical block devices or file system / disc images.
- **Network.** This option lets you add additional network adapters to the domain.
- **Operating System Installation.** Here you can configure the installation source and additional installation options.

For our example we stay with the default and select **Next**. Now the domain environment and the installation process is started.

Figure 1-8



YaST2@linux-klg1

Installation

Preparing the virtual machine for installation...

Preparing Installation of the Virtual Mach

- ✓ Install packages
- ✓ Prepare virtual machine environment
- ✓ Write virtual machine configuration
- Start the virtual machine

Starting the virtual machine

75%

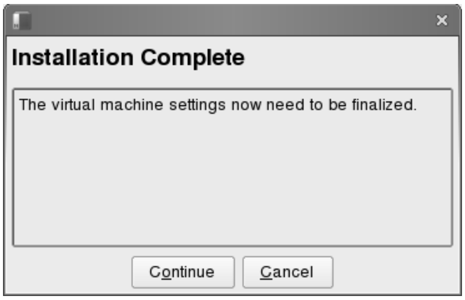
Xen - vm1

```
YaST @ 149.44.171.92                                Press F1 for Help

   Choose the           Language
Language to use
during installation
and for the                          Cymraeg
installed system.                    Dansk
   Click Accept to                   Deutsch
proceed to the next                  English (UK)
dialog.                              English (US)
   Nothing will                      Espanol
happen to your                       Francais
computer until you                   Greek
confirm all your                     Hindi
settings in the last                 Hrvatski
installation dialog.
   You can select
Abort at any time to
abort the
installation
process.               [Back]             [Abort]              [Next]
```

The installation itself is a standard SUSE Linux Enterprise Server installation, exept that it runs in text mode. After the packages have been installed, the following dialog appears:

Figure 1-9



Installation Complete

The virtual machine settings now need to be finalized.

Select **Continue**.

The following dialog gives you a resume about the domain configuration. Usually there is nothing to do here. Select **Next** in this dialog and in the domain overview.

A terminal window opens up where you can finish the remaining steps of the OS installation with YaST.

Exercise 1-2 Install a Guest Domain

In the following you can practice how to install a Xen guest domain using YaST. Before you start with this exercise you must have installed xen on your system.

You can find this exercise in the workbook.

(End of Exercise)

Objective 5 Manage Xen Domains at the Command Line

In the following you learn how to manage Xen domains at the command line. This includes:

- Understand a Domain Configuration File
- Use the xm Tool
- Migrate a Guest Domain

Understand a Domain Configuration File

Every Xen domain needs a configuration file. For domains which have been created with YaST, the configuration file is usually located in **/etc/xen/vm/**.

Under **/etc/xen/examples**, you find two example files, which can be used if you would like to create a configuration from scratch.

- **xmexample1.** This is a template configuration file for a single domain.
- **xmexample2.** This is an example for multiple domain configurations in one file.

For the beginning, **xmexample1** is a better choice.

A configuration file contains several keywords which configure different aspects of a Xen domain. The following is an example configuration file using the most common options. The **#** character is used for comments. Please read the comments in the example for details about the configuration options.

```
# Unique name of the domain
name = “SLES10-WebServer”

# The following lines point to the kernel and initrd file
# on the filesystem of the domain. The filesystem itself is
# defined later.
kernel = "/boot/vmlinuz-Xen"
ramdisk = "/boot/initrd-Xen"

# The amount of memory that is initally assigned to the
# domain. This can be changed at runtime.
memory = 256

# The next line defines a some details about the network
# configuration. When left blank, defaults are used,
# which work fine in most cases.
vif = [ '' ]

# This defines the disc of the domain. “phy” means that the
# physical device /dev/hda1 is mapped to the virtual device
# /dev/hda1 in the domain. “w” indicates, that the disc is
# writable.
disk = [ 'phy:hda1,hda1,w' ]

# The following is an example for a file based filesystem
# image. In this case the “file:” keyword is used.
# disk = [ ’file:/data/vm/SLES10-disc.img,hda1,w’ ]

# Sets the device for the Linux Kernel
root = “/dev/hda1 ro‟
```

A good source for detailed documentation and howtos about Xen and the domain configuration files is the Xen wiki at: http://wiki.xensource.com/xenwiki/

Use the xm Tool

xm is the administration tool for Xen domains. xm communicates with the **xend** management process running on the domain0 Linux installation.

The following is the general format of a xm command line:

```
xm command [options] [arguments] [variables]
```

You can get a complete list of the most common xm commands by entering **xm help.** A complete list can be viewed with **xm help --long**. It is also possible to display specific information about a certain command with **xm help [command_name]**.

To start a virtual machine, the **create** command is used:

```
xm create -c -f /data/xen/SLES10-WebServer.conf
```

The **-c** option lets xm connect to the terminal of the started domain, so that you can interact with the system. To disconnect from the terminal and return to the original command line, enter the key combination **Ctrl-]**.

The **-f** option specifies the configuration file of the domain that should be started.

The command **list** displays information about the currently running Xen domains:

```
xm list
```

The output of the list command contains the following fields:

- **name.** The name of the domain as specified in the configuration file.
- **domid.** A numeric, consecutive domain ID, which is automatically assigned when the domain starts.

- **memory.** The amount of memory assigned to the domain.
- **vcpus.** The number of virtual CPUs utilized by this domain.
- **state.** The current state of the domain. This could be:
  - **r**. The domain is running.
  - **b**. The domain has been created, but is currently blocked. This can happen, when a domain is waiting for I/O or when there is nothing to do for domain.
  - **p**. The domain is paused. The state of the domain is saved and can be restored.
  - **s**. The domain is in the process of being shutdown.
  - **c**. The domain is crashed, due to an error or missconfiguration.

An alternative to list is the command **top**, which displays domain information updated in realtime.

The **console** command connects you with the terminal of a running domain:

```
xm console <domain_id>
```

The command takes the domain id as a parameter, which can be determined with the **list** command (field domid). As mentioned before, use the key combination **Ctrl-]** to disconnect from a terminal.

With the **pause** command you can interrupt the execution of a domain temporarily:

```
xm pause <domain_id>
```

A paused domain is not completely shut down. The current state is saved and the execution of the domain can be continued with the **unpause** command:

```
xm unpause <domain_id>
```

To shutdown a domain, use the **shutdown** command:

```
xm shutdown <domain_id>
```

In case the domain is not responding anymore, you can force the shutdown of the domain with the **destroy** command:

```
xm destroy <domain_id>
```

To save the state of a domain for a longer time (eg. over a reboot of domain0) you can use the **save** command:

```
xm save <domain_id> <filename>
```

The domain can be restored from the resulting file with the **restore** command:

```
xm restore <filename>
```

Another commonly used command is **mem-set**, which allows you to change the memory allocation of a domain:

```
xm mem_set <domain_id> <amount_of_momory>
```

The amount of memory is specified in megabytes.

Instead of the domain ID <domain_id>, you can also use the domain name in all xm commands.

Exercise 1-3 Change Memory Allocation of a Guest Domain

In the following you learn how to change the memory allocation of a guest domain by changing the domain configuration file.

You can find this exercise in the workbook.

(End of Exercise)

Automate Domain Startup and Shutdown

When you start, shutdown or reboot domain0 of a Xen system, this also affects the other running Xen domains. Without a running domain0, the other Xen domains cannot operate.

SUSE Linux Enterprise Server 10 comes with a start script called **xendomains** which is included in the package **xen-tools**.

The script should be installed on domain0 and does the following:

- When domain0 is booted, all domains with configuration files located under **/etc/xen/auto/** are started.
- When domain0 is shutdown or rebooted, running Xen domains are shutdown automatically.

The script has some configuration options, which can be adjusted in the file **/etc/sysconfig/xendomains**. The configuration variables in this file are well documented.

One interesting option is to migrate domains automatically to a different host when a domain0 is shutdown. This can be configured in the variable **XENDOMAINS_MIGRATE**. The variable has to be set to the IP address of the target machine. When the variable is empty, no migration is performed.

Exercise 1-4 Automate Domain Startup

In this exercise, you learn how to startup domains automatically when the system is booted.

You can find this exercise in the workbook.

(End of Exercise)

Objective 6 Understand Xen Networking

Usually the network connection of Xen domains works out of the box. However, in case you would like to change the configuration, networking with Xen can be a bit tricky. The following should give you an overview of how Xen domains are connected to the physical network.

To better understand the concept of Xen networking, do the following:

- Understand the Basic Networking Concept
- Understand Bridging
- Understand the Network Interfaces in domain0

Understand the Basic Networking Concept

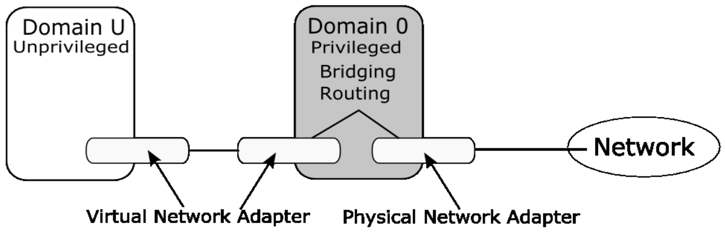
In a Xen setup, domain0 is controlling the physical network interfaces of a host system. Unprivileged domains are connected to domain0 through virtual ethernet adapters.

One virtual adapter in an unprivileged domain is connected to one virtual adapter in domain0.

In domain0, standard Linux networking mechanisms like bridging or routing are used to connect the virtual adapters through the physical adapter to the network.

The following is an illustration of this basic concept:

Figure 1-10



Understand Bridging

On SUSE Linux Enterprise Server 10, the default mechanism to connect virtual and physical interfaces in domain0 is bridging. Other mechanisms like routing with or without Network Address Translation (NAT) are out of the scope of this course.

Bridging basically means that multiple network interfaces are combined to one. Traditionally this technique is used to connect two physical network interfaces or network segments.

In a Xen system, bridging is used to connect virtual and physical network adapters in domain0. In a Xen system, you can consider the bridge as a kind of virtual switch which all virtual and physical interfaces are connected to.

The configuration of the bridge is done by the xend management process. When a new domain is created, the following changes to the network configuration are made (simplified):

1. Xen provides a virtual interface to the new domain.
2. xend creates a new virtual interface in domain0.
3. Both virtual interfaces are connected through a virtual point to point connection.
4. The virtual interface in domain0 is added to the bridge with the physical interface.

These steps only affect the general network connectivity. The IP configuration in the Xen domains has to be done separately with DHCP or a static network configuration.

xend is performing these network changes with the help of scripts, which are located at **/etc/xen/scripts/**. The following scripts are used for bridged networking:

- **network-bridge**. This script is called initially when xend is started. It sets up the bridge **xenbr0** and moves the physical interfaces onto that bridge.
- **vif-bridge**. This script is called for every domain that is started and adds the virtual interface to the bridge.

In the file **/etc/xen/xend-config.sxp** you can configure which network scripts are used by xend.

Understand the Network Interfaces in domain0

When you look at the network interfaces in domain0 with the command **ip a**, you can see that there are many more interfaces than in a regular Linux installation.

The following is an example output of **ip a** on domain0 (shortened):

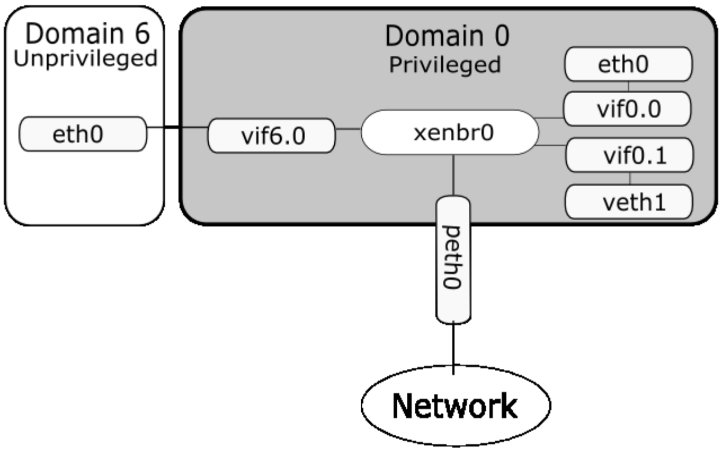
```
linux-3rsm:~ # ip a
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: peth0: <BROADCAST,MULTICAST,NOARP,UP> mtu 1500 qdisc
pfifo_fast qlen 100
    link/ether fe:ff:ff:ff:ff:ff brd ff:ff:ff:ff:ff:ff
    inet6 fe80::fcff:ffff:feff:ffff/64 scope link
       valid_lft forever preferred_lft forever
4: vif0.0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
    link/ether fe:ff:ff:ff:ff:ff brd ff:ff:ff:ff:ff:ff
    inet6 fe80::fcff:ffff:feff:ffff/64 scope link
       valid_lft forever preferred_lft forever
5: eth0: <BROADCAST,MULTICAST,NOTRAILERS,UP> mtu 1500 qdisc
noqueue
    link/ether 00:11:25:81:4c:5b brd ff:ff:ff:ff:ff:ff
    inet 149.44.171.67/23 brd 149.44.171.255 scope global
eth0
    inet6 2001:780:101:aa00:211:25ff:fe81:4c5b/64 scope
global dynamic
       valid_lft 29998sec preferred_lft 9996sec
    inet6 fe80::211:25ff:fe81:4c5b/64 scope link
       valid_lft forever preferred_lft forever
6: vif0.1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop
    link/ether fe:ff:ff:ff:ff:ff brd ff:ff:ff:ff:ff:ff
7: veth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
[...]
19: veth7: <BROADCAST,MULTICAST> mtu 1500 qdisc noop
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
20: xenbr0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
    link/ether fe:ff:ff:ff:ff:ff brd ff:ff:ff:ff:ff:ff
    inet6 2001:780:101:aa00:fcff:ffff:feff:ffff/64 scope
global dynamic
       valid_lft 29998sec preferred_lft 9996sec
    inet6 fe80::200:ff:fe00:0/64 scope link
       valid_lft forever preferred_lft forever
23: vif3.0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
    link/ether fe:ff:ff:ff:ff:ff brd ff:ff:ff:ff:ff:ff
    inet6 fe80::fcff:ffff:feff:ffff/64 scope link
       valid_lft forever preferred_lft forever
```

The following interface naming schema is used in domain0:

- **peth**. These are **p**hysical interfaces in domain0. peth devices are connected to the network bridge.
- **vif**. These are **v**irtual **i**nter**f**aces which are part of the bridge. The name of a vif interface identifies to which domain this interface is connected to. **For example:** vif6.0 is connected to the first virtual interface in domain 6.
- **veth**. These virtual interfaces are connected to the vif interfaces of domain0 (vif0.x). By default 7 vif <-> veth pairs are created. The veth interfaces can be used for more complex network setups.
- **eth0**. The first veth interface is named eth0 and connected with vif0.0. This is the "default" network interface of domain0.
- **xenbr0**. This is the default bridge that connects virtual and physical interfaces.

The following illustration gives you an overview of the interfaces in domain0.

Figure 1-11



You can use the command **brctl show** in domain0, to see which interfaces have been added to the network bridge.

Due to the complexity of the Xen network setup, the default firewall (SuSEFirewall2) is not working correctly in domain0. It is therefore recommended to disable SuSEFirewall2 and to setup a customized firewall if needed.

Exercise 1-5 Check the Network Configuration

This exercise assumes that you have a Xen system with domain 0 and one more Xen domain running.

You can find this exercise in the workbook.

(End of Exercise)

Objective 7 Migrate a Guest Domain

One advantage of virtualization is that domains can easily be moved from one physical system to another. Under Xen this procedure is called a **domain migration**.

A domain migration is performed by copying the current memory content. Please note the following before migrating a domain:

- There is no automatic way to copy the mass storage devices of a domain to another system. You have to make sure that the file systems (file system images or physical partitions) are available on the current and on the new host system. This can either be done by copying the data manually or by using a distributed file system (like NFS or SAN/NAS storage solutions).
- When a domain is migrated, the network settings are not automatically adjusted. Therfore the current and the new host system have to be in the same subnet or the network settings have to be manually adjusted after the migration.

You have the following two options to migrate a Xen domain:

- Use Domain Save and Restore
- Use Migration and Live Migration

Use Domain Save and Restore

A very simple way to migrate a domain is to use the save and restore function of the xm tool.

With the command **xm save <domain_id> <filename>**, you can suspend the specified domain and save the status to the given filename.

This file can then be copied to the new host system. To restore the domain, use the command **xm restore <filename>**.

As mentioned above, besides the file created with xm, you might also have to copy the filesystems to the new host system.

Use Migration and Live Migration

Instead of the save and restore commands of xm, you can also use the command **xm migrate <domain_id> <target_host>**. This command migrates a domain directly to a new host. In this case it´s not necessary to copy memory state files manually.

In order to get this working, the current and new host must be running Xen and xend.

By adding the option **--live** to the migration command line, the downtime during the migration can be reduced to typically 60-300ms. Instead of shutting down the domain before the migration starts, Xen attempts to keep it running while the migration is in progress.

The xend configuration file **/etc/xen/xend-config.sxp** contains two options concerning domain migration:

(xend-relocation-server yes)

This option enables the migrating functionality in xend.

(xend-relocation-hosts-allow '^localhost$')

This option controls which hosts are allowed to conntect to xend for domain migration. By default, only localhost is allowed to connect. The option takes regular expressions as parameter. Have a look at the configuration file for examples.

Please note, that there are two xend involved in a domain migration (current and new host). You might have to adjust the xend-config.sxp file on both systems.

Summary

Objective	Summary
1. Understand the Concept of Virtualization	Virtualization technology separates a running instance of an operating system from the physical hardware. Instead of a physical machine, the operating system runs in a so called virtual machine. Multiple virtual machines share the resources of the underlying hardware. Virtualization allows you to run multiple virtual systems on one single physical machine.
2. Understand How Xen Works	There are two different kinds of virtualization: ■ Full-Virtualization ■ Para-Virtualization Xen uses para-virtualization. It provides access to the physical hardware through an API.

Objective	Summary
3. Install Xen	The following packages have to be installed in the initial SUSE Linux Enterprise Server 10 installation: ■ **xen.** This package contains the Xen Virtual Machine Monitor (Hypervisor). ■ **xen-tools.** Contains xend and a collection of command line tools to administer a Xen system. ■ **kernel-xen.** This package contains a modified Linux kernel that runs in a Xen domain. ■ **xen-doc-*** (optional)**.** Xen documentation in various formats. The installation of xen adds an entry in the grub configuration file.
4.Manage Xen Domains with YaST	YaST provides a module which can be used to create and manage Xen domains. The module is called: **Virtual Machine Management (Xen).** This module offers a convenient way to create and control the Xen domains on your system. The module can be started from the **System** section in the YaST Control Center, and has to run on the Linux system running in domain0.

Objective	Summary
5. Manage Xen Domains at the Command Line	Every Xen domain needs a configuration file. Usually this is located in **/etc/xen/vm/**. **xm** is the central administration tool for xen domains. To start a virtual machine, the **create** command is used. For example: xm create -c -f SLES10.conf Some services are not required in a xen environment and can be removed. ■ **insserv -r earlykbd** ■ **insserv -r kbd** ■ **insserv -r irq_balancer** Under Xen, all domains are connected with the physical network through domain0.
6. Understand Xen Networking	Domain0 is the central point to configure the network connections on a Xen system. A network bridge in domain0 is used as a virtual switch. This bridge is set up and controlled by xend.

Objective	Summary
7. Migrate a Guest Domain	One advantage of virtualization is, that domains can easily be moved from one physical system to another. Under Xen this procedure is called a **domain migration**. Domains can be migrated with xm‘s **save** and **restore** commands or with the **migrate** command.

SECTION 2 Configure a Web Application Server

In this section, you learn how to install and configure Apache and Tomcat on SUSE Linux Enterprise Server 10.

Objectives

1. Setup a Basic Web Server
2. Configure Virtual Hosts
3. Limit Access to the Web Server
4. Configure Apache with OpenSSL
5. Install PHP
6. Describe Tomcat
7. Install and Configure Tomcat
8. Install Web Applications
9. Use Tomcat‘s Administration Tools
10. Use Port 80 to Access Tomcat

Objective 1 Setup a Basic Web Server

The Apache web server is the leading web server software. To set up a basic Apache web server, you need to know the following:

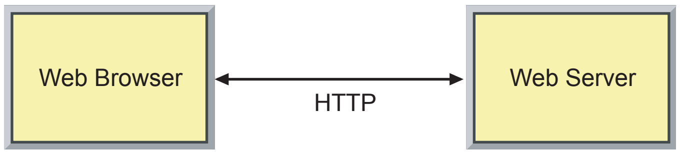
- The Basic Functionality of a Web Server
- Install a Basic Apache Web Server
- Understand the Structure and the Basic Elements of the Apache Configuration Files
- Understand the Default Apache Configuration

The Basic Functionality of a Web Server

A web server delivers data that is requested by a web browser. The data can have different formats such as HTML files, image files, Flash animations, or sound files.

Web browsers and web servers communicate using HTTP (Hyper Text Transfer Protocol). The following diagram shows the relationship between the browser, server, and HTTP:

Figure 2-1



In addition to delivering data to the web browser, a web server can perform tasks such as limiting access to specific web pages, logging access to a file, and encrypting the connection between a server and browser.

Install a Basic Apache Web Server

To set up a basic Apache web server, you need to do the following:

- Install the Required Software Packages
- Start and Test the Web Server
- Locate the DocumentRoot of the Web Server

Install the Required Software Packages

To run a basic Apache web server, you need to install the following packages with YaST:

- **apache2.** The basic web server software.
- **apache2-prefork.** An additional Apache package that influences the multiprocessing behavior of the web server.
- **apache2-example-pages.** Sample HTML pages.

When you install the packages listed above, YaST prompts you to also install one or more additional packages required by Apache. Confirm the additional package installation by selecting **OK** to resolve all dependencies of the Apache packages.

Start and Test the Web Server

After installing the required software, you need to start the web server. Do this as the root user by entering the following:

rcapache2 start

As with all services, enter the following to stop the web server:

rcapache2 stop

If you want the web server to start up at boot time, you need to enter the following:

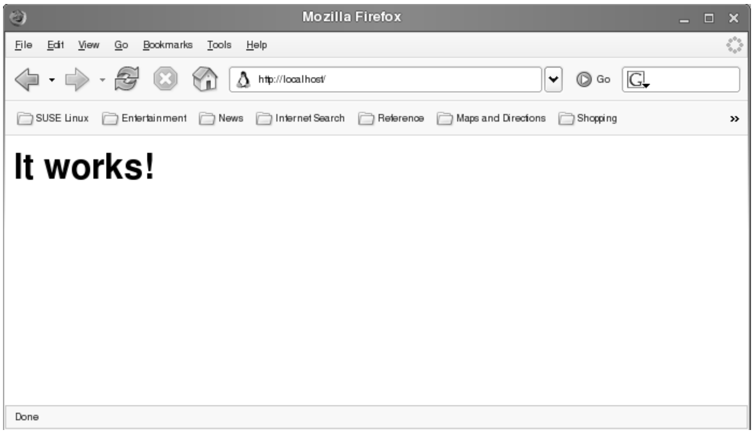
insserv apache2

To test whether the web server is properly installed, open a web browser and enter the following address:

http://localhost

The browser displays the following page:

Figure 2-2


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/
Go
SUSE Linux Entertainment News Internet Search Reference Maps and Directions Shopping
Done

It works!

If your SUSE Linux Enterprise Server 10 is connected to a network, you (and other hosts on the network) can remotely access the web server by entering the following:

http://*your_system_IP_address*

If your network provides a DNS server, you can use the hostname instead of the IP address.

Locate the DocumentRoot of the Web Server

The default directory of the data provided by Apache is **/srv/www/htdocs**.

This directory is also called the *DocumentRoot* of the web server. After the installation, it contains the Apache example pages, which are displayed above.

You can replace the data in the DocumentRoot directory to display your own web server content. Because the web server runs with the user id wwwrun, you have to make sure that this user has *read* access to files in the DocumentRoot directory.

If you create subdirectories in DocumentRoot, you can access those subdirectories with the following web address scheme:

http://*your_server*/*name_of_subdirectory*

If no specific file is requested in the address, Apache looks for a file with the name index.html. You can change the name of this default file in the Apache configuration files.

Exercise 2-1 Install Apache

In this exercise, you install the apache components on your system

You can find the exercise in the workbook.

(End of Exercise)

Exercise 2-2 Test the Apache Installation

In this exercise, you check if the installation of apache was successful.

You can find the exercise, in the workbook.

(End of Exercise)

Understand the Structure and the Basic Elements of the Apache Configuration Files

To configure the Apache web server, you need to do the following:

- Locate the Apache Configuration Files
- Understand the Basic Rules of the Configuration Files

Locate the Apache Configuration Files

The configuration of the Apache web server is spread over several configuration files located in the directory /etc/apache2.

The following is a list of the most important Apache configuration files:

- **httpd.conf.** This is the main Apache configuration file.
- **default-server.conf.** This file contains the basic web server setup. However, all options set in this file can be overwritten by other configuration files.
- **vhost.d/.** This is a directory containing configuration files for virtual host setups. You will learn more about virtual hosts later in this section.
- **uid.conf.** This configuration file sets the user and group id for Apache. By default, Apache uses the user id wwwrun and the group id www.
- **listen.conf.** In this configuration file, you can specify the IP addresses and TCP/IP ports Apache is listening to. By default, Apache listens to all assigned interfaces on port 80.
- **server-tuning.conf.** You can use this configuration file to fine tune the performance of Apache. The default values should be fine unless you are going to run a web server that has to handle a lot of requests at the same time.
- **error.conf.** In this file you configure the behavior of Apache when a request cannot be performed correctly.

- **ssl-global.conf.** Configure the connection encryption with SSL in this configuration file.

Understand the Basic Rules of the Configuration Files

The options of the Apache configuration files are called *directives***.** Directives are case sensitive, which means that a word such as “include” is not the same as “Include.”

Directives can be grouped so that they do not apply to the global server configuration. In the following, the directives only apply to the directory /srv/www/htdocs:

```
<Directory "/srv/www/htdocs">
        Options None
        AllowOverride None
        Order allow,deny
        Allow from all
</Directory>
```

The directives are grouped by <Directory “/srv/www/htdocs”> and </Directory> which limits their validity to the directory /srv/www/htdocs only.

You can use the # character to indicate comments in the configuration file. All lines starting with a # are ignored by the Apache server.

Whenever you edit the Apache configuration files, you need to reload the web server by entering the following:

rcapache2 reload

In some cases it´s not enough to reload Apache. You need to stop and restart the web server by entering the following:

rcapache2 restart

If you are not sure that your changes use the correct syntax, you can verify the syntax of the configuration files by entering the following:

apache2ctl configtest

If the syntax is correct, the command displays the following message:

```
Syntax OK
```

Understand the Default Apache Configuration

The main Apache web server configuration is defined in the file **/etc/apache2/default-server.conf.** The follwing is an overview of the most important directives used in that file:

Table 2-1

Directive	Description
DocumentRoot	Specifies the DocumentRoot of the web server.
<Directory "*dir_name*"> /<Directory>	All directives used within this block apply only to the specified directory.
Options	With this directive additional options can be applied to logical blocks like directories.
AllowOverride	Determines whether directives are allowed to be overwritten by a configuration found in a .htaccess file in a directory.
Alias "*fakename*" "*realname*"	Allows you to create an alias to a directory.
ScriptAlias	Allows you to create an alias to a directory containing scripts for dynamic content generation.

In most cases the default settings are suitable and don't need to be changed.

An overview of all Apache directives can be found at http://httpd.apache.org/docs-2.0/mod/directives.html.

Objective 2 Configure Virtual Hosts

To use the virtual host feature of Apache, you need to know the following:

- The Concept of Virtual Hosts
- Configure a Virtual Host

The Concept of Virtual Hosts

With the default setup, the Apache server can be reached with a browser using the following web addresses (URLs):

- **http://localhost** (from the computer where the web server is running)
- **http://*web_server_IP_address***
- **http://*web_server_hostname***

For all of these addresses, Apache serves the same files located in the DocumentRoot directory.

Using this setup, you would need a dedicated computer for every domain of the Internet. To avoid this, Apache can be configured to host multiple virtual web servers on one physical system. These virtual web servers are called virtual hosts.

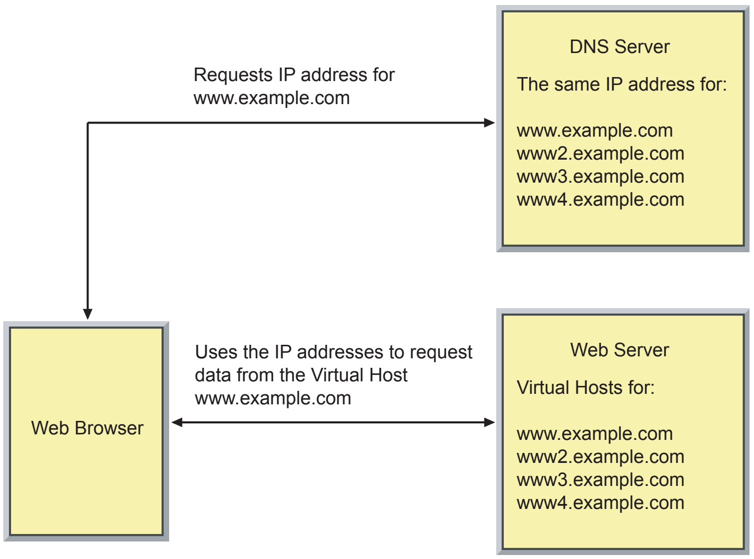
To access virtual hosts, a DNS entry is needed for every virtual host of the Apache web server.

The following outlines the steps of sending a request to the virtual host www.example.com:

1. The web browser requests the IP address of the host www.example.com.
2. The browser uses the IP address to request a file from the Apache web server listening on the IP address of www.example.com.
3. In the HTTP request, the browser includes the hostname of the server it wants to reach.
4. Apache uses the hostname to determine the corresponding virtual host and delivers the requested data from that host.

The following illustrates this process:

Figure 2-3



Configure a Virtual Host

For every virtual host you need to create a configuration file in the directory /etc/apache2/vhosts.d/. The name of the configuration file has to end with .conf.

You can find a template file vhost.template in the directory /etc/apache2/vhosts.d/ to use as a base for your configuration file.

You need to edit the following directives in the template:

Table 2-2

Directive	**Description**
ServerAdmin	Enter the email address of the Virtual Host administrator here.
ServerName	Enter the hostname of the virtual host as it is configured in the DNS.
DocumentRoot	Set the DocumentRoot of the virtual host. The directory and the files in the directory must be readable by the user wwwrun.
ErrorLog	Enter a filename for the error log. The file must be writable for the user wwwrun.
CustomLog	Enter a filename for the general log file. The file must be writable for the user wwwrun.
ScriptAlias	Set the ScriptAlias to a directory of your choice. The directory must not be under the DocumentRoot of the virtual host. If you don’t need scripts for dynamic content creation, delete this directive.
<Directory “script_dir”>	If you’ve set a ScriptAlias before, you have to configure a directory which contains the script files. If you are not using a ScriptAlias, delete this directory block.
<Directory “document_root”>	You need to adjust the path name of this directory directive to the path of your DocumentRoot.

After customizing the template file, you need to reload the Apache web server. You also need to make sure that the settings in DNS are updated so that the **hostname** of your virtual host is resolved correctly.

Exercise 2-3 Configure a Virtual Host

In this exercise, you configure a virtual host for the accounting department.

You can find this exercise, in the workbook.

(End of Exercise)

Objective 3 Limit Access to the Web Server

Normally Apache delivers data to all hosts in the network that can reach the web server. Sometimes it can be useful to restrict access to the content delivered by Apache.

The following are the most common methods used:

- Limit Access on an IP Address Basis
- Limit Access with User Authentication

Limit Access on an IP Address Basis

Apache offers the following directives to limit access to the web server on an IP address basis:

Table 2-3

Directive	Description
allow	IP addresses or networks listed after this directive are allowed to access the web server.
deny	IP addresses or networks listed after this directive are not allowed to access the web server.
order	This directive sets the order in which the allow and deny directives are evaluated.

These directives must be used within a **<Directory>** block and control the access to all data in that directory and all subdirectories.

The following example allows only hosts from the network 10.0.0.0/24 to access the data in the directory /srv/www/htdocs:

```
<Directory "/srv/www/htdocs">
        Order deny,allow
        Deny from all
        Allow from 10.0.0.0/24
</Directory>
```

The following lists and describes the lines in the example:

- **<Directory "/srv/www/htdocs">.** This directive starts the directory block. The directives that follow apply to the directory /srv/www/htdocs only.
- **Order deny,allow.** The Order directive determines in which order the allow and deny directives are evaluated.

  You have the following options:

  - **Deny,Allow.** The deny directives are evaluated before the allow directives. Access is allowed by default. Any client which does not match a deny directive or does match an allow directive is allowed access to the server.
  - **Allow,Deny.** The allow directives are evaluated before the deny directives. Access is denied by default. Any client which does not match an allow directive or does match a deny directive is denied access to the server.
  - **Mutual-failure.** Only those hosts that appear in the Allow list and do not appear on the Deny list are granted access. This has the same effect as **Order Allow,Deny**.
  - **Deny from all.** The Deny directive is evaluated first, and in this case access is denied for all clients. You can use the following options with the deny and the allow directives:
  - **all.** This option applies to all hosts.
  - **A (partial) domain-name.** This option applies to hosts which names match or end in the given expression (such as novell.com). Only complete domain components are matched. "novell.com" would match www.novell.com but not foonovell.com.
  - **A full IP address.** This option applies to a specific IP address (such as 10.0.0.23).
  - **A partial IP address.** This option applies to IP addresses starting with the given IP address fragment (such as 10.0.0).
  - **A network/netmask pair.** This option applies to IP addresses matching to the given network/netmask pair (such as 10.0.0.0/255.255.255.0)

    - **A network/nnn CIDR specification.** This option applies to IP addresses matching to the given CIDR expression (such as 10.0.0.0/24).

- **Allow from 10.0.0.0/24**. This allow directive is evaluated after the deny directive. In this case, the access is allowed for hosts in the network **10.0.0.0/24**.
- **</Directory>.** This directive ends the directory block.

Limit Access with User Authentication

By limiting access to certain IP addresses, you can control the hosts that access the web server, but you have no control over the user that sits in front of the computer.

Apache offers another option of access control called basic authentication. If you protect content on your web server with this method, users are required to log in before they can access the data.

Before you can configure Apache to use basic authentication, you first have to create user accounts for the web server. You can do this by using the tool htpasswd2.

The following command creates a password file and an account for the user tux.

```
htpasswd2 -c /etc/apache2/htpasswd tux
```

After entering this command, htpasswd2 prompts you for a password for the user you want to create. The passwords are stored in the file /etc/apache2/htpasswd.

You can specify a different location for the password file, but you have to make sure that it is readable for the user wwwrun and that it is *not* located within the DocumentRoot of your server.

When you use a password file for the first time, you have to call htpasswd2 with the -c option to create the file. If you want to add more users later, use the following command:

htpasswd2 /etc/apache2/htpasswd *username*

To delete a user from the password file, use the following command:

htpasswd2 -D /etc/apache2/htpasswd *username*

After you have created the user accounts, you need to configure Apache to prompt for a password when accessing restricted data. You need to add the following lines to the directory block of the directory that should be restricted:

```
AuthType Basic
AuthName "Restricted Files"
AuthUserFile /etc/apache2/htpasswd
Require user tux
```

The following describes each line:

- **AuthType Basic.** This directive sets the authentication method. For the type described in this section, the value is Basic.
- **AuthName "Restricted Files".** With this directive, you have to choose a name for the restricted directory of your web server. This name is used for the authentication process between the browser and the web server.
- **AuthUserFile /etc/apache2/htpasswd.** This directive sets the password file used for the restricted directory.
- **Require user tux.** This directive lists the user from the password file who is allowed to access the directory. You can add more than one user by separating the user names with spaces, or you can use the following directive:
- **Require valid-user**

  This defines that any valid username / password combination is granted access.

The password is transfered in clear text over the network. For critical applications it's recommended to configure SSL encription. See next objective for details.

Exercise 2-4 Configure User Authentication

In this exercise, you add user authentication to a virtual host.

You can find this exercise in the workbook.

(End of Exercise)

Objective 4 Configure Apache with OpenSSL

By default, the connection between the web browser and the web server are not encrypted. Anyone who can listen to the network packets exchanged between browser and server can access the transferred information.

Apache can use the SSL (Secure Socket Layer) protocol to encrypt the connection. To configure an SSL encryption with an Apache web server, you need know the following:

- The Basics of SSL Encryption
- Create a Test Certificate
- Configure Apache to Use SSL
- The Limitations of the SSL Configuration

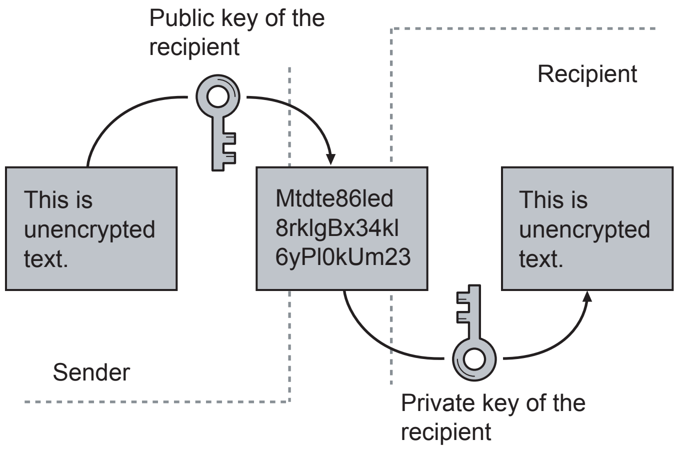
The Basics of SSL Encryption

RSA keys are a very often used for data encryption. RSA is for example used by the encryption software PGP (Pretty Good Privacy) to encrypt emails, by ssh (Secure Shell) for encrypted data transfers between two computers, and by Apache for secure data transmission between the web server and the web browser.

This encryption is based on two different keys: a private key and a public key. While the private key is known only to the owner, the public key should be accessible to the public.

The following shows the encryption process:

Figure 2-4

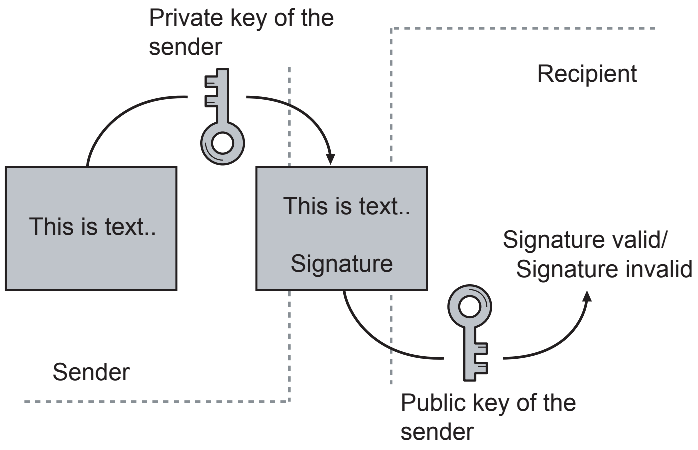

Public key of the recipient
Recipient
This is unencrypted text.
Mtdte86led 8rklgBx34kl 6yPl0kUm23
This is unencrypted text.
Sender
Private key of the recipient

Public and private keys can also be used to sign data. In principle, when data is signed, an encrypted checksum is generated from the data. The sender signs the data with his private key.

The signature can be checked by the recipient by using the public key of the sender to determine whether the data is really from her or whether the text has been modified by a third party.

The following illustrates the signing process:

Figure 2-5



A problem with the encryption procedure described above is that you cannot determine who the owner of a public key is. The solution to this problem is a Certificate Authority (CA) which signs the public keys with its own private keys.

A public key that is signed by a CA is also called a *Certificate*.

CAs are well-known companies or organizations like VeriSign or VISA. The public keys of these organizations are built into the web browsers. By verifying the signature with the public key of the CA, the browser can make sure that a public key of a web server is valid.

The following explains the process of using a CA with SSL encryption for a web server:

1. The browser recognizes a web address starting with https**://.**

   This means that the connection to this server should be encrypted. The default port for SSL connections is 443 instead of port 80 (used for normal unencrypted HTTP connections).

2. The web browser asks the server for its public RSA key (certificate).

3. The web server sends the public key to the web browser.

4. The web browser verifies the key of the server with the public key of the CA that signed the key.
5. If the key is valid, the web browser and web server establish a secure connection.

You need an officially signed key to set up a secure web server. You can sign a key by yourself, but this should only be done for test purposes.

Create a Test Certificate

To set up a secure web server for test purposes, you can create a certificate by yourself. You should never use such a certificate for a production system.

To create a test certificate, you do the following:

- Create an RSA Key Pair
- Sign the Public Key to Create a Certificate

Create an RSA Key Pair

To create a key pair, you need a file with as many random numbers as possible. You can generate this file by entering the following command:

```
cat /dev/random > /tmp/random
```

Stop this procedure after a few seconds by pressing **Ctrl+C**. The file generated should be at least a thousand bytes in size. You can now generate the key pair by entering the following command:

```
openssl genrsa -des3 -out server.key -rand /tmp/random 1024
```

During the process, you are prompted to enter a password. This password is used to secure the private key of the key pair.

The generated keys are saved together in the file server.key.

Sign the Public Key to Create a Certificate

Next you need to sign your public key to create a certificate by entering the following command:

openssl req -new -x509 -key server.key -out server.crt

During the process, you are prompted for the following information:

- **Enter pass phrase for /tmp/server.key:**

  Enter the passphrase you chose for the server key.

- **Country Name (2 letter code) [AU]**

  Enter the country code of your country (such as DE for Germany).

- **State or Province Name (full name) [Some-State]:**

  Enter your state or province name. You can enter a period (.) to leave this field blank.

- **Locality Name (eg, city) []:**

  Enter the name of your city.

- **Organization Name (eg, company) [Internet Widgits Pty Ltd]:**

  Enter the name of your company.

- **Organizational Unit Name (eg, section) []:**

  Enter the name of your unit, or you can enter a period (.) to leave it blank.

- **Common Name (eg, YOUR name) []:**

  Enter the fully qualified domain name of your system (such as www.example.com). The certificate will be valid for this hostname only.

- **Email Address []:**

  Enter the email address of the administrator who is responsible for the server.

After you have answered all questions, the server certificate is saved into the file server.crt.

Finally, you need to copy the files server.key and server.crt to the correct locations:

- Copy the file server.key to the directory **/etc/apache2/ssl.key**.
- Copy the file server.crt to the directory **/etc/apache2/ssl.crt**.

Configure Apache to Use SSL

After you have generated the RSA key pair and the server certificate, you have to configure Apache to use SSL. First, you need to change two settings in the file /etc/sysconfig/apache2.

The settings in this file apply to the Apache startup script and do not belong to the server configuration.

Set the following variables to the appropriate values:

- **APACHE_START_TIMEOUT=”10”**

  This setting extends the start timeout of Apache so that you have more time to enter the passphrase of the private RSA key.

- **APACHE_SERVER_FLAGS=”SSL”**

  The additional server flag SSL defines the SSL variable when evaluating the Apache configuration files. This enables some directives that are necessary for SSL encryption.

For example, it lets Apache listen on port 443 instead of only to port 80.

You also need to change the server configuration files to enable SSL by doing one of the following:

- Configure the Main Server to Use SSL Encryption
- Configure a Virtual Host to Use SSL Encryption

Configure the Main Server to Use SSL Encryption

To configure the main server, you need to add the following directives to the file /etc/apache2/default-server.conf:

```
SSLEngine on
SSLCipherSuite
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
SSLCertificateFile /etc/apache2/ssl.crt/server.crt
SSLCertificateKeyFile /etc/apache2/ssl.key/server.key
```

Each line is described below:

- **SSLEngine on**

  This directive enables the Apache SSL engine.

- **SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL**

  This directive sets the details of the encryption method. The line displayed above is the default configuration that comes with Apache.

For more information about this directive, go to http://httpd.apache.org/docs-2.0/mod/mod_ssl.html#sslciphersuite.

- **SSLCertificateFile /etc/apache2/ssl.crt/server.crt**

  This directive points to the server certificate file.

- **SSLCertificateKeyFile /etc/apache2/ssl.key/server.key**

  This directive points to the server key file.

After you make the described changes, you have to restart Apache. Apache prompts you for the passphrase of the server key file.

The server might not start up correctly at boot time, because it requires the passphrase for the server key. You should remove Apache from the init process and start it manually after the system starts up.

You can access the SSL host by using the address **https://*name_of_your_host***.

Configure a Virtual Host to Use SSL Encryption

You can also configure a virtual host instead of the main server to use SSL. Place the directives described above in your virtual host configuration and define you virtual host with a directive such as the following:

<VirtualHost *your_hostname*:443>

The Limitations of the SSL Configuration

The SSL setup as described in this section is a very basic configuration. To run Apache with SSL on a server that can be reached from the Internet, you need a more thorough understanding of SSL and the available configuration directives.

For more information about SSL and Security, we recommend you to take our course 3075 (SUSE Linux Enterprise Server 10: Security).

Exercise 2-5 Configure SSL

In this exercise, you add SSL encryption to a virtual host.

You can find this exercise in the workbook.

(End of Exercise)

Objective 5 Install PHP

PHP is a very popular programming language for web application. In this objective you learn how to install PHP on a SUSE Linux Enterprise Server 10. This includes:

- Understand how PHP Works
- Install PHP
- Test the PHP Installation

Understand how PHP Works

PHP is a scripting language. To install a PHP web application, the program files need to be copied into the document root of the web server. These files usually end in **.php**.

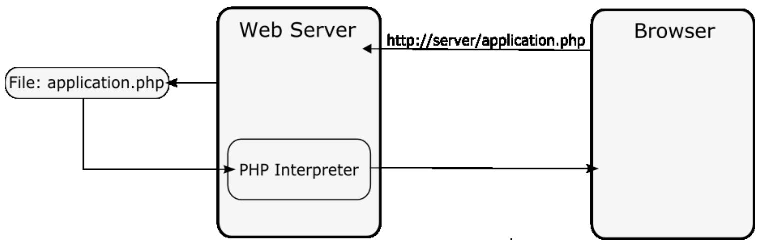
The application can be started by accessing the PHP file with an ULR like **http://www.mydomain.com/application.php**. The web server opens the PHP file, but instead of sending it directly to the browser, it is passed through the PHP interpreter first.

The PHP interpreter runs the PHP code in the file and passes the output through the webserver to the browser. Therefore the user in front of the browser does not see the code of the PHP application but the output of it. The output is formatted in HTML so that it can be displayed by a browser.

The PHP interpreter is realized as an Apache extension module. It is also possible to run PHP applications directly via CGI, but this is not covered in this course.

The following is an overview of the PHP architecture:

Figure 2-6



Install PHP

On SUSE Linux Enterprise Server 10, the PHP components are split into multiple software packages. You need at least the following packages for a basic PHP web application server:

- **PHP5.** The core PHP interpreter and libraries.
- **apache2_mod_php5.** The PHP module for Apache.

When you search for "php" in YaST's software management module, you'll notice that there are many more php packages available (php-*). These modules extend the functionality of PHP. Which packages you need, depends on the requirements of the PHP application you would like to run.

The PHP interpreter has some configuration options, which can be adjusted in the file **/etc/php5/apache2/php.ini**. The default configuration should be fine for most purposes. The following are a few selected options:

- **memory_limit**. This defines how much memory a script is allowed to use. For complex applications, this might need to be set to a higher value. Default is 8MB.
- **max_execution_time**. This option sets the maximum execution time in seconds. Complex applications sometimes need a longer execution time. The default is 30 seconds.

- **display_errors.** Determines whether errors or warning messages are displayed in the HTML output. For production systems, this option should be set to Off, while on a development system it is usefull to switch it On. The default is Off.

After installing PHP packages, you have to restart Apache with the command:

rcapache2 restart

Test the PHP Installation

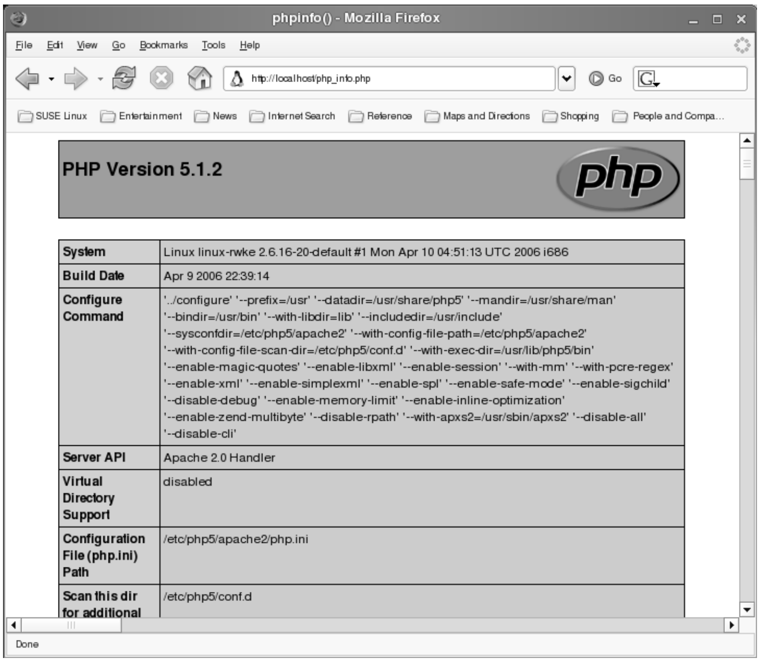
A PHP installation can easily be tested by creating a file with the following content somewhere in the document root of the web server:

```
<?PHP
phpinfo();
?>
```

The content is a simple php application. Calling the function **phpinfo()** outputs a complete web page with information about the php installation.

When you request this file with a URL like **http://www.myserver.php/php_info.php** (assuming you named the file php_info.php) a page like the following is displayed:

Figure 2-7



PHP Version 5.1.2

System	Linux linux-rwke 2.6.16-20-default #1 Mon Apr 10 04:51:13 UTC 2006 i686
Build Date	Apr 9 2006 22:39:14
Configure Command	'../configure' '--prefix=/usr' '--datadir=/usr/share/php5' '--mandir=/usr/share/man' '--bindir=/usr/bin' '--with-libdir=lib' '--includedir=/usr/include' '--sysconfdir=/etc/php5/apache2' '--with-config-file-path=/etc/php5/apache2' '--with-config-file-scan-dir=/etc/php5/conf.d' '--with-exec-dir=/usr/lib/php5/bin' '--enable-magic-quotes' '--enable-libxml' '--enable-session' '--with-mm' '--with-pcre-regex' '--enable-xml' '--enable-simplexml' '--enable-spl' '--enable-safe-mode' '--enable-sigchild' '--disable-debug' '--enable-memory-limit' '--enable-inline-optimization' '--enable-zend-multibyte' '--disable-rpath' '--with-apxs2=/usr/sbin/apxs2' '--disable-all' '--disable-cli'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2/php.ini
Scan this dir for additional	/etc/php5/conf.d

Exercise 2-6 Install PHP

In this exercise, you install and test PHP.

You can find the exercise in the workbook.

(End of Exercise)

Objective 6 Describe Tomcat

In the early days of the World Wide Web almost all web sites provided static content. This means that the content was stored in HTML files on a web server, and the pages were delivered “as they are” to a requesting web browser.

Today more and more web sites provide customized content, which is different for every viewer accessing the site. For example, content can be a webmail service or a customized portal page.

These pages are generated on the fly when they are requested. The logic that generates the web pages is written in a programming language. Popular web programming languages are PHP, Perl, or Python.

Programs that interact with the user by outputting HTML pages are called **web applications**. The servers they are running on are called **web application servers**.

The Java programming language is also used for web applications. The advantages of Java, such as portability, performance, reusability, and crash protection make it the programming language of choice for larger and more complex web applications.

The following two terms are often used when speaking about web applications written in Java:

- **Java servlets**. These are the actual web applications that are developed according to the Java servlet specifications.

  They are platform independent and can be run on any platform providing a Java environment.

- **JSP** (Java Server Pages). This is an extension to the Java servlet specification, which allows developers to separate HTML and application code.

  By using special JSP tags in an HTML document, a developer can access functions in a Java servlet application.

To run an application that is implemented as a Java servlet, you need a servlet container. The servlet container provides a framework that allows you to load and run Java servlets on your server.

Tomcat is such a servlet container. It is developed by the Apache Foundation, which is also in charge of the Apache web server.

The homepage of the Tomcat project can be found at

http://jakarta.apache.org/tomcat/index.html

SUSE Linux Enterprise Server 10 ships with Tomcat version 5.0.19, which implements servlets 2.4 und JSP 2.0 specifications. Like the hosted web applications, Tomcat itself is written in Java.

Objective 7 Install and Configure Tomcat

To install and configure Tomcat on SUSE Linux Enterprise Server 10, you have to do the following:

- Install the Tomcat Packages
- Understand the File System Structure
- Edit the server.xml File

Install the Tomcat Packages

Tomcat is split into the following packages on SUSE Linux Enterprise Server 10:

- **tomcat5**. This is the main package. It contains the core Tomcat components.
- **tomcat5-admin-webapps**. This package contains Tomcats web based administration tools.
- **tomcat5-webapps**. This package contains some example web applications.

To run Tomcat on your system, you should at least install the packages **tomcat5** and **tomcat5-admin-webapps**.

After installation and configuration, you can start Tomcat by entering **rctomcat5 start**

To stop Tomcat, enter **rctomcat5 stop**

Unlike other services, there is no reload option for Tomcat. Instead you can use the command **rctomcat5 restart**

This is a combination of **rctomcat5 stop** and **rctomcat5 start**.

Exercise 2-7 Install Tomcat

In this exercise, you install the Tomcat components

You can find the exercise in the workbook.

(End of Exercise)

Understand the File System Structure

There are two configuration variables you always have to deal with when configuring Tomcat:

- **CATALINA_HOME**. This variable points to the root directory of all directories and files that belong to a Tomcat installation.

  This is comparable to the **ServerRoot** directive of the Apache web server.

  All instances of Tomcat that run on the same system share one CATALINA_HOME directory.

- **CATALINA_BASE**. This variable is used to run multiple instances of Tomcat on the same machine.

  In this case, every instance has its own CATALINA _BASE directory for configuration and log files.

When you run only a single instance of Tomcat, CATALINA_HOME and CATALINA_BASE point to the same directory.

This section covers only a single instance setup of Tomcat.

CATALINA_HOME and CATALINA_BASE are passed to Tomcat by the startup scripts, which read the variables from the file **/etc/sysconfig/j2ee**.

On SUSE Linux Enterprise Server 10, CATALINA_HOME points by default to **/usr/share/tomcat5/** and CATALINA_BASE points to **/srv/www/tomcat5/base/**.

The following are the most important sub-directories in **/usr/share/tomcat5/**:

- **bin/**. This directory contains scripts for administering the Tomcat server such as the start and stop scripts.

- **server/**. This directory contains the Tomcat server itself.

  It has subdirectories for the Java libraries that the Tomcat server needs to run and a subdirectory that includes web applications that can be used to administer Tomcat.

- **conf/**. This directory contains the configuration files of Tomcat.

  On SUSE Linux Enterprise Server 10, this is a link to **/etc/tomcat5/base/**.

- **logs/**. This directory contains Tomcat log files.

  On SUSE Linux Enterprise Server 10, this is a link to **/var/log/tomcat5/base/**.

- **webapps/**. This directory contains the web applications that are hosted by Tomcat.

  On SUSE Linux Enterprise Server 10, this is a link to **/srv/www/tomcat5/base/webapps/**.

All paths including **"base"** belong to the default instance of Tomcat on SUSE Linux Enterprise Server 10.

All directory paths not including **"base"** can be shared with other instances of Tomcat running on the same system when running a multiple instances setup.

Edit the server.xml File

The most important configuration file for every Tomcat instance is

server.xml

For the default instance, the file is located in

/usr/share/tomcat5/conf/

The configuration file is in XML format.

The following is a very basic example file:

```
<Server port="8005" shutdown="SHUTDOWN">
  <GlobalNamingResources>
    <!-- Used by Manager webapp -->
    <Resource name="UserDatabase" auth="Container"
              type="org.apache.catalina.UserDatabase"
      description="User database that can be updated and saved">
    </Resource>
    <ResourceParams name="UserDatabase">
      <Parameter>
        <name>factory</name>
        <value>org.apache.catalina.users.MemoryUserDatabaseFactory</value>
      </Parameter>
      <Parameter>
        <name>pathname</name>
        <value>conf/tomcat-users.xml</value>
      </Parameter>
    </ResourceParams>
  </GlobalNamingResources>

  <Service name="Catalina">
    <Connector port="8080" />

    <!-- This is here for compatibility only, not required -->
    <Connector port="8009" protocol="AJP/1.3" />

    <Engine name="Catalina" defaultHost="localhost">
      <Logger className="org.apache.catalina.logger.FileLogger" />

      <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
             resourceName="UserDatabase" />

      <Host name="localhost" appBase="webapps" />
    </Engine>
  </Service>
</Server>
```

The configuration file contains several configuration elements, which are nested into each other. Each element uses the following syntax:

```
<element_name argument1=”value” argument2=”value” >
<nested_element ...>
</nested_element>
</element_name>
```

When an element does not contain other nested elements, it can be written on one line without a closing element by putting a **/** before the closing **>**.:

```
<element_name argument1=”value” argument2=”value”/>
```

The following is an overview of the most important configuration elements:

Table 2-4

Element	Description
<server ...> **</server>**	This is the top level element, also called the *root element*. All other elements are nested in the server element. The following are the most important attributes: ■ **port**. This sets the port on which the server is listening for the shutdown command. The server will only accept shutdown commands that are sent from the same system Tomcat is running on. The default value is **8005**. ■ **shutdown**. This argument sets the string that needs to be sent to Tomcat in order to shut down the server. The default string is **SHUTDOWN**. If you have other users who can log in to the system running Tomcat, you might want to set this to a different value in order prevent normal users to shut down the server.
<GlobalNamingRessources> **</GlobalNamingRessources>**	The GlobalNamingResources element defines the global JNDI resources for the Server. JNDI is the acronym for the Java Naming and Directory Interface API. By making calls to this API, applications locate resources and other program objects. A resource is a program object that provides connections to systems, such as database servers and messaging systems.

Table 2-4 *(continued)*

Element	Description
<Ressource> **</Ressource>**	The Ressources element represents resources provided by the tomcat server. These resources can be used by web applications. In our example the Resource element is used to define a user database for the tomcat manager application.
<RessourceParams> **</RessourceParams>**	The RessourceParams element is used to group parameters that belonging to a resource. The attribute name is used to reference a previously defined resource.
<Parameter> **</Parameter>**	The Parameter element is nested in a RessourceParam element and is used to define parameters of a resource.
<service> **</service>**	This is a container element that holds **<connector>** and **<engine>** elements. The service element has an argument called **name** that gives the specified service a name. For example the name is used in the log output of the Tomcat server.
<connector/>	A connector is the part of a Tomcat system that handles the communication with clients. A connector is nested in a service element. There can be multiple connectors in one service element. The most important connector is the **http** connector, which handles the communication with web browsers. With the **port** attribute you can specify on which port Tomcat should listen for incoming http requests. Later in this section we will discuss the option to let Tomcat listen on ports below 1024.

Table 2-4 *(continued)*

Element	Description
<engine> **</engine>**	The engine receives and sends data from and to the connectors. It is responsible for the entire request processing machinery associated with a particular service. There is exactly **one** engine element nested in a service element. The attribute **name** is used to assign a name to the engine which is used in log and error messages. The attribute **defaultHost** determines the host that will process a request in case none of the configured hosts match the requested one. The value of defaultHost must match one of the configured host elements.
<host> **</host>**	The host element configures a virtual host in the Tomcat server. The virtual host mechanism allows you to host multiple hostnames on one physical system. This is very similar to a virtual host setup of the Apache web server. Multiple host elements can be nested in an **engine** element. The **name** attribute is used to configure the hostname the virtual host is responsible for. The **appBase** argument determines which directory the web applications of this host are installed in. The value is relative to **$CATALINA_HOME**.

Table 2-4 *(continued)*

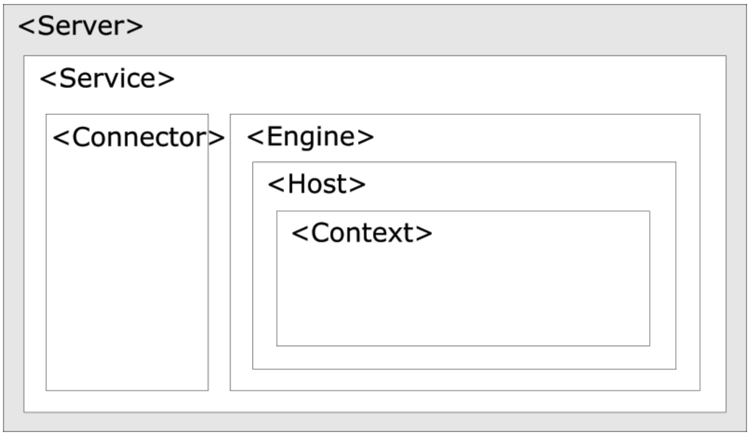
Element	Description
<context> **</context>**	The context element maps a URL path to a specific web application that is installed in a virtual host. Therefore, this element is nested in a host element. The **path** attribute is matched against the beginning of each requested URL and is used to determine if the context should be used. The **docBase** attribute points to the directory where the selected web application is installed. You can use an absolute path or one relative to the one specified in the **appBase** attribute of the related host element. For Tomcat 5.*x*, unlike Tomcat 4.*x*, it is **NOT** recommended to place the **<Context>** elements directly in the **server.xml** file. Instead, you should put the <Context> fragments in the **META-INF/context.xml** directory of your application or in the directory **/usr/share/tomcat5/conf/<EngineName>/<HostName>/**. You do not have to configure a context manually for every web application. Tomcat automatically creates a context for every application based on the name of the applications directory under **$CATALINA_HOME/webapps/**. An application that is installed under **$CATALINA_HOME/webapps/abcd** can therefore be accessed with the URL **http://<hostname>:<port>/abcd**
<logger/>	This element can be used in an engine or all other container elements to configure logging. As there are different logger implementations shipped with Tomcat, the **className** attribute is used to select an implementation.

Table 2-4 *(continued)*

Element	Description
<realm/>	This element refer to databases like usernames, passwords, and roles, which can be accessed within the Tomcat server.

The following is an illustration showing how the major elements are nested into each other:

Figure 2-8



This was an overview of the most important configuration elements and their attributes. For full documentation of all elements and attributes, see http://jakarta.apache.org/tomcat/tomcat-5.0-doc/config/index.html.

Exercise 2-8 Use a Configuration Template

In this exercise, you use the minimal configuration template as server.xml file.

You can find this exercise in the workbook.

(End of Exercise)

Objective 8 Install Web Applications

Web applications that are developed in Java are usually distributed in WAR files.

WAR stands for **W**eb**AR**chive, and the corresponding files end in **.war**

WAR files are compressed archives similar to zip archives.

The content must conform to the following structure:

- The static **HTML** files and **JSPs** are stored in the top level directory.
- The servlet and related Java technology class files must be stored in the **WEB-INF/classes** directory.
- Any auxiliary library JAR files must be stored in the **WEB-INF/lib** directory.
- The deployment descriptor is stored as a file named **web.xml** in the **WEB-INF** directory.

The **web.xml** file in the WEB-INF directory is also called **Deployment Descriptor**. It contains information about how an application should be deployed into the Tomcat environment. The file can also contain settings and start parameters for the web application.

If you are not sure how to configure a certain option of a web application, look at the web.xml file.

The easiest way to install an application is to use the auto deploy feature of Tomcat. This means that you just copy the .war file of the application in the /webApps directory of the virtual host that you want to use.

By restarting Tomcat, the WAR archive is automatically unpacked and the application is deployed. By default, you can access the application under the path name of the application directory.

Auto deploy requires the auto deploy feature to be switched on for the specified virtual host. This can be done in the host element of the Tomcat server.xml file. If you don’t configure auto deploy at all, it‘s switched on by default.

Exercise 2-9 Install an Example Application

In this exercise, you learn how to install an application using the auto deploy feature of Tomcat.

You can find this exercise in the workbook.

(End of Exercise)

Objective 9 Use Tomcat‘s Administration Tools

Tomcat comes with two web based applications that can be used to administer the server. Both are realized in Java and run with privileged rights on the Tomcat server.

The following tasks can be done with the administration tools:

- Use the Manager to Control Web Applications
- Use the Admin Interface to Adjust the Server Configuration
- Limit Access to the Administration Tools

Use the Manager to Control Web Applications

The Manager can be used to deploy and control web applications. The default virtual host **localhost** has the Manager configured in the context **/manager**.

The Manager offers functions that should not be accessible by everyone. Therefore, you have to authenticate yourself before you can access the Manager.

To get authentication working, you have to create a user in the file **/usr/share/tomcat5/conf/tomcat-users.xml**.

This file contains a user database that can be used to authenticate users for the Manager or other web applications.

The following two configuration elements are used in the file:

- **<role/>**. Permission rights are granted to authenticated users based on roles. The role element takes the attribute **rolename** which defines the name of the role. To control access to the manager application, define the role **manager** as in the following example:

```
<role rolename="manager"/>
```

- **<user/>**. Every user in the file is defined by a user element. The element takes the attributes **username**, **password**, and **roles**. The **password** parameter contains the password in clear text. The **roles** parameter is used to assign roles to the user. You can specify multiple roles by separating them with a comma.

  The following example creates the user **kbailey**, which has the role **manager**:

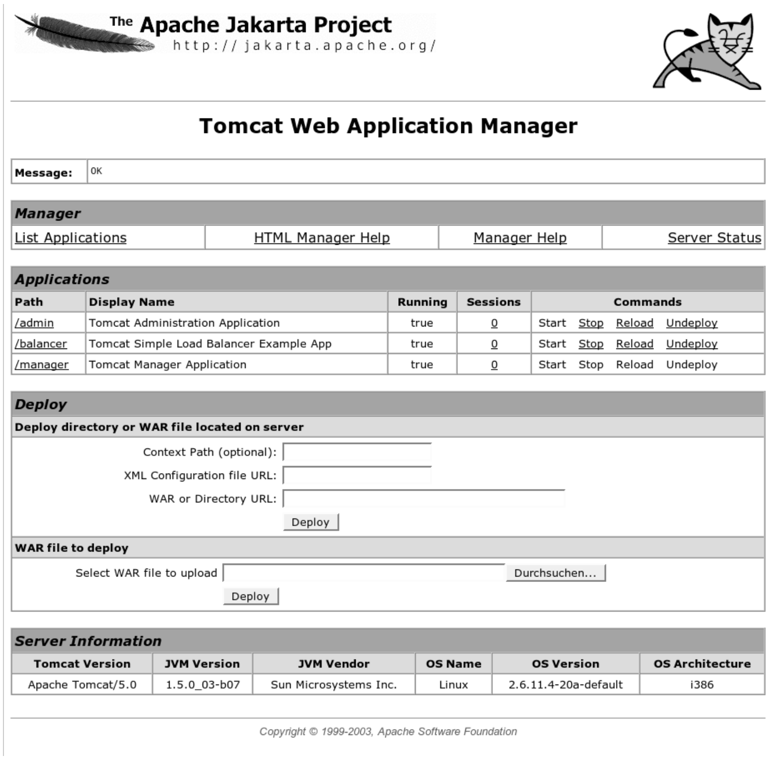
```
<user username="kbailey" password="kbailey" roles="manager"/>
```

After you have made changes on the tomcat-user.xml file, you have to restart Tomcat. Tomcat won't reread the file automatically.

After you have defined the manager role and assigned the role to a user, you can start the Manager by pointing your browser to **http://*hostname*:*port*/manager/html**

A password dialog appears. After you have entered the correct credentials, the following should appear in your browser:

Figure 2-9



The Apache Jakarta Project
http://jakarta.apache.org/

Tomcat Web Application Manager

Message:	OK

Manager

List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Display Name	Running	Sessions	Commands
/admin	Tomcat Administration Application	true	0	Start Stop Reload Undeploy
/balancer	Tomcat Simple Load Balancer Example App	true	0	Start Stop Reload Undeploy
/manager	Tomcat Manager Application	true	0	Start Stop Reload Undeploy

Deploy

Deploy directory or WAR file located on server

Context Path (optional):
XML Configuration file URL:
WAR or Directory URL:
Deploy

WAR file to deploy

Select WAR file to upload Durchsuchen...
Deploy

Server Information

Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture
Apache Tomcat/5.0	1.5.0_03-b07	Sun Microsystems Inc.	Linux	2.6.11.4-20a-default	i386

Copyright © 1999-2003, Apache Software Foundation

At the top of the page you can access different functions of the Manager.

The default selection is **List Applications**.

With **Server Status** you get an overview of the running Tomcat modules.

When you are on the List Application page, you can see all applications that are currently installed on the server in the second table.

In the **Commands** column you find links, which can be used to **Start**, **Stop**, **Reload**, or **Remove** installed applications. This is very useful, because all this can be done without restarting Tomcat.

In the next table you can install new applications on the server. You can either select an **application directory** or a **.war file** that is already on the server, or upload a **.war** file to the server.

In the last table you can see the version numbers of the used Tomcat components.

For more information about the Manager, you can select one of the **Help Links** in the top navigation table.

Use the Admin Interface to Adjust the Server Configuration

The second administration tool that comes with Tomcat can be used to adjust settings of the server configuration. In the standard configuration, the Admin Interface can be accessed with the following URL:

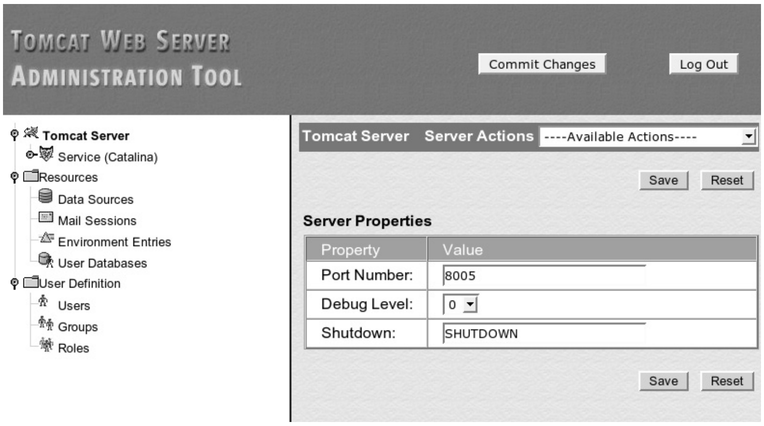
http://*hostname*:*port*/admin

Like the Manager, the Admin Interface requires user authentication. You need to create a role with the name **admin** and assign this role to the user who should be allowed to use the Admin Interface.

How to create a role and a user is described in the previous section about the Manager.

After entering the correct username and password, the following page appears:

Figure 2-10



Server Properties

Property	Value
Port Number:	8005
Debug Level:	0
Shutdown:	SHUTDOWN

On the left-hand side you can browse the server configuration and after you have selected an item, you can change the corresponding values on the right-hand side.

The configuration is displayed in a tree structure with the following top elements:

- **Tomcat Server**. This part of the tree follows basically the structure of the server.xml file of the running Tomcat instance.
- **Resources**. You can configure resources like user databases or data sources used by the server.
- **User Definition**. You can edit the user-related part of the configuration.

After you have made your changes, select **Commit Changes** at the top of the Admin tool.

Limit Access to the Administration Tools

When you run Tomcat as an Internet service, you do not want everyone to be able to access the administration tools.

You can limit the access using the **<valve>** element in the context configuration.

This needs to be done in the context files

/usr/share/tomcat5/conf/Catalina/localhost/manager.xml

for the Manager tool and

/usr/share/tomcat5/conf/Catalina/localhost/admin.xml

for the Admin tool.

In the Context element, you need to insert a **<valve>** element, like in the following example:

```
<Context path="/manager"
docBase="/usr/share/tomcat5/server/webapps/manager"
        debug="0" privileged="true">

  <!-- Link to the user database we will get roles from -->
  <ResourceLink name="users" global="UserDatabase"
                type="org.apache.catalina.UserDatabase"/>

        <Valve className="org.apache.catalina.valves.RemoteAddrValve"
                allow="127.0.0.1"/>

</Context>
```

In this example, access to the Manager is only allowed from localhost (IP address 127.0.0.1). For more entries use a comma to separate them.

Exercise 2-10 Enable the Manager and Admin Tools

In this exercise, you enable the manager and the admin tool.

You can find the exercise in the workbook.

(End of Exercise)

Objective 10 Use Port 80 to Access Tomcat

Tomcat accepts requests on the port that is configured in the <connector> elements of the server.xml configuration file. A very commonly used port is 8080.

If Tomcat is used standalone, it makes sense to configure port 80, because this is the default port for the HTTP Protocol.

The problem here is that on Linux-only processes running with root permissions are allowed to open ports below 1024. These ports are also called the **privileged ports**.

By default Tomcat runs as the user **tomcat** on a SUSE Linux Enterprise Server 10 system. This setup does not allow Tomcat to open a privileged port. It is possible to run Tomcat as root user, but this is not recommended due to security reasons.

In a setup where Tomcat runs as root user, an exploit of Tomcat would automatically lead to root access on the underlying server system.

To let Tomcat process requests arriving on port 80 in a secure way, you can use one of the following options:

- Use a port forwarder like **rinetd** to forward incoming requests on port 80 to the port where Tomcat is listening.
- Use **iptables** to forward incoming requests on port 80 to the port where Tomcat is listening.
- Integrate Tomcat with the Apache web server.

  This might also make sense if your content contains a lot of static HTML pages and only some dynamically-generated pages.

  Letting Apache serve the static content can lead to a better overall performance.

  You can find more information about the pro and cons of the Apache integration at http://jakarta.apache.org/tomcat/faq/connectors.html#integrate,

- Use the **Squid** cache to forward requests to Tomcat.

To use **rinetd** for port forwarding, you can use a configuration like the following when rinetd is installed on the same system like Tomcat.

The configuration file of rinetd is done using the file /etc/rinetd.conf:

```
10.0.0.1 80 10.0.0.1 8080
allow 10.0.0.*
logfile /var/log/rinetd.log
logcommon
```

In this example, all incoming requests on IP address **10.0.0.1** and **port 80** are forwarded to **10.0.0.1** port **8080**.

You have to stop the Apache daemon (httpd) if you want to use rinetd, because it is not possible that both use port 80 together.

More information about rinetd, see in the directory **/usr/share/doc/packages/rinetd** after you have installed the package **rinetd**.

Exercise 2-11 Configure rinetd to Forward Port 80 to Port 8080

In this exercise, you configure rinetd to forward port 80 to port 8080.

You can find this exercise in the workbook.

(End of Exercise)

Summary

Objective	Summary
1. Setup a Basic Web Server	■ Apache is the leading web server software. ■ Apache delivers data to a web browser using the HTTP protocol. ■ For a basic web server, you need to install the following packages: ■ apache2 ■ apache2-prefork ■ apache2-example-pages ■ The locally running web server can be accessed using the address **http://localhost**. ■ The default document root of the web server is /etc/www/htdocs. ■ The Apache configuration files are located in the directory /etc/apache2. ■ The options of the Apache configuration files are called directives. ■ You can check the syntax of the configuration file with the command **apache2ctl configtest**.
2. Configure Virtual Hosts	■ By configuring virtual hosts you can host multiple domains on one physical machine. ■ You need to create a configuration file in the directory /etc/apache2/vhosts.d/ for every virtual host.

Objective	Summary
3. Limit Access to the Web Server	■ You can limit the access to the Apache web server. ■ On an IP address basis. ■ Based on user authentication.
4. Configure Apache with OpenSSL	■ To encrypt the connection between the browser and server, you can configure Apache to use SSL. ■ To run a production system under SSL, you need a certificate signed by a CA. ■ To access an SSL-enabled system, use an address starting with https://.
6. Describe Tomcat	Tomcat is a servlet container for web applications developed in Java. It provides a framework for deploying and running Java servlets and JSP files. Like the Apache web server, Tomcat is developed by the apache foundation. You can find further information an complete documentation on: http://jakarta.apache.org/tomcat

Objective	**Summary**
7. Install and Configure Tomcat	The CATALINA_HOME variable points to the root of the Tomcat installation. On SUSE Linux Enterprise Server 10, this is **/usr/share/tomcat**. CATALINA_HOME is passed to Tomcat by the start scripts. On SUSE Linux Enterprise Server 10, you can set CATALINA_HOME in the file **/etc/sysconfig/j2ee**. The main server configuration is located in the file **/usr/share/tomcat5/conf/ server.xml** The file is written in an XML format and contains several configuration elements. The root element is **<server>** and all other elements are nested in it.
8. Install Web Applications	Web applications are located in the directory **/usr/share/tomcat5/webapps** When using the auto deploy feature of Tomcat, can copy a .war file into this directory and restart Tomcat. Tomcat will then unpack the .war file and deploy the application. By default applications can be access by entering an address following this scheme: **http://*host*:*port*/*name_of_ application_dir*>**

Objective	Summary
9. Use Tomcat's Administration Tools	The **Manager** and the **Admin** web interfaces can be used to administer Tomcat with a web browser. The tools can by default be accessed under the path names **/manger/html** and **/admin** Both tools require authentication. You need to create the roles **manager** and **admin** in the file **/usr/share/tomcat5/conf/ tomcat-users.xml** Then you have to assign the roles to the users which should be allowed to access the administration tools.
10. Use Port 80 to Access Tomcat	Due to security reasons, Tomcat should not be run with the privileges of the root user. Therefore, Tomcat can not open ports lower than 1024. To let Tomcat listen on port 80 (HTTP port) you can redirect port 80 to the one Tomcat is listening on using **rinetd** or **iptables**. You can also connect Tomcat with the Apache web server or use the Squid Cache to redirect requests to Tomcat.

SECTION 3 Configure and Use Samba

In this section, you will learn how to configure a file server for Windows hosts using Samba and how to use some more advanced features of Samba.

Objectives

1. Understand Samba
2. Configure a Simple File Server
3. Configure User Authentication
4. Use Samba‘s Client Tools
5. Use Samba as a Domain Controller
6. Integrate Samba in a Windows Domain
7. Configure Samba as Print Server

Objective 1 Understand Samba

Samba is a suite of applications used to integrate a Linux system into a Windows network. Samba is most commonly used as a file server for Windows hosts.

The Server Message Block (SMB) protocol is a network protocol that provides file and print services in a Windows network. Samba enables Linux to use SMB so that Linux can be integrated in a Windows environment.

You can use Samba for the following purposes:

- To provide file and print services for Windows clients.
- To access SMB file and print services on a Linux system.
- To act as a domain controller for Windows clients.

SMB services are provided by the NetBIOS protocol. NetBIOS makes its own name space available, which is completely different from the domain name system.

This name space can be accessed with the Unique Naming Convention (UNC) notation: all services provided by a server are addressed as **\\Server\Servicename**.

File or print services offered by a server are also called *shares*.

The server side of Samba consists of 2 parts:

- **nmbd.** This daemon handles all NetBIOS-related tasks.
- **smbd.** This daemon provides file and print services for clients in the network.

To integrate Linux as client in a Windows environment, Samba provides 2 tools:

- **winbind.** This daemon integrates a Linux system into a Windows authentication system (Active Directory).
- **nmblookup.** This tool can be used for NetBIOS name resolution and testing.

- **smbclient.** This tool provides access to SMB file and print services.

SUSE Linux Enterprise Server 10 (original version without updates or service packs) includes Samba version 3.0.22. An important new feature in this version is the Kerberos support in winbind. This allows a Kerberos based integration into Active Directory domains.

Novell is an important contributor of the Samba project. You can find more information about the Novell/SUSE Samba packages and the Novell/SUSE Samba team under **http://www.opensuse.org/samba**.

Objective 2 Configure a Simple File Server

To set up a simple file server with Samba, you need to:

- Install Samba
- Understand Samba‘s Configuration File

Install Samba

The following packages need to be installed for basic Samba setup:

- **samba.** This is the main Samba package. It contains the Samba server software.
- **samba-client.** This package contains the Samba client tools.
- **samba-doc (optional).** This package provides additional documentation about Samba.

After the packages have been installed, you can start the 2 Samba daemons with the following commands:

```
rcnmb start
rcsmb start
```

To start the Samba services automatically when the system is booting, enter the following commands:

```
insserv nmb
insserv smb
```

Exercise 3-1 Install Samba

In this exercise, you learn how to install the Samba components.

You can find this exercise in the workbook.

(End of Exercise)

Understand Samba's Configuration File

The Samba services are configured in the file **/etc/samba/smb.conf**.

The options in the this file are grouped into different sections. Each section starts with a keyword in square brackets.

To set up a simple file server with Samba, do the following:

- Create a Section for the General Server Configuration
- Create a Section for the Files to be Shared

Create a Section for the General Server Configuration

The section for the general server configuration starts with the keyword [global]. The following is an example of a basic global section:

```
[global]
    workgroup = DigitalAirlines
    netbios name = Fileserver
    security = share
```

The entries of the global section in this example are described below:

- **workgroup = DigitalAirlines**

  This line sets the Windows workgroup of the Samba server (in this case, DigitalAirlines).

- **netbios name = Fileserver**

  This line sets the name of the system in the NetBIOS name space (in this case, Fileserver).

- **security = share**

  This line determines how a client has to authenticate itself when accessing a share. This option can have the following values:

- **share.** The client does not need to provide a password when initially connecting to the server. However, a password might be necessary when the client tries to access a share.
- **user.** The client needs to provide a user name and password when connecting to the server. Samba validates the password against the users available on the Linux system and its own password file.
- **server.** The client needs to provide a user name and password when it connects to the server. Samba contacts another SMB server in the network to validate the password.
- **domain.** The client needs to provided a user name and password when connecting to the server. Samba connects to the domain controller and validates the password. This works only if Samba joins a Windows domain.
- **ads.** Samba acts as domain member of an ADS realm to validate the user name and password.

You might need to configure additional settings for these options to work correctly. For more information, see the man page of smb.conf.

Create a Section for the Files to be Shared

After the global section, you need to add a section for the share of your file server. The following example is the simplest way to set up a share:

```
[data]
    comment = Data
    path = /srv/data
    read only = Yes
    guest ok = Yes
```

The entries of the section in this example are described below:

- **[data]**

This is the identifier for the share. The share can later be accessed with the address \\Fileserver\data.

- **comment = Data**

  This option is a comment with additional information about the share. The comment is displayed when you browse the network with Windows Explorer.

- **path = /srv/data**

  This option sets the path to the exported data on the local file system. You have to make sure that the local user who needs to access the files of this share has sufficient file system rights.

- **read only = Yes**

  If this option is set to yes, the client accessing the share is not allowed to modify, delete or create any files. This is also the default value: If you leave out this parameter, a share is read-only.

- **guest ok = Yes**

  If this option is set to Yes, a password is not required to access the share.

There many more configuration options available than those discussed in this section. For an overview of all options, see the man page of smb.conf.

After you have created a smb.conf file, you should restart the Samba server daemons.

Before you restart the daemons, you can test the syntax of the Samba configuration file by entering the following command:

testparm

The output of the command looks like the following:

```
Load smb config files from /etc/samba/smb.conf
processing section "[data]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions
```

In this case, no errors are found. If there were any errors in the file, the command would display the errors grouped by configuration sections.

An interesting option for testparm is **--section-name <sectionname>**, which outputs only the specified section. This can be very usefull when you have a very long smb.conf.

Objective 3 Configure User Authentication

In the previous example, the Samba share is accessible without supplying a user name and password. In most cases, this type of accessibility in not recommended.

The following shows you how to configure Samba to require authentication with a user name and password:

- Prepare the Server for User Authentication
- Configure a Share that Is Accessible to Only One User
- Configure Shared Access for a Group of Users
- Configure the Export of Home Directories

Prepare the Server for User Authentication

The first task is to change the security option in the smb.conf file to the following:

```
security = user
```

The value user for the option security forces user authentication when the client attempts to connect to the server.

In the following examples, the configuration is based on User Level Security. In this security level, the Windows-compatible encrypted password file is stored in the file /etc/samba/smbpasswd (by default).

Users who want to access SMB shares must first be created as Linux users. Then an SMB password needs to be set using the smbpasswd tool.

The following example sets a SMB password for the user tux:

smbpasswd -a tux

Smbpasswd prompts you to enter the password twice and confirms the setting of the password by displaying the following message:

```
New SMB password: novell
Reenter smb password: novell
Added user tux
```

To disable an account, use

smbpasswd -d *login*:

```
da2:~ # smbpasswd -d kbailey
Disabled user kbailey.
```

To reactivate a disabled account, use

smbpasswd -e *login*:

```
da2:~ # smbpasswd -e kbailey
Enabled user kbailey.
```

To remove a Samba user, you can use the command:
smbpasswd -x *login*

```
da2:~ # smbpasswd -x kbailey
Deleted user kbailey.
```

To change a Samba password, use the command

smbpasswd

The user himself does not need to add any options:

```
kbailey@da2:~> smbpasswd
Old SMB password:
New SMB password:
Retype new SMB password:
Password changed for user kbailey
```

The system administrator has to use **smbpasswd** with the login of the user:

```
da2:~ # smbpasswd kbailey
New SMB password:
Retype new SMB password:
Password changed for user kbailey.
Password changed for user kbailey.
```

Configure a Share that Is Accessible to Only One User

The following example configures a share that is accessible only for the user tux:

```
[tux-dir]
    comment = Tux Directory
    path = /srv/share
    valid users = tux
    read only = no
```

Each line of this share is described below:

- **comment = Tux Directory**

  This option sets the comment for the share.

- **path = /srv/share**

  This option sets the path to the share.

- **valid users = tux**

  This option lists all user who are allowed to connect to this share. User names have to be separated by commas. You can add an entire UNIX group with the syntax **@*group_name***. However, all the users of the UNIX group need accounts in the smbpasswd file.

- **read only = no**

  This option makes the share writable by setting the read only option to no.

Configure Shared Access for a Group of Users

The following example creates a share that is readable and writable for all users of the UNIX group *accounting*:

```
[accounting]
    comment = Accounting department
    path = /srv/share
    valid users = @accounting
    force user = tux
    force group = accounting
    read only = no
```

Compared to the previous example, the following lines are new or have changed:

- **valid users = @accounting**

  This line allows all users who are in the UNIX group *accounting* to access the shared folder.

- **force user = tux**

  This line forces the Samba server to perform all file operations in the shared folder as user tux. This ensures that all files in the shared folder are readable and writable for every user who is allowed to access the share.

- **force group**

  This line forces the Samba server to perform all file operations with the group accounting.

Configure the Export of Home Directories

The following example exports the home directory of all UNIX users of the Samba server. You need to add the users to the smbpasswd file before the setup works:

```
[homes]
    comment = Home Directories
    valid users = %S
    read only = No
    browseable = No
```

In this example, you must name the share homes. If Samba finds a share with this name in the configuration file, it is treated in a special way.

When a share is requested, Samba first scans the existing sections of the configuration file. If no section is found, Samba uses the requested share name as a user name and looks up the user in the local password file.

If the user is found and the correct password is supplied, Samba automatically creates a share for the home directory of the user.

The following describes the lines in the example:

- **valid users = %S**

  The %S macro sets the value of the valid users option to the name of the requested share.

- **read only = No**

  The exported home directory should be readable and writable for the authenticated user.

- **browseable = No**

  For security reasons, the share is not browseable.

To access an exported home directory, use the address ***//server_name/user_name***.

Exercise 3-2 Configure a Share for the User Geeko

In this exercise, you learn how to configure a basic samba share.

You can find this exercise in the workbook.

(End of Exercise)

Objective 4 Use Samba‘s Client Tools

Although the main purpose of Samba is to provide services for Windows clients, it also provides tools to access SMB shares from Linux. It doesn't matter if these shares are provided by Samba or a native Windows server.

You can perform 3 basic tasks with the Samba tools:

- Use nmblookup
- Use smbclient

Use nmblookup

With the tool nmblookup, you can resolve NetBIOS names into IP addresses. In the following example, the IP address for the Samba server with the NetBIOS name Fileserver is looked up:

nmblookup Fileserver

The output of the command looks like the following:

```
querying Fileserver on 10.0.0.255
10.0.0.1 Fileserver<00>
```

In the first line, nmblookup states that it queries the IP address with a broadcast to the address 10.0.0.255. In the second line, it displays the result of the query, in this case, address 10.0.0.1 for the system with the NetBIOS name Fileserver.

If the system you are querying is not in the same subnet as yours, the name cannot be resolved with a broadcast query. Instead, nmblookup uses a WINS server to resolve the name.

For more information, see the man page for nmblookup.

Use smbclient

With the smbclient tool, you can access SMB shares on the network. It's also a very useful tool to test a Samba server configuration.

You can perform 3 basic tasks with smbclient.

- Browse Shares Provided by a Server
- Access Files Provided by an SMB Server
- Print on Printers Provided by an SMB Server
- Mount SMB Shares into the Linux File System

Browse Shares Provided by a Server

To display the shares offered by an SMB server, enter a command such as the following.

smbclient -L //Fileserver

When smbclient asks for a password, press **Enter** to proceed.

The output of smbclient looks like the following:

```
Domain=[DigitalAirlines] OS=[Unix] Server=[Samba 3.0.22-SUSE]
        Sharename       Type      Comment
        ---------       ----      -------
        data            Disk      Data
        IPC$            IPC       IPC Service
        ADMIN$          IPC       IPC Service
Domain=[DigitalAirlines] OS=[Unix] Server=[Samba 3.0.22-SUSE]
        Server               Comment
        ---------            -------
        Workgroup            Master
        ---------            -------
        DigitalAirlines     Fileserver
```

smbclient first displays all available shares of the SMB server. Besides the shares you have configured in the smb.conf file, an SMB server always offers at least 2 other shares:

- **IPC$.** This share provides information about the other shares available on the SMB server.
- **ADMIN$.** On a Windows computer this share points to the directory where Windows itself is installed. This can be useful for administrative tasks. When Samba tries to emulate a Windows server, it also offers this share. However, it is not needed to administer a Linux server.

The lower part of the smbclient output gives some information about the workgroup of the system.

This command can also very be valuable for testing purposes. After you have set up a share, you can check the availability of the share with smbclient.

Some shares are not browseable without authentication. In this case, you can pass a user name to smbclient, as in the following:

```
smbclient -L //Fileserver -U tux
```

In the example, smbclient connects to the server with the user name tux and prompt for the corresponding password.

Access Files Provided by an SMB Server

The command to access a share on a server is similar to the command used to browse for available shares, but instead of supplying just the server name, the full path to the share needs to be supplied without the -L option.

In the following example, smbclient connects to the share data on the server Fileserver:

```
smbclient //Fileserver/data
```

In this case, it is not necessary to supply a user name because the share data is configured with the **guest ok = yes** option. A user name can be supplied with the -U option.

After smbclient has connected to a share, it displays the following prompt:

```
Smb: \>
```

Smbclient can be used like a command-line FTP client. The most important commands are the following:

- **ls.** Displays the content of the current directory.
- **cd.** Changes to a directory.
- **get.** Copies a file from the share to the current working directory.
- **put.** Copies a file to the share. The share must be writable to use this command.

Print on Printers Provided by an SMB Server

You can use smbclient to print on shared network printers. The basic syntax of a print command is shown in the following:

smbclient //Printserver/laser -c 'print letter.ps'

In this example, the file letter.ps is printed on a network printer accessed through the share laser of the SMB server Printserver.

You can also use the command **print** on the smbclient command line after you have connected to the server. The **-c** option performs the given command automatically after the connection to the server has been established.

Exercise 3-3 Access the Share of the User Geeko with smbclient

In this exercise, you learn how to access a share with smbclient.

You can find this exercise in the workbook.

(End of Exercise)

Mount SMB Shares into the Linux File System

Instead of accessing shared files with smbclient, you can mount a share into the file system like a hard disk partition or a CD-ROM drive.

The basic mount command is shown in the following:

```
mount -t cifs //Fileserver/data /mnt
```

In this example, the share data of the SMB server Fileserver is mounted into the directory /mnt. The option **-t cifs** is necessary to specify that the resource to be mounted is an SMB share.

If the share requires authentication, you can supply a username and password, as in the following:

```
mount -t cifs -o username=tux,password=novell
//Fileserver/data /mnt
```

Exercise 3-4 Mount Geeko's Share

In this exercise, you mount a Samba share on a Linux system.

You can find this exercise in the workbook.

(End of Exercise)

Objective 5 Use Samba as a Domain Controller

Samba can be configured to act as Windows NT4 style domain controller.

Before going through this section, note the following: To configure a domain controller with Samba, you need to have in-depth knowledge about the Windows domain concept and about Windows networking. This knowledge is out of the scope of this course. A very good collection of Samba Howtos and Examples can be found at:
http://www.samba.org/samba/docs/man/Samba-Guide/

To configure Samba as an NT4 style Primary Domain Controller, complete the following:

- Understand a Domain Controller
- Configure /etc/samba/smb.conf
- Create Workstation Accounts

Understand a Domain Controller

A domain controller is a system, that is used to manage the users for a group of networked computers. In the Windows terminology, such a group is called a domain.

The domain controller provides a user database, which is used to authenticate users when they login into one of the managed computers.

Besides user authentication, the domain controller also provides information about network shares that a user is allowed to access and other user policies.

Since Windows 2000, a domain controller is based on Microsoft‘s Active Directory service. With the introduction of Active Directory, the communication protocols between domain controller and host changed.

Samba can act as an NT4 Domain Controller. Windows 2000 and Windows XP Professional workstations are backward compatible to NT4 and can therefore also be used with Samaba. However, in this case some of the new Active Directory features are not available.

Configure /etc/samba/smb.conf

The following is a basic Domain Controller configuration in smb.conf.:

```
[global]

netbios name = da50
workgroup = berlin
security = user
passdb backend = ldapsam:ldap://da2.digitalairlines.com
logon script = %U.bat
domain master = yes
os level = 50
local master = yes
preferred master = yes
domain logons = yes

[netlogon]
path = /netlogon
```

The new options are described below:

- **passdb backend.** Specifies in which kind of database the user and group information is stored.

Enter one of the following:

- **smbpasswd**
- **tdbsam** (smbpasswd plus NT SAM)
- **ldapsam**

If there is a backup domain controller (BDC) in your network, you have to select **ldapsam** here and to specify the LDAP server.

The use of a non-LDAP backend SAM database is particularly problematic because domain members periodically change the machine trust account password. The new password is then stored only locally.

- **logon script**. Specifies a windows batch script, that is executed on the client machine when a user logs in. The variable %U is replaced with the user name. In this configuration every user has therefore an own login script. The scripts must be placed in the directory which is specified in the netlogon share.
- **domain master**. This option tells samba to be a domain controller.
- **local master**. This option allows Samba to participate in the election of the local master browser.
- **os level**. This integer value controls at what level Samba advertises itself for browse elections.

  You need a value of 32 (or higher) to become PDC.
- **preferred master**. If this option is set, the nmbd will force an election immediately.
- **domain logons.** This option defines, whether other systems are allowed to join a domain. For a Domain Controller, this needs to be set to yes.

Samba cannot act as Active Directory Server or Active Directory Primary Domain Controller.

Create Workstation Accounts

For each Windows workstation which is going to be a member of a domain you need a workstation account on the Samba server. The workstation account is used to establish a trust relationship and a secure connection between Domain Controller and Client.

To create local accounts for Windows NT workstations, use the command

smbpasswd -a -m *workstation*

Workstation accounts are created with a “$” at the end of the name.

There must be a user account on Linux with this name before you enter the command.

See the following example:

```
da2:~ # smbpasswd -a -m da50
User da50$ does not exist in system password file (usually /etc/passwd).
Cannot add account without a valid local system user.
Failed to modify password entry for user da2$
da2:~ # useradd da51$
da2:~ # smbpasswd -a -m da51
Added user da51$.
da2:~ # cat smbpasswd
# Sample smbpasswd file.
# To use this, set 'encrypt passwords = yes' in the [global]-section
# of /etc/samba/smb.conf
kbailey:1008DD098F35B3B42417306D272A9441BB:6478A80295C56CEBBBC1A6FDEE3709
71:
[U]:LCT-39C8CC96:
da50$:101:6AD9466E87D89AAD3B435B51404EE:F05822FA2A7B7166AF98CF16CFA5FFDF:
[W]:LCT-39CB1383:
```

Objective 6 Integrate Samba in a Windows Domain

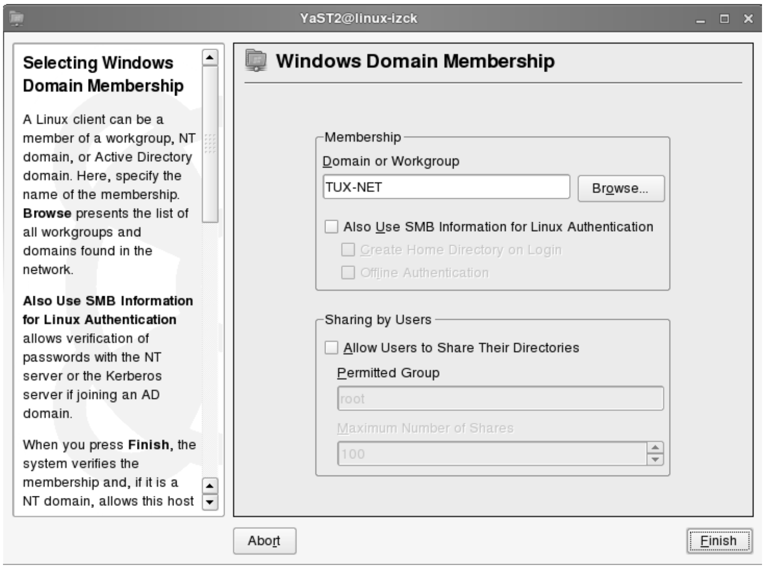
SUSE Linux Enterprise Server 10 comes with a new YaST module, which helps you to integrate a Linux system in a Workgroup, Windows NT Domain or Active Directory Domain.

The support for Active Directory has been greatly improved in SUSE Linux Enterprise Server 10. The winbind deamon now supports Kerberos, which provides an up to date way to authenticate against Active Directory.

The module can be found in the YaST Control Center under **Network Services > Windows Domain Membership**.

When you start the module, it looks like the following:

Figure 3-1



YaST2@linux-izck

Selecting Windows Domain Membership

A Linux client can be a member of a workgroup, NT domain, or Active Directory domain. Here, specify the name of the membership. **Browse** presents the list of all workgroups and domains found in the network.

Also Use SMB Information for Linux Authentication allows verification of passwords with the NT server or the Kerberos server if joining an AD domain.

When you press **Finish**, the system verifies the membership and, if it is a NT domain, allows this host

Windows Domain Membership

Membership

Domain or Workgroup: TUX-NET

- [ ] Also Use SMB Information for Linux Authentication
- [ ] Create Home Directory on Login
- [ ] Offline Authentication

Sharing by Users

- [ ] Allow Users to Share Their Directories

Permitted Group: root

Maximum Number of Shares: 100

In the text field at the top of the dialog, you enter the name of the Domain or Workgroup you would like to add the system to. You can also browse for available workgroups or domains by selecting **Browse**.

If you would like to use SMB information to authenticate Linux users, select the corresponding option. When activating this option, two more check boxes become active.

- **Create Home Directory on Login.** When you activate this option, a home directory is automatically created when a user logs in for the first time.
- **Offline Authentication.** This options allows authentication even when a system is not connected to a domain controller anymore. This can be a very useful option for portable computers.

After selecting **Finish**, you might be prompted for the Administration Credentials of the Domain Controller you are connecting to.

The configuration is written to **/etc/samba/smb.conf**.

Objective 7 Configure Samba as Print Server

Samba can be used as print server in a Windows network. This allows to share printers with other computers in the network.

To print with Samba, your first have to install a working CUPS printing system on your Samba server and configure the connected printers. This can be done with a YaST modul (see **Hardware > Printer** in the YaST Control Center).

Before you start to configure the printer shares in Samba, you should test the CUPS installation by printing from a local Linux application (such as the Firefox browser).

Once the CUPS system and the printers are configured, Samba can be used to share the CUPS printers with Windows workstations in the network.

On Linux, the standard print format that applications produce is PostScript. When a printer does not understand postscript directly, a printer driver in the CUPS system translates the data into the native format of the printer.

Windows uses Windows Metafile as its standard print format. When sharing printers with Samba, there are two ways how to convert this format into something a printer understands.

- Preprocess on the Samba Server
- Preprocess on the Windows Client

Preprocess on the Samba Server

The configuration depends on the number of printers you want to share.

You have to determine whether to:

- Share One Printer
- Share Many Printers

Share One Printer

If you want to provide only one printer for Windows, you can simply create a share.

The commands to create a share looks like the following:

```
[laserjet]
printable = yes
printer = lp_raw
path = /var/tmp
```

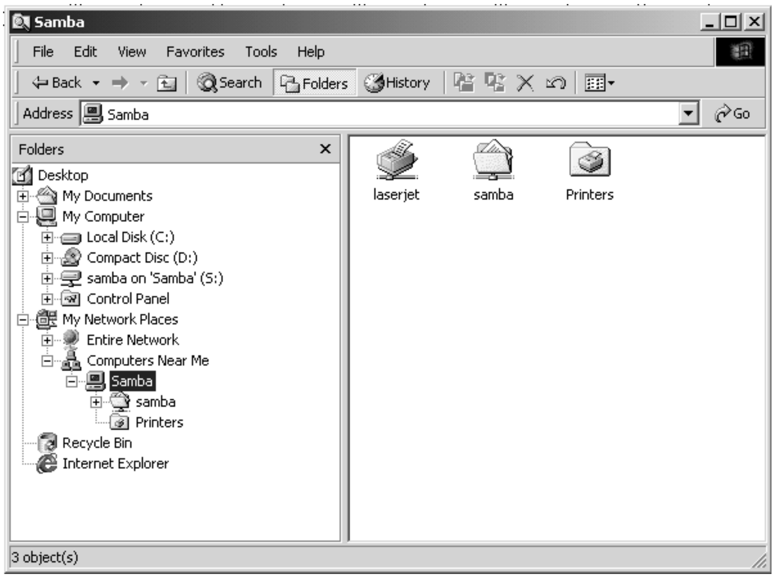
The option **printable = yes** tells Samba that this share is a printer share and not a shared directory.

Use the option **printer** to specify which Linux printer queue should be used via this share. If you want to preprocess on the client, enter the RAW queue here.

The spool directory is specified by the option **path**. The directory must be writeable for all users that are allowed to print.

Here, the print jobs are stored till they can be printed.

Figure 3-2



If you have problems to configure the printer share, use the option
print command = cp %s /tmp/out.ps
The information that arrives at the Samba server will be written to the file /tmp/out.ps.

Share Many Printers

If you want to install many printers on a Samba server being accessible from Windows, the method described above is not suitable.

Several printers can be configured in the following way.

Add the following lines in the [global] section if you use the CUPS printing system:

```
printing = CUPS
printcap name = CUPS
```

Using the option **printing**, you can specify the printing system.

The following values are valid:

- AIX
- BSD
- CUPS
- HPUX
- LPRNG
- PLP
- QNX
- SYSV

Use the parameter **printcap name** to specify the printer’s configuration file.

The printer shares can be configured in the [printers] section using the options introduced before.

The section will look like this, for example:

```
[printers]
printable = yes
path = /var/tmp
```

Preprocess on the Windows Client

If the print preprocessing is done by the clients, you must install the printer driver on each client manually.

To do this you need the Windows Installation CD.

It is better to store the available drivers on the Samba server so they will be accessible on the whole network.

To do this, you first need a printer share in the file /etc/samba/smb.conf.

The following is an example:

```
[laserjet]
printable = yes
print command = lpr -Plp -I %s
```

A special share labeled [print$] is also needed. Specify the path where the Windows drivers should be stored.

This share should be writable only for the administrator and readable for all users.

The [print$] share looks like the following:

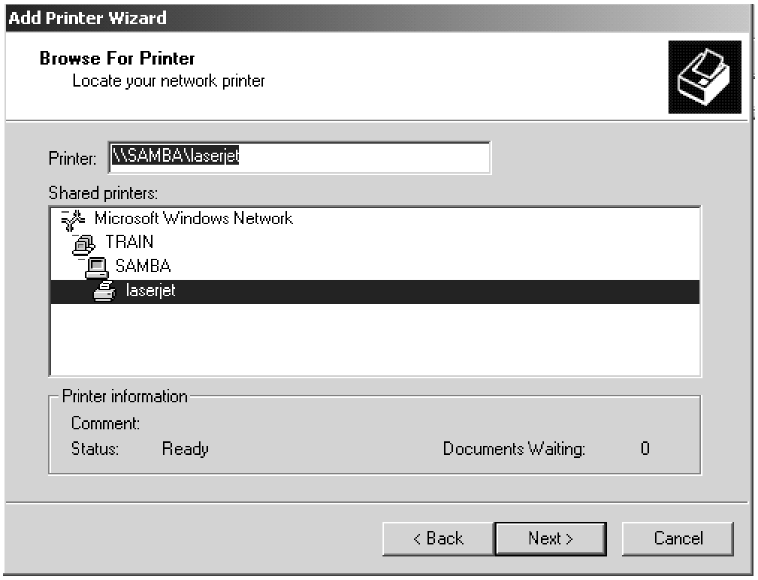
```
[print$]
path=/var/lib/samba/drivers
write list = root
```

Now you can install the driver on your Windows client.

If you select **Add Printer** in the printer configuration, the Add Printer Wizard will start.

If you browse for printers, your printer share should be found.

Figure 3-3



Printer: \\SAMBA\laserjet

Status: Ready   Documents Waiting: 0

Some steps later, you can select your printer model. After the driver installation, Samba creates new directories for different Windows versions using **path** in the [print$] section of the specified directory.

Here, you can find the installed printer driver (HP2100_6.PPD in the following example):

```
da2: # ls /var/lib/samba/drivers/
. .. W32ALPHA W32MIPS W32PPC W32X86 WIN40
da2: # ls /var/lib/samba/drivers/W32X86/
. .. 2
da2: # ls /var/lib/samba/drivers/W32X86/2/
. .. HP2100_6.PPD PSCRIPT.DLL PSCRIPT.HLP PSCRPTUI.DLL
```

Exercise 3-5 Configure Samba as a Print Server

In this exercise, you configure a Samba Print Server.

You can find the exercise in the workbook.

(End of Exercise)

Summary

Objective	Summary
1. Understand Samba	Samba can be used to integrate a Linux system into a Windows environment. Windows services are delivered using the SMB protocol. The network protocol NetBIOS is used in a Windows environment. NetBIOS creates its own name space independently from DNS. An SMB share can be accessed with the address schema \\server_name\service_name. Samba can be used for the following purposes: ■ As a file and print server ■ To access SMB shares ■ As a domain controller
2. Configure a Simple File Server	The Samba server is configured in the file /etc/samba/smb.conf. The Samba configuration file is structured in sections. You can check the syntax of the configuration file with the command **testparm**.

Objective	Summary
3. Configure User Authentication	Use the **security** option to specify the level of authentication: ■ **share**. A password is assigned to each share. ■ **user**. Before a user can access any share, he has to authenticate. ■ **server**. This is like the user authentication, but the authentication is done by a SMB server. ■ **domain**. The authentication is done by a PDC. ■ **ads**. The Samba server is part of an Active Directory domain. The option **guest ok = yes** allows guest users to connect to a server or to a share. **encrypt passwords = yes** activates the password encryption. User accounts are managed by the command **smbpsswd**: ■ **-a *login***. Add user account. ■ **-d *login***. Disable user account. ■ **-e *login***. Reactivate disabled user account. ■ **-x *login***. Remove user account. To change a Samba password, you also use the command **smbpasswd**.

Objective	Summary
4. Use Samba's Client Tools	Use **nmblookup** to resolve NetBIOS names to IP addresses. Use **smbclient** to access shares from the command line. Use **mount -t cifs** to mount SMB shares into the Linux file system. You can limit access to a Samba server with user authentication.
5. Use Samba as a Domain Controller	Set the **domain master** option so that nmbd becomes a domain master browser for its workgroup. The **local master** option allows Samba to participate at the election of the local master browser. The **os level** option controls at what level Samba advertises itself for browse elections. The **preferred master** option will force an election immediately. To create local accounts for Windows NT workstations, use the command **smbpasswd -a -m *workstation***
6. Integrate Samba in a Windows Domain	SUSE Linux Enterprise Server 10 comes with a new YaST module, which allows an easy integration into Windows domains and workgroups. The module can be found in the YaST Control Center under **Network Services > Windows Domain Membership**.

Objective	Summary
7. Configure Samba as Print Server	There are two ways to preprocess printer information: ■ Preprocess on the Samba server: ■ Share one printer ■ Share many printers ■ Preprocess on the Windows client Use the **printable = yes** option to tell Samba, that this share is a printer share and not a shared directory. Use the **printer** option to specify which Linux printer queue should be used by this share. Use the **path** option to specify the spool directory. Use the **printing** option to specify the printing system. Use the **printcap name** parameter to specify the printer's configuration file. In the [print$] share, you can specify the path where the Windows drivers should be stored.

SECTION 4 Enable Fundamental Network Services

In this section you learn the basics of enabling some of the more commonly-used network services available in SUSE Linux Enterprise Server 10.

Objectives

1. Enable the Extended Internet Daemon (xinetd)
2. Enable an FTP Server
3. Configure Time on SUSE Linux Enterprise Server 10
4. Configure NFS (Network File System)

Objective 1 Enable the Extended Internet Daemon (xinetd)

In this objective you learn how to enable the extended internet daemon (xinetd) by reviewing the following:

- What xinetd Is
- Configure xinetd with YaST
- Manage xinetd Manually

What xinetd Is

Services can either run standalone, meaning they listen on a port themselves, or they can be run via the “super daemon” xinetd. In this case, the super daemon acts as a mediator of connection requests for a series of services. It accepts the connection requests, starts the required service, and passes the request to the newly started server process.

If the connection between the client and the server is terminated, the server process started by xinetd is removed from memory.

Starting services through xinetd has both advantages and disadvantages. The most significant advantage is saving resources (especially memory), since a server process is only started when it is needed. A disadvantage, however, is that a delay occurs while the required service is loaded, started, and connected.

As a rule, you only want to use xinetd for services that are occasionally (not permanently) needed on the server. Some of the services run traditionally by xinetd include Telnet and FTP.

For detailed information about xinetd, enter **man 8 xinetd**.

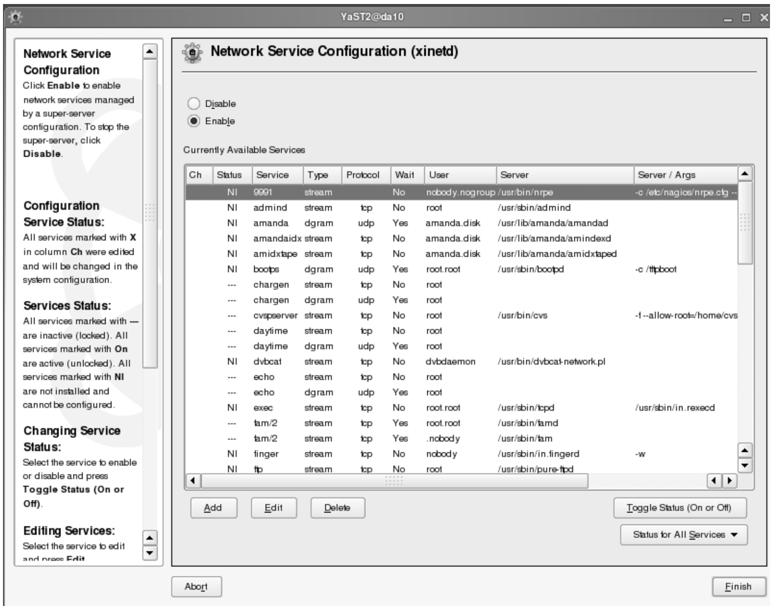
Configure xinetd with YaST

To configure the services mediated by xinetd, you can use the YaST Network Services (xinetd) module. Start the **YaST Control Center** and then select **Network Services > Network Services (xinetd)**. Or open a terminal window, **su -** to root and then enter **yast2 inetd**.

The YaST module to configure xinetd is called inetd. The reason for this is that in the past the default super daemon on SUSE Linux was inetd, not xinetd.

Enable the xinetd super daemon by selecting **Enable**. This activates the Currently Available Services list. You can add, edit, or delete services in the list:

Figure 4-1



Network Service Configuration
Click **Enable** to enable network services managed by a super-server configuration. To stop the super-server, click **Disable**.

Configuration Service Status:
All services marked with **X** in column **Ch** were edited and will be changed in the system configuration.

Services Status:
All services marked with — are inactive (locked). All services marked with **On** are active (unlocked). All services marked with **NI** are not installed and cannot be configured.

Changing Service Status:
Select the service to enable or disable and press **Toggle Status (On or Off)**.

Editing Services:
Select the service to edit and press **Edit**

Network Service Configuration (xinetd)

( ) Disable
[● Enable]

Currently Available Services

Ch	Status	Service	Type	Protocol	Wait	User	Server	Server / Args
	NI	9991	stream		No	nobody.nogroup	/usr/bin/nrpe	-c /etc/nagios/nrpe.cfg --
	NI	admind	stream	tcp	No	root	/usr/sbin/admind	
	NI	amanda	dgram	udp	Yes	amanda.disk	/usr/lib/amanda/amandad	
	NI	amandaidx	stream	tcp	No	amanda.disk	/usr/lib/amanda/amindexd	
	NI	amidxtape	stream	tcp	No	amanda.disk	/usr/lib/amanda/amidxtaped	
	NI	bootps	dgram	udp	Yes	root.root	/usr/sbin/bootpd	-c /tftpboot
	---	chargen	stream	tcp	No	root		
	---	chargen	dgram	udp	Yes	root		
	---	cvspserver	stream	tcp	No	root	/usr/bin/cvs	-f --allow-root=/home/cvs
	---	daytime	stream	tcp	No	root		
	---	daytime	dgram	udp	Yes	root		
	NI	dvbcat	stream	tcp	No	dvbdaemon	/usr/bin/dvbcat-network.pl	
	---	echo	stream	tcp	No	root		
	---	echo	dgram	udp	Yes	root		
	NI	exec	stream	tcp	No	root.root	/usr/sbin/tcpd	/usr/sbin/in.rexecd
	---	fam/2	stream	tcp	Yes	root.root	/usr/sbin/famd	
	---	fam/2	stream	tcp	Yes	.nobody	/usr/sbin/fam	
	NI	finger	stream	tcp	No	nobody	/usr/sbin/in.fingerd	-w
	NI	ftp	stream	tcp	No	root	/usr/sbin/pure-ftpd	

To manage the services available through xinetd (except for enabling services such as Telnet or FTP) requires a skill set beyond the objectives of this course. This is especially true of configuring services with **Edit**.

Notice that some services are off (---), while others are not installed (NI).

To configure a service select the service, then select **Toggle Status (On or Off)**.

If a service is not installed it will be installed. The word "On" appears in the Status column. An "X" appears in the Changed (Ch) column to indicate that the service has been edited and will be changed in the system configuration.

You can change the status of all installed services to on or off by selecting **Status for All Services > Activate All Services** or **Status for All Services > Deactivate All Services**.

When you finish configuring the services, save the configuration setting and start the inetd (or xinetd) daemon by selecting **Finish**.

Manage xinetd Manually

To manage xinetd manually, you need to know how to do the following:

- Start, Stop, and Restart xinetd
- Configure xinetd
- Configure Access Control
- Configure Log Files

Start, Stop, and Restart xinetd

To provide services through xinetd, you need to install and start the daemon on your SUSE Linux Enterprise Server. To have the daemon automatically activated at boot, enter **insserv xinetd**.

xinetd ist controlled by the script /etc/init.d/xinetd. /usr/sbin/rcxinetd is a link to this script. You can start and stop the daemon by entering **rcxinetd start** or **rcxinetd stop**. You can find out whether the daemon is activated or not by entering **rcxinetd status**.

Additionally, xinetd can be influenced by signals sent with kill or killall. The following table lists some of the signals that can be used with xinetd:

Table 4-1

Signal	Number	Description
SIGUSR1	10	Causes an internal state dump (the default dump file is /var/run/xinetd.dump).
SIGQUIT	3	Causes xinetd termination.
SIGTERM	15	Terminates all running services before terminating xinetd.
SIGHUP	1	xinetd re-reads the configuration file and stops listening on ports of services that are no longer available and/or binds to ports now available according to the new configuration..
SIGIO	29	Causes an internal consistency check to verify that the data structures used by the program have not been corrupted.

Configure xinetd

The configuration of xinetd is distributed across several files. /etc/xinetd.conf lists general options, while files in /etc/xinetd.d/ contain the configuration of specific services provided via xinetd. These files are included into the xinetd configuration by an include statement at the end of /etc/xinetd.conf.

To configure xinetd, you need to know the following:

- The File /etc/xinetd.conf
- The Directory /etc/xinetd.d/
- chkconfig
- Internal Services

The File /etc/xinetd.conf

In SUSE Linux Enterprise Server, the file /etc/xinetd.conf just contains general options, no service configurations. The following is the syntax of /etc/xinetd.conf for the default configuration parameters of xinetd:

```
defaults
     {
          key operator parameter parameter. . .
     }
```

Operators include **=**, **-=**, and **+=**. Most attributes (keys) only support the operator **=**, but you can include additional values to some attributes by entering **+=** or remove them by entering **-=**.

The defaults entry in the configuration file is optional and allows to set defaults such as the following:

```
defaults
{
        log_type        = FILE /var/log/xinetd.log
        log_on_success  = HOST EXIT DURATION
        log_on_failure  = HOST ATTEMPT
#        only_from       = localhost
        instances       = 30
        cps             = 50 10
}
includedir /etc/xinetd.d
```

The configurations for log_type and instances will be overwritten if something else has been defined in the individual service entries. For all other attributes, the default configurations are combined with the values set in the services.

The log_type statement can define whether (as in the example) the output is written directly to a log file (/var/log/xinetd.log) or forwarded to the daemon syslog (such as
log_type = SYSLOG authpriv).

If there are high security demands, you might want to consider leaving logging up to the syslog daemon in order to prevent potential unwanted access to the xinetd log file.

The keys **log_on_success** and **log_on_failure** configure what should be recorded in the log file, depending on whether the connection to a network service succeeds or fails.

The key **instances** can be used to limit the maximum possible number of daemons for each service, which protects the machine from either intentional or accidental overload due to too many simultaneous connections (denial-of-service attempts).

cps stands for connections per second. The first value (50) is the maximum number of connections per second that can be handled. the second value (10) is the wait period before accepting new connections after the maximum has been exceeded (helpful in preventing Denial of Service attacks).

The directive **includedir /etc/xinetd.d** prompts xinetd to search all files in the directory /etc/xinetd.d/ for the configuration of the services. The same attributes and the same syntax can be used as in /etc/xinetd.conf.

The Directory /etc/xinetd.d/

In the directory /etc/xinetd.d/, there is a separate configuration file for every service. The main advantage of splitting the configuration in several files is improved transparency.

The syntax for configuring network services in these files is similar to the one used for the options in /etc/xinetd.conf above:

```
service service_name
     {
          key operator parameter parameter. . .
          key operator parameter parameter. . .
     }
```

The following is an example of the configuration of finger:

```
# default: off
# description: The finger server answers finger requests.
# Finger is a protocol that allows remote users to see
# information such as login name and login time for
# currently logged in users.
service finger
{
        socket_type     = stream
        protocol        = tcp
        wait            = no
        user            = nobody
        server          = /usr/sbin/in.fingerd
        server_args     = -w
#        disable         = yes
}
```

The significance of the keywords in the example is as follows:

Table 4-2

Keyword	Description
socket_type	Refers to the type of socket (stream, dgram, raw, or seqpacket).
protocol	Refers to the protocol (usually tcp or udp) used by the corresponding network service. The protocol must be listed in the file /etc/protocols.
wait	Specifies whether xinetd must wait for the daemon to release the port before it can process further connection requests for the same port (Yes: single-threaded) or not (No: multithreaded).
user	Indicates under which user ID the daemon will start. The user name must be listed in the file /etc/passwd.
server	Specifies the absolute path name of the daemon to start.
server_args	Specifies which parameters to pass to the daemon when it starts.
disable	If set to yes, the service is disabled.

For a description of all possible parameters, enter **man xinetd.conf**.

Internal Services

Certain services (such as echo, time, daytime, chargen, and discard) are provided by xinetd itself without calling another program. These are called internal services and are labeled in the configuration as follows:

```
type = INTERNAL
```

Without this line xinetd assumes that external services are involved. With services such as echo, which are both TCP and UDP-based services, you not specify the respective socket_type, but you also need to identify the service in the id field in such a way that it is properly distinguished from other services.

The following two examples show this for echo:

```
# /etc/xinet.d/echo
# default: off
# description: An echo server. This is the tcp version.

service echo
{
        type            = INTERNAL
        id              = echo-stream
        socket_type     = stream
        protocol        = tcp
        user            = root
        wait            = no
        disable         = yes
}
```

```
# /etc/xinet.d/echo-udp
# default: off
# description: An echo server. This is the udp version.

service echo
{
        type            = INTERNAL UNLISTED
        id              = echo-dgram
        socket_type     = dgram
        protocol        = udp
        user            = root
        wait            = yes
        disable         = yes
        port            = 7
}
```

chkconfig

The program **chkconfig** can be used to list services covered by xinetd:

```
da10:~ # chkconfig -l
...
xinetd based services:
        chargen:            off
        chargen-udp:        off
        daytime:            on
        daytime-udp:        on
        echo:               off
...
```

It can also be used to turn services on and off:

```
da10:~ # chkconfig daytime
daytime  xinetd
da10:~ # chkconfig daytime off
da10:~ # chkconfig daytime
daytime  off
```

Configure Access Control

The daemon xinetd recognizes four parameters used to control access monitoring:

- **only_from.** With this parameter, you define which hosts can use which service. You can specify complete IP addresses from hosts or networks or just the network or host names.

  You can define this parameter in the defaults section or in the service section.

- **no_access.** With this parameter, you define which hosts are excluded from access. You can specify complete IP addresses from hosts or networks or just the network or host names.

  You can define this parameter in the defaults section or in the service section.

- **access_time.** You can use this parameter to define at which times the service is available (in 24-hour format).

  You can define this parameter in the defaults section or in the service section.

- **disabled.** You can use this parameter to completely shut off a server. This also applies to logging access attempts.

  The following is an example for the attribute disabled:

  ```
  disabled = finger
  ```

  With this setting, the service finger is switched off completely. If a computer tries to access the service, the attempt is not even logged.

  The parameter disabled can only be used in the defaults section. (Within a service section, the corresponding parameter to use is **disable.** Note the missing **d** at the end!)

The following is an example for the Telnet service:

```
# default: off
# description: Telnet is the old login server which is
#  INSECURE and should therefore not be used. Use secure
#  shell (openssh). If you need telnetd not to
#  "keep-alives" e.g. if it runs over an ISDN uplink,
#  add "-n". See 'man telnetd' for more details.
service telnet
{
        socket_type     = stream
        protocol        = tcp
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        server_args     = -n
        only_from       = 192.168.0.3  192.168.0.7
        only_from      += 192.168.0.10 192.168.0.12
        only_from      += 192.168.1.0/24
        no_access       = 192.168.1.10
        flags           = IDONLY
        access_times    = 07:00-21:00
#        disable = yes
}
```

These settings result in the following:

- Access is permitted for machines with the following IP addresses:

  192.168.0.3
  192.168.0.7
  192.168.0.10,
  192.168.0.12
  192.168.1.0-255

- Access is denied to the host with the IP address 192.168.1.10.
- The service is available from 7:00 A.M. to 9:00 P.M.

If you place high demands on access monitoring, you can tighten the security level even more by using the parameters INTERCEPT and IDONLY in the flags entry.

If the parameter USERID was set in the log_on_access and log_on_failure entries, IDONLY then makes sure that a connection to the network service is permitted only when the user identification service (such as identd) of the host requesting the network service issues the user ID.

If the parameter INTERCEPT has been entered as well, xinetd also attempts to make sure that an authorized host is on the other end of already existing connections—that the connection has not been intercepted.

However, connection monitoring does not function with multithreaded or internal xinetd services. In addition, it puts a heavy burden on the network connection and the performance of the network service.

Configure Log Files

Almost every hacker has to make several attempts and needs some time before achieving success. To protect your server, you not only need hacker-resistant software, but you need log files that the system administrator can use to detect unauthorized login attempts.

Because of this, it does not make sense to only deter unauthorized access attempts. To maintain optimal system security, you need to record failed and unauthorized connection attempts.

To shut off a service but still retain its logging functions, configure only_from without using any additional parameters (such as the following):

```
only_from     =
```

Logging through xinetd is controlled by the log_type statement along with the attributes log_on_success and log_on_failure.

These let you record from which host and for how long an access attempt was made, and which user was using the service (if the remote host supports this feature).

In addition, you can also log the circumstances of how and why the network service was used. However, even the best log does not mean much if you do not check it on a regular basis for failed connection attempts.

Exercise 4-1 Configure the Internet Daemon (xinetd)

In the first part of this exercise, use the YaST module Network Services (xinetd) to set up a telnet server on your computer.

In the second part, install vsftp if it is not yet installed, and edit its configuration in /etc/xinetd.d/ to activate the service.

You can find this exercise in the workbook.

(End of Exercise)

Objective 2 Enable an FTP Server

To enable an FTP server on SLES 10 you need to understand the following:

- The Role of an FTP Server
- How FTP Works
- Advantages of PureFTPd Server
- Install and Run PureFTPd Server
- Configure PureFTPd Server

The Role of an FTP Server

As the name indicates, the File Transfer Protocol (FTP) enables the transfer of files from one computer to another. Today, FTP is used mainly for file transfer on the Internet.

The basic features supported by FTP and available to the user are:

- Sending, receiving, deleting, and renaming files
- Creating, deleting, and changing directories
- Transferring data in binary or ASCII mode

An FTP server allows access after authentication against a password database. As a rule, these are the files /etc/passwd and /etc/shadow. Other authentication systems, such as NIS or LDAP, are possible.

An FTP server such as PureFTPd also supports authentication against its own password database, which is independent from the files /etc/passwd and /etc/shadow.

In addition, guest access can be set up as anonymous FTP (**aFTP**). Generally, users logging in to aFTP use **anonymous** or **ftp** as their user name and their email address as the password.

The address is normally not checked for correctness, although some servers check the syntax and require an entry in the format ***user@hostname.domain***. An anonymous user is normally given access to a restricted directory tree (a chroot environment).

How FTP Works

The FTP protocol uses the TCP transport protocol. FTP uses two TCP connections between the client and the server, one for commands and the other for data.

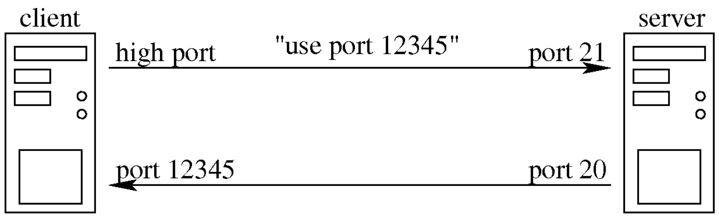
The first of these connections sends FTP commands from the client to the server. To begin an FTP session, the client addresses the FTP command channel on port 21 of the server. The client then sends its commands to the FTP server.

For the actual file transfer, or in response to certain commands like ls, FTP uses the second TCP connection, which is only created when a file is ready for transfer (for example, by a GET or PUT command).

There are 2 different types of data transfer:

- **Active data transfer.** The FTP client offers the FTP server an unprivileged TCP port for the data channel connection. The server then initializes the data channel from its port 20 to the port offered by the client.

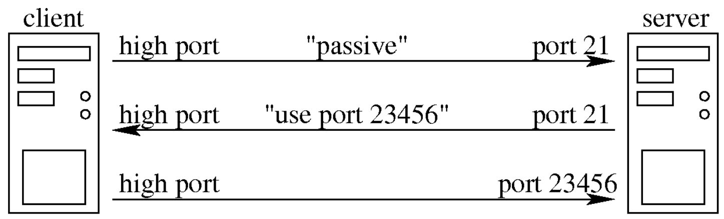
Figure 4-2



- **Passive data transfer.** The FTP client informs the FTP server that it wants to use a passive data transfer using the PASV command.

The FTP server then offers the FTP client an unprivileged TCP port for a data channel connection and the client initializes the data channel to the port offered by the server.

Figure 4-3



Passive FTP transfer avoids the need of having to allow incoming connections on the client. This makes it easier for firewall administrators to establish a secure configuration.

Advantages of PureFTPd Server

A number of FTP servers for Linux are available, such as the standard FTP server in.ftpd, the FTP server from Washington University wu.ftpd, proftpd, or the PureFTPd FTP server (pure-ftpd).

Although you can use other FTP servers to provide FTP network services, PureFTPd has several features that make it stand out from other FTP servers:

- Consistent use of chroot environments.
- Uncomplicated configuration of virtual FTP servers.
- Virtual users independent of the system users listed in the file /etc/passwd.
- Configuration via command line parameters or with a configuration file.

Install and Run PureFTPd Server

You can install the PureFTPd server with the YaST Software Management module by selecting the package pure-ftpd.

After installation, you configure the FTP server manually by editing the configuration file /etc/pure-ftpd/pure-ftpd.conf.

You can run PureFTPd server using one of the following methods:

- **From the command line.** To start PureFTPd from the command line, enter **pure-ftpd** ***options*** (such as **pure-ftpd -B -e**). If you start pure-ftpd this way, no configuration file is used.

  For details on the possible pure-ftpd options, enter **man pure-ftpd**.

- **From a start script.** To start PureFTPd, enter **/etc/init.d/pure-ftpd start** (or **rcpure-ftpd start**). To stop the PureFTPd service, enter **rcpure-ftpd stop**.

  The configuration file /etc/pure-ftpd/pure-ftpd.conf is parsed by the Perl script /usr/sbin/pure-config-args to translate the parameters in the configuration file to command-line options.

  These options are then passed to the daemon /usr/sbin/pure-ftpd.

  If you want pure-ftpd to be initialized upon startup, you need to set symbolic links by entering the following:

  insserv /etc/init.d/pure-ftpd

- **From xinetd.** If you want to start PureFTPd via xinetd, you need to edit the file /etc/xinetd.d/pure-ftpd and add the required options as in the following example:

```
# default: off
# description: The ftpd server serves FTP connections. It uses normal,
# unencrypted usernames and passwords for authentication.
# This ftpd is the pure-ftpd.
#   ** NOTE ** when using pure-ftpd from xinetd the arguments to control
#    it's behaviour should be added here in this file in the
#    "server_args" line since the configuration file
#    /etc/pure-ftpd.conf is only for standalone pure-ftpd.
#    The command "/usr/sbin/pure-config-args /etc/pure-ftpd.conf"
#    will print the arguments needed for behaviour like standalone
#    pure-ftpd.

service ftp
{
    socket_type = stream
    server = /usr/sbin/pure-ftpd
    server_args = -A -i
    protocol = tcp
    user = root
    wait = no
    disable = yes
}
```

Because the configuration file pure-ftpd.conf is not parsed or evaluated when PureFTPd is started via xinetd, all the required options must be given in the file /etc/xinetd.d/pure-ftpd as server arguments in the line server_args.

For details on all command line options for PureFTPd, enter **pure-ftpd --help**.

Configure PureFTPd Server

To perform basic configuration tasks for PureFTPd server, you need to know the following:

- Configure Anonymous FTP
- Configure FTP with Virtual Hosts for Anonymous FTP
- Configure FTP for Authorized Users
- Configure FTP with Virtual Users not Included in /etc/passwd

Configure Anonymous FTP

To configure anonymous FTP for PureFTPd, you need to have an FTP user and home directory (such as /srv/ftp/) in the file /etc/passwd (exists by default in SLES 10).

However (unlike other FTP servers), you do not need to create any subdirectories (such as bin) in the home directory.

The following is an example of a simple pure-ftpd.conf file:

```
# Cage in every user in his home directory
ChrootEveryone              yes

# Don't allow authenticated users - have a public anonymous FTP only.
AnonymousOnly               yes

# Disallow anonymous users to upload new files (no = upload is allowed)
AnonymousCantUpload         yes
```

In this configuration file, it is only possible to log in as an anonymous user, regardless of what user name is given. It is not possible to change to a directory above /srv/ftp/, and no files can be uploaded to the server—only downloads are possible.

The equivalent command on the command line would be **pure-ftpd -A -e -i**.

If you want anonymous users to upload files to the server, the configuration file would look like the following:

```
# Cage in every user in his home directory
ChrootEveryone              yes

# Don’t allow authenticated users - have a public anonymous FTP only.
AnonymousOnly               yes

# Allow anonymous users to upload new files
AnonymousCantUpload         no

# Disallow downloading of files owned by “ftp”, ie.
# files that were uploaded but not validated by a local admin.
AntiWarez                   yes

# Never overwrite files. When a file whose name already exists is
# uploaded, it gets automatically renamed to file.1, file.2, file.3, ...
AutoRename                  yes
```

The AntiWarez option is recommended because the server could otherwise be misused to handle undesirable (or even illegal) data.

Files uploaded to the server belong to the user ftp, but files of the user ftp cannot be downloaded from the server because of this option. The administrator must change the owner of the file (for instance to root) using the command chown before this is possible.

The last line ensures that a file that might already exist is not overwritten. Instead, a new file is created with a number on the end (such as **file.1**).

The equivalent command on the command line would be **pure-ftpd -A -e -s -r**.

Configure FTP with Virtual Hosts for Anonymous FTP

Virtual FTP hosts allow a number of FTP sites to be hosted on one machine (such as ftp.slc.digitalairlines.com and ftp.muc.digitalairlines.com). Each of these FTP sites requires its own IP address, because the FTP protocol cannot handle host names.

For this reason, you need to assign multiple IP addresses to your network card. In addition, you need to configure a service such as DNS that guarantees an IP address is matched correctly to a domain.

Instead of using the command line or the file pure-ftpd.conf, you configure a virtual host through the directory /etc/pure-ftpd/vhosts/. The configuration is avery simple 2-step process:

1. From the command line, use ip to create virtual network devices, as in the following example:

   ```
   ip address add 10.0.0.80/24 brd + dev eth0
   ip address add 10.0.0.81/24 brd + dev eth0
   ```

2. Create a symbolic link in /etc/pure-ftpd/vhosts/ with this IP address, which is linked to the directory with the files to offer over anonymous FTP at this address.

   The following is an example:

   ```
   cd /etc/pure-ftpd/vhosts/
   ln -s /ftp/directory/of/ftp.slc.digitalairlines.com 10.0.0.80
   ln -s /ftp/directory/of/ftp.muc.digitalairlines.com 10.0.0.81
   ```

To prevent these anonymous areas from being filled with undesired files, start PureFTPd with the option **-i**. This makes it impossible for anonymous users to upload files.

Virtual FTP servers only handle anonymous FTP users and not authorized users.

Configure FTP for Authorized Users

Configuring an FTP server for authorized users is important for those who are hosting web sites. Individual customers maintain their own pages in directories to which they alone have access.

The following is an example configuration in which no anonymous FTP access is allowed and where all users are limited to their home directory:

```
# Cage in every user in his home directory
ChrootEveryone              yes

# Disallow anonymous connections. Only allow authenticated users.
NoAnonymous                 yes
```

The equivalent command on the command line would be **pure-ftpd -A -E**.

If you want to modify the above configuration so that certain users are not held in a chroot environment (for example, members of a group ftpadmin with the GID 500), you could enter the following:

```
# Cage in every user in his home directory
ChrootEveryone              no

# If the previous option is set to "no", members of the following group
# won't be caged. Others will be. If you don't want chroot()ing anyone,
# just comment out ChrootEveryone and TrustedGID.
TrustedGID                  500

# Disallow anonymous connections. Only allow authenticated users.
NoAnonymous                 yes
```

The equivalent command on the command line would be **pure-ftpd -a 500 -E**.

Configure FTP with Virtual Users not Included in /etc/passwd

PureFTPd provides a way of administering FTP users in its own file, similar in structure to the file /etc/passwd.

The advantages are that PureFTP users are separated from system users and can only access the system by FTP. A normal login is not possible if there are no matching entries in the file /etc/passwd.

To administer PureFTPd users in a separate user database, you need to create a system user with whose UID the FTP users appear in the system:

useradd -m ftpusers

Once this is done, you can then create the FTP users with **pure-pw** (in the file /etc/pure-ftpd/pureftpd.passwd) by entering the following (using user **joe** as an example):

pure-pw useradd joe -u ftpusers -d /home/ftpusers/joe

You are requested to enter a password (twice) for the user.

With the help of command line options, you can specify user options such as quotas for the number of files (**-n 100)**, size limits in MB (**-N 10**), or the times when users can log in (**-z 0900-1800**).

PureFTPd does not use the ASCII file /etc/pure-ftpd/pureftpd.passwd directly, but the binary file /etc/pure-ftpd/pureftpd.pdb. This file must be regenerated every time changes are made by entering **pure-pw mkdb**.

To access the special user database, you need to start PureFTPd with **-j** to ensure that the home directory is created as soon as the user logs in and, after the -l option, the password database file.

The following is an example:

pure-ftpd -j -l puredb:/etc/pureftpd.pdb

You can modify FTP users by entering **pure-pw usermod** and delete users by entering **pure-pw userdel**.

For additional details on using the pure-pw syntax, enter **man 8 pure-pw** or **pure-pw --help**.

Manage PureFTPd Logs

PureFTPd sends its messages to the syslog daemon, so these messages appear in the usual log files.

It is also possible for PureFTPd to write its own log files in various formats. The option for this is **-O *format*:*logfile***, where ***format*** can be **clf** (Common Log Format, a format similar to that used by the Apache web server), **stats** (special output format, designed for log file analysis software), or **w3c** (special output format parsed by most commercial log analysers).

Suitable entries already exist in the configuration file /etc/pure-ftpd/pure-ftpd.conf. However, you might need to remove the comment symbol (#) to activate the entry.

The following is an example entry:

```
AltLog              clf:/var/log/pureftpd.log
```

Exercise 4-2 Configure Anonymous PureFTPd Access

In this exercise, you configure anonymous FTP access with the permission to upload files.

You can find this exercise in the workbook.

(End of Exercise)

Objective 3 Configure Time on SUSE Linux Enterprise Server 10

In order to implement a uniform time on all computers in a network, all computers must be able to access at least one time server so clocks will synchronize.

There are two ways of synchronizing the time on a SUSE Linux Enterprise Server: netdate and NTP. To configure and synchronize the time, you need to understand the following:

- Overview
- Synchronize Time with netdate and hwclock
- The Network Time Protocol (NTP)
- Synchronize Time with NTP

Overview

In order to configure and synchronize time on a SUSE Linux Enterprise Server, you need to understand the following fundamental concepts:

- Hardware Clock and System Clock
- GMT (UTC) and Local Time
- Time Configuration Files

Hardware Clock and System Clock

There are two main clocks in a Linux system:

- **Hardware clock.** This is a clock that runs independently of any control program running in the CPU. It even runs when you turn off the server.

  This clock is part of the ISA (Industry Standard Architecture) standard and is commonly called the *hardware clock*. It is also called the time clock, the RTC (Real Time Clock), the BIOS clock, or the CMOS (Complementary Metal-oxide Semiconductor) clock.

  The term hardware clock is used on Linux systems to indicate the time set by the hwclock utility.

- **System time.** This is the time kept by a clock inside the Linux kernel and is driven by a *timer interrupt* (another ISA standard).

  System time is meaningful while Linux is running on the server. System time is the number of seconds since 00:00:00 January 1, 1970 UTC (or the number of seconds since 1969).

On a Linux server, it’s the system time that is important. The hardware clock's basic purpose is to keep time when Linux is not running.

The system time is synchronized to the hardware clock when Linux first starts. After that, Linux only uses the system time.

Once the system time is set on the Linux server, it’s important that you do not use commands such as **date** to adjust the system time without considering the impact on applications and network connections.

For a Linux server connected to the Internet (or equipped with a precision oscillator or radio clock), the best way to regulate the system clock is with **ntpd**.

For a standalone or intermittently connected machine, you can use **adjtimex** instead to at least correct systematic drift (man adjtimex lists the options).

You can set the hardware clock (with a command such as **hwclock**) while the system is running. The next time you start Linux, it will synchronize with the adjusted time from the hardware clock.

The Linux kernel maintains a concept of a local time zone for the system.

Some programs and parts of the Linux kernel (such as file systems) use the kernel time zone value. An example is the vfat file system. If the kernel timezone value is wrong, the vfat file system reports and set the wrong time stamps on files.

However, programs that care about the time zone (perhaps because they want to display a local time for you) almost always use a more traditional method of determining the timezone such as using the file /etc/localtime and the files in the directory /usr/share/zoneinfo/.

GMT (UTC) and Local Time

On startup, Linux reads the time from the computer's local hardware (CMOS clock) and takes control of the time. The hardware clock can be set using one of the following:

- **UTC (Universal Time Coordinated).** This time is also referred to as GMT (Greenwich Mean Time). For this setting, the variable HWCLOCK in the file /etc/sysconfig/clock has the value **-u**.
- **Local time.** If the hardware clock is set to the local time, the variable HWCLOCK in the file /etc/sysconfig/clock has the value **--localtime**.

Choosing GMT as the hardware time makes it easier to coordinate a large number of computers in different places (especially if the computers are located in different time zones.)

Time Configuration Files

The current time (system time) is calculated with the help of the variable TIMEZONE in the file /etc/sysconfig/clock, which also handles the required changes between daylight saving time and standard time.

The following is an example of the settings in /etc/sysconfig/clock:

```
HWCLOCK="--localtime"

TIMEZONE="Europe/Berlin"

DEFAULT_TIMEZONE="Europe/Berlin"
```

By means of the variable TIMEZONE, the time configured on the local host (= system time) is set in the file /etc/localtime, a copy of the respective timezone file from /usr/share/zoneinfo/. The directory /usr/share/zoneinfo/ is a database of all time zones.

In SLES 9, there used to be a symbolic link /usr/lib/zoneinfo/localtime pointing to /etc/localtime. This link does not exist anymore in SLES 10, even if it might still be mentioned in /etc/sysconfig/clock.

Synchronize Time with netdate and hwclock

To synchronize time between network servers with netdate and hwclock, you need to know the following:

- Use hwclock
- Use netdate

Use hwclock

hwclock is a tool for accessing the hardware clock. You can display the current time, set the hardware clock to a specified time, set the hardware clock to the system time, and set the system time from the hardware clock.

You can also run hwclock periodically to insert or remove time from the hardware clock to compensate for systematic drift (where the clock consistently gains or loses time at a certain rate if left to run).

hwclock uses several methods to get and set hardware clock values. The normal way is to initialize an I/O process to the device special file /dev/rtc (RTC: Real Time Clock), which is maintained by the rtc device driver.

However, this method is not always available. The rtc driver is a relatively recent addition to Linux and is not available on older systems.

On older systems, the method of accessing the hardware clock depends on the system hardware.

For additional details on how the system accesses the hardware clock and other hwclock options, see **man hwclock**.

Some of the more commonly used options with hwclock include the following:

Table 4-3

Option	Description
-r or **--show**	This options displays the current time of the hardware clock. The time is always shown in local time, even if you keep your hardware clock set to UTC time.
-w or **--systohc**	This option sets the hardware clock to the current system time.
-s or **--hctosys**	This option sets the system time to the current hardware clock time. It also sets the kernel's timezone value to the local time zone as indicated by the TZ variable.
-a or **--adjust**	This option adds or subtracts time from the hardware clock to account for system drift (enter **man hwclock** for details).
-v or **--version**	This option displays the version of hwclock.
--set **--date=*newdate***	This option sets the hardware clock to the date given by the --date option. For example: **hwclock --set --date="9/22/06 16:45:05"**

You can also view the hardware clock time by entering **cat /proc/driver/rtc**.

Use netdate

To setup the system time once only, you can use the command netdate as follows:

netdate ***timeserver1 timeserver2. . .***

where ***timeserver*** represents a time server on the network or on the Internet that offers the time service on UDP port 37.

After querying the time servers, the netdate client compares their times with its own time.

Time differences are then sorted into groups to determine which is the largest group of servers with an identical time (within certain limits). The first computer in the group is then used to update the time on the local server.

To synchronize the time to a specific external time source, you enter **netdate** ***time_source***, as in the following:

netdate ptbtime1.ptb.de

In this case, the client queries the time server at the Physikalisch-Technische Bundesanstalt (PTB) in Braunschweig, Germany.

You then need to set the hardware clock to the system clock time by entering **hwclock --systohc** or **hwclock -w**.

The simplest way to implement time synchronization with netdate and hwclock is to use a script that is run regularly by cron.

The Network Time Protocol (NTP)

The disadvantage of using netdate is that it causes jumps of the system time into the past or the future compared to the current system time. NTP provides a means to avoid such jumps by slightly speeding or slowing system time, thus (within limits) keeping the time continuum of the system time while adjusting it.

As the networking environment continues to expand to include mixed operating system environments, time synchronization is becoming more dependent on NTP.

To configure NTP on SUSE Linux Enterprise Server, you need to understand the following:

- The Network Time Protocol
- Stratum
- NTP Daemon (ntpd)
- NTP Terms
- How the NTP Daemon Works

For more information on NTP, visit www.ntp.org.

The Network Time Protocol

NTP is an industry standard protocol that uses UDP on port 123 to communicate between time servers and time clients.

An NTP server uses the NTP protocol to provide time information to other servers or to workstations on the network.

An NTP client is a computer that understands the Network Time Protocol and gets time information from an NTP server. A time client can also, in turn, act as a time server for other servers and client workstations on the network.

Any computers on your network with Internet access can get time from NTP servers on the Internet. NTP synchronizes clocks to the UTC standard, the international time standard.

NTP not only corrects the time, but it keeps track of consistent time variations and automatically adjusts for system time drift on the client. It allows for less network traffic and it keeps the client clocks more stable, even when the network is down.

Stratum

NTP introduces the concept of a stratum. Stratum ***x*** is used as a designation of the location of the servers in NTP tree hierarchy.

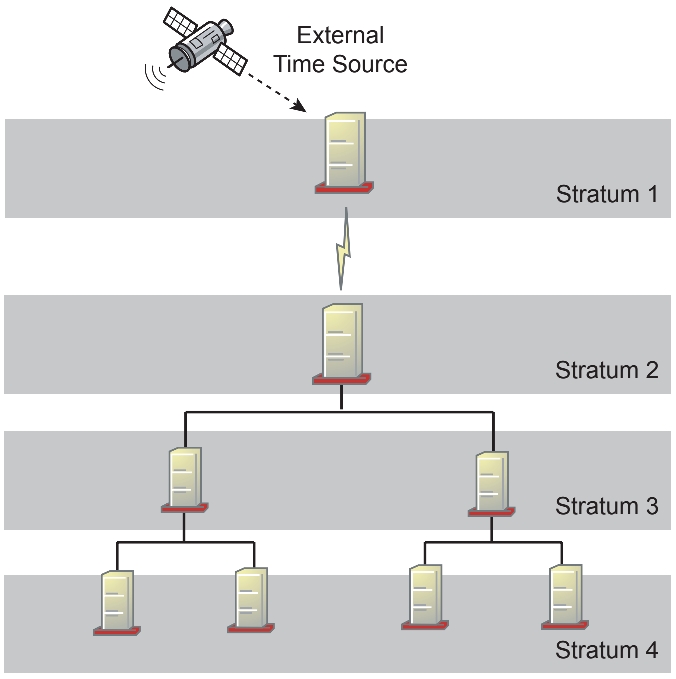
Stratum 1 is the first (highest) level in the hierarchy. It denotes servers that adjust their time by means of some external reference time source (such as a GPS (Global Positioning System), an atomic clock, or radio).

Servers that synchronize their time to stratum 1 servers are denoted as stratum 2, and those that use stratum 2 servers to synchronize their time are denoted as stratum 3, and so on until you reach a stratum level of 16 (the maximum allowed).

Differences between stratum 2 and stratum 1 servers are normally very small and, for the majority of users, unnoticeable.

The following figure depicts the stratum hierarchy.

Figure 4-4



Generally only one server in a network communicates with an external time provider. This reduces network traffic across geographical locations and minimizes traffic across routers and WANs.

NTP Daemon (ntpd)

The NTP distribution in the package xntp includes **ntpd**, the NTP daemon. This daemon is used by both the time server and the time client to give and to obtain time, respectively.

The ntpd process is designed to adjust time continuously, making the time adjustments very small.

ntpd can also limit the drift of the system clock based on historical data, even when an external time server is unavailable.

The ntpd process requires little resource overhead. This allows NTP to be easily deployed on servers hosting other services, even if the servers are heavily loaded.

ntpd uses the following approaches to avoid sudden time changes:

- ntpd regularly corrects the local computer clock on the basis of collected correction data.
- ntpd continuously corrects the local time with the help of time servers in the network.
- ntpd enables the management of local reference clocks, such as radio-controlled clocks.

NTP Terms

To configure and adjust NTP, you need to understand the following terms:

- **Drift.** During operation, ntpd measures and corrects incidental clock frequency errors and writes the current value to a file under /var/lib/ntp/drift/.

  If you start and stop ntpd, the daemon initializes the frequency from this file. This helps prevent a potentially long interval to relearn the frequency error.
- **Jitter.** This is the estimated time error of the peer clock (the delta between the client and server since the last poll.)

How the NTP Daemon Works

After starting the NTP Daemon, it automatically synchronizes the system time with a time server on an ongoing basis. The correction takes place in small increments by expanding or compressing the system time (not abruptly as when netdate and hwclock are used).

Transactions between the client and the server occur about once per minute, increasing gradually to once per 17 minutes under normal conditions. Poorly synchronized clients will tend to poll more often than well-synchronized clients.

The client uses the information it gets from the server or servers to calibrate its clock. This consists of the client determining how far its clock is off and adjusting its time to match that of the server.

To allow clocks to quickly achieve high accuracy yet avoid overshooting the time with large time adjustments, NTP uses a system where large adjustments occur quickly and small adjustments occur over time.

For small time differences (less than 128 milliseconds), NTP uses a gradual adjustment. This is called *slewing*. For larger time differences, the adjustment is immediate. This is called *stepping*.

If the accuracy of a clock becomes too insufficient (off by more than about 17 minutes), NTP aborts the NTP daemon, with the assumption that something has gone wrong with either the client or the server.

Because NTP averages the results of several time exchanges in order to reduce the effects of variable latency, it might take several minutes for NTP to even reach consensus on what the average latency is.

It often takes several adjustments (and several minutes) for NTP to reach synchronization.

In the long run, NTP tries to decrease the amount of polling it does by making the clock on each system become more accurate.

Because of the algorithm that the NTP daemon uses, it is best to synchronize with multiple servers to help protect the client from an incorrect or downed server. In many environments, it is unlikely that an NTP server failure will be noticed quickly.

Synchronize Time with NTP

To synchronize network time with NTP, you need to know how to do the following:

- Start ntpd From the Command Line
- Adjust the Time With ntpdate
- Configure the NTP Server (/etc/ntp.conf)
- Trace the Time Source with ntptrace
- Configure an NTP Client with YaST

Start ntpd From the Command Line

In SUSE Linux Enterprise Server the time server daemon ntpd is contained in the package xntp, which is installed by default. Its start script is /etc/init.d/ntp, and its central configuration file is /etc/ntp.conf.

You can start the NTP daemon by entering **rcntp start**. You can check the status of ntpd by entering **rcntp status**. To stop the NTP Daemon, use **rcntp stop**.

The start script uses variables defined in **/etc/sysconfig/ntp**.

To start NTP automatically when the system is booted, you need to set the symbolic links in the respective runlevel directories by entering **insserv ntp**.

If any changes are made to the ntp.conf file, you need to restart ntpd in order to update the configuration using the command **rcntp restart**.

After the /etc/ntp.conf file has been read by ntpd, the client sends a request to the server and the server sends back a time stamped response, along with information such as its accuracy and stratum.

Once the NTP daemon is started, other computers can use it as time server.

Adjust the Time With ntpdate

If the difference between the system time and the time server is too big, the synchronization of the system time in small increments might take too long.

To avoid this, the startscript /etc/init.d/ntp sets the system time using the program ntpdate before ntpd is started. In case you want to use ntpdate manually, the syntax is:

ntpdate ***timeserver***

You need to stop the NTP daemon before updating the system time with ntpdate.

Configure the NTP Server (/etc/ntp.conf)

As soon as you start ntpd on a host, it serves as an NTP server and can be queried via NTP. You configure the NTP server by editing the configuration file /etc/ntp.conf.

For example, you can configure one network host as a server that synchronizes the time with a time server on the Internet. The other network hosts can access this time server on the Intranet.

First you need to make sure that the following entries exist in the file /etc/ntp.conf for the local clock, which is used if the time server is not available:

```
server 127.127.1.0         # local clock (LCL)
fudge 127.127.1.0 stratum 10  # LCL is unsynchronized
```

The value after stratum indicates the distance of a server from the atomic clock, showing how reliable the time server is.

The value **1** means that the server is connected directly to the atomic clock and is very reliable. A server with value **2** gets the time from a server with the stratum value 1 and so on. A value of **16** indicates that the server is very unreliable.

The next entry in /etc/ntp.conf is for the time servers that are to be asked for the current time:

```
## Outside source of synchronized time
server ptbtime1.ptb.de
server ptbtime2.ptb.de
```

There are 2 possible methods of synchronization between the time server and the client:

- **Polling.** With polling, the client asks the server for the current time.

  Polling starts at 1 minute intervals. If the time interval is determined to be trustworthy, the interval is reset to once every 1024 seconds.

You can set the minimum and maximum limits of the polling in /etc/ntp.conf, as in the following:

```
server ptbtime1.ptb.de minpoll 4 maxpoll 12
```

The minpoll and maxpoll values are interpreted as powers of 2 (in seconds). The default settings are 6 (2^6 = 64 seconds) and 10 (2^{10} = 1024 seconds), respectively. Values between 4 and 17 are permitted.

- **Broadcasting.** By means of broadcasting, the server sends the current time to all clients, and the clients receive the signal through the option **broadcastclient** in their ntpd.conf.

  In large networks, traffic caused by polling of the time can be significant. In this case, you might want to configure the time server to distribute time information by sending broadcast packets.

  To do this, you need to enter the following in /etc/ntp.conf on the server (where the IP address is the broadcast address used in the network):

```
broadcast 10.0.0.255
```

  On the client:

```
disable auth
broadcastclient
```

  For reasons of security, broadcast-based synchronization should be used together with an authentication key so that the client only accepts information from trustworthy time servers. See the documentation under /usr/share/doc/packages/xntp-doc/ (package xntp-doc).

You also need to include the name for the drift file and log file in /etc/ntp.conf, as in the following:

```
driftfile /var/lib/ntp/drift/ntp.drift
logfile /var/log/ntp
```

The drift file contains information that describes how the hardware clock drifts. When the daemon ntpd is started for the first time, this file does not exist. It takes about 15 minutes for the daemon to gather enough information to create the file.

To have ntpd reread the configuration, enter **rcxntp restart**.

For time requests of other kinds (such as time servers for netdate) to be processed, the services must be made available by means of inetd or xinetd.

For this reason, the prepared entries for daytime and time must be enabled for UDP and TCP in the configuration file of inetd or xinetd.

Configure an NTP Client with YaST

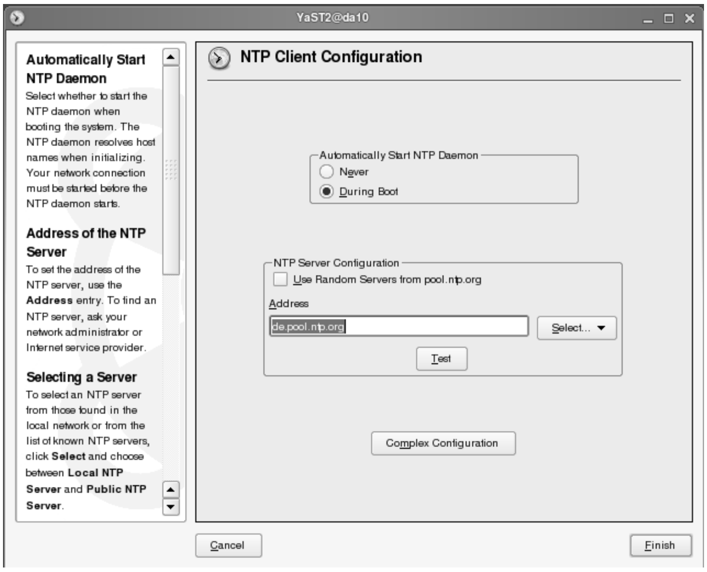
YaST provides an NTP Client module for configuring an NTP client on your SUSE Linux Enterprise Server so it can synchronize with an existing NTP server.

To configure the NTP client with YaST, start YaST and select **Network Services > NTP Client**. From a terminal you can start the module directly as root by entering **yast2 ntp-client**.

The NTP Client configuration dialog appears. Configure the NTP client to start each time you boot your system by selecting **During Boot**.

A message appears that ntpd should have a permanent internet connection to function properly. After selecting continue, you can enter a time server in the dialog:

Figure 4-5



YaST2@da10

Automatically Start NTP Daemon
Select whether to start the NTP daemon when booting the system. The NTP daemon resolves host names when initializing. Your network connection must be started before the NTP daemon starts.

Address of the NTP Server
To set the address of the NTP server, use the **Address** entry. To find an NTP server, ask your network administrator or Internet service provider.

Selecting a Server
To select an NTP server from those found in the local network or from the list of known NTP servers, click **Select** and choose between **Local NTP Server** and **Public NTP Server**.

NTP Client Configuration

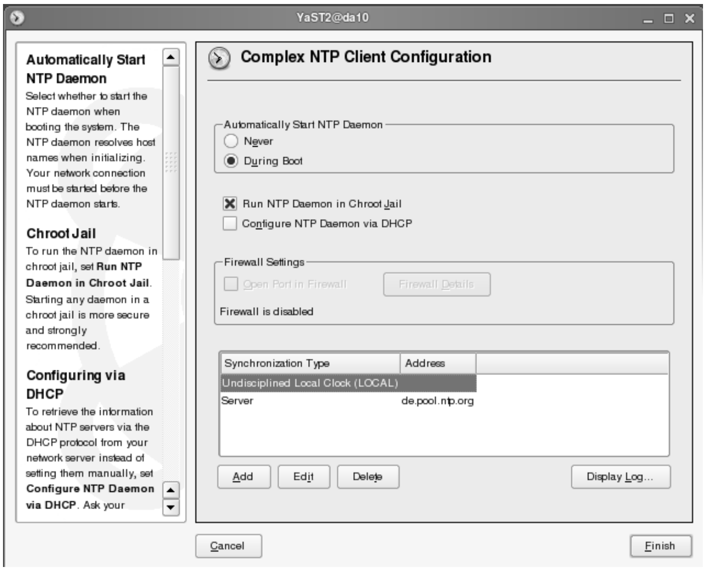
Automatically Start NTP Daemon
( ) Never
(●) During Boot

NTP Server Configuration
[ ] Use Random Servers from pool.ntp.org
Address: de.pool.ntp.org

You can enter it manually, or choose from a list after choosing **Public NTP Server** from the **Select** drop-down menu. You can also simply select **Use Random Servers from pool.ntp.org**.

Complex Configuration opens another dialog with additional configuration options:

Figure 4-6



Automatically Start NTP Daemon
Select whether to start the NTP daemon when booting the system. The NTP daemon resolves host names when initializing. Your network connection must be started before the NTP daemon starts.

Chroot Jail
To run the NTP daemon in chroot jail, set **Run NTP Daemon in Chroot Jail**. Starting any daemon in a chroot jail is more secure and strongly recommended.

Configuring via DHCP
To retrieve the information about NTP servers via the DHCP protocol from your network server instead of setting them manually, set **Configure NTP Daemon via DHCP**. Ask your

Complex NTP Client Configuration

Automatically Start NTP Daemon
- ( ) Never
- (●) During Boot

- [x] Run NTP Daemon in Chroot Jail
- [ ] Configure NTP Daemon via DHCP

Firewall Settings
- [ ] Open Port in Firewall

Firewall is disabled

Synchronization Type	Address
Undisciplined Local Clock (LOCAL)	
Server	de.pool.ntp.org

You can choose to run ntpd in a change root environment and allow a change of the configuration via DHCP. Providing the firewall is active, you can open the port necessary for NTP.

In the lower section of the dialog you can add, edit, or delete entries to /etc/ntpd.conf, like server and broadcast as covered in “Configure the NTP Server (/etc/ntp.conf)” on 4-42.

Trace the Time Source with ntptrace

The NTP distribution also includes the ntptrace program. ntptrace is an informational tool that traces the source of time that a time consumer is receiving. It can be a useful debugging tool.

The following is an example of ntptrace output:

```
da10:~ # ntptrace
localhost: stratum 3, offset 0.000723, synch distance 1.18225
tick.east.ca: stratum 2, offset 1.601143, synch distance 0.06713
tock.usask.ca: stratum 1, offset 1.712003, synch distance 0.00723, refid
'TRUE'
```

The ntptrace output lists the client name, its stratum, its time offset from the local host, the synchronization distance, and the ID of the reference clock attached to a server, if one exists.

The synchronization distance is a measure of clock accuracy, assuming that it has a correct time source.

Query the NTP Daemon Status

To verify that the time server is working properly, you can enter ntpq -p. The command queries the status of the xntpd daemon, and returns information similar to the following:

```
 remote      refid    st t when poll reach   delay   offset  jitter
=========================================================================
 LOCAL(0)    LOCAL(0) 10 l   15   64    1    0.000    0.000   0.008
*ptb1.ptb.de .PTB.     1 u   14   64    1   27.165    2.348   0.001
 ntp2.ptb.de .PTB.     1 u   13   64    1   26.159    0.726   0.001
```

Displayed information includes the following:

- **remote.** Hostname or IP address of the time server.
- **refid.** Type of reference source (0.0.0.0 = unknown).
- **st.** Stratum value for the server.
- **when.** Number of seconds since the last poll.
- **poll.** Number of seconds between two polls.
- **reach.** Indicates if the time server was reached in the last poll attempt. reach begins with the value 0 when you start ntpd.

For every successful attempt, a 1 is added to the binary register on the right. The maximum value of 377 means that the server was reachable in the last 8 requests.

- **delay.** Time between the ntpd request and the arrival of the answer (in milliseconds).
- **offset.** Difference between the reference time and the system time (in milliseconds).
- **jitter.** Size of the discrepancies between individual time comparisons (in milliseconds).

An asterisk ("*") in front of a server name means that this server is the current reference server with which system time is compared. If this server cannot be reached then the server that is marked with a preceding plus sign ("+") is used.

Exercise 4-3 Configure ntpd

In this exercise, you configure your server to get time information from server da1.

You can find this exercise in the workbook.

(End of Exercise)

Objective 4 Configure NFS (Network File System)

Network File System (NFS) lets you configure an NFS file server that gives users transparent access to programs, files, or storage space on the server.

To configure NFS for your network, you need to know the following:

- Network File System Basics
- How NFS Works
- NFS Configuration Overview
- Configure an NFS Server Manually
- Configure an NFS Server with YaST
- Export a Directory Temporarily
- Configure NFS Client Access with YaST
- Configure NFS Client Access from the Command Line
- Monitor the NFS System

Network File System Basics

NFS is designed for sharing files and directories over a network, and requires configuration of an NFS server (where the files and directories are located) and NFS clients (computers that access the files and directories remotely).

File systems are exported by an NFS server, and appear and behave on a NFS client as if they were located on a local machine.

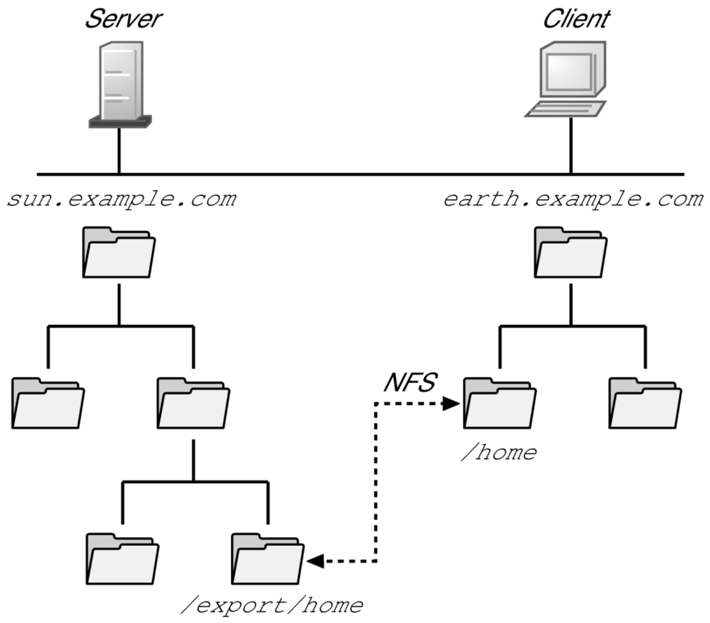
For example, with NFS each user's home directory can be exported by an NFS server and imported to a client, so the same home directories are accessible from every workstation on the network.

Directories like /home/, /opt/, and /usr/ are good candidates for export via NFS. However, others, including /bin/, /boot/, /dev/, /etc/, /lib/, /root/, /sbin/, /tmp/, and /var/, should be available on the local disk only.

Using NFS for home directories only makes sense with a central user management (for instance OpenLDAP).

The following is an example of mounting the directory /home/ (exported by the NFS Server sun) on the computer earth:

Figure 4-7



A computer can be both a NFS server and an NFS client. It can supply file systems over the network (export) and mount file systems from other hosts (import).

The NFS daemon is part of the kernel and only needs to be configured, then activated. The start script is /etc/init.d/nfsserver. The kernel NFS daemon includes file locking, which means that only 1 user at a time has write access to files.

How NFS Works

NFS is a RPC (Remote Procedure Call) service. An essential component for RPC services is the *portmapper* that manages these services and needs to be started first. The portmap daemon is activated by default on SUSE Linux Enterprise Server 10.

When an RPC service starts up, it binds to a port in the system (as any other network service), but also communicates this port and the service it offers (such as NFS) to the portmapper.

Because every RPC program must be registered by the portmapper when it is started, RPC programs must be restarted each time you restart the portmapper.

The following lists the services required on an NFS server:

Table 4-4

Service	Program (daemon)	Start Script
Port mapper	/sbin/portmap	/etc/init.d/portmap
NFS server	/usr/sbin/rpc.nfsd /usr/sbin/rpc.mountd	/etc/init.d/nfsserver

The /sbin/rpc.lockd program starts the NFS lock manager (NLM) on kernels that don't start it automatically. In SUSE Linux Enterprise Server 10 the kernel does this.

You can use the command /etc/init.d/nfsserver to start the NFS server. The script nfsserver passes the list of exported directories to the kernel, and then starts or stops the daemon rpc.mountd and, using rpc.nfsd, the nfsd kernel threads.

The mount daemon (/usr/sbin/rpc.mountd) accepts each mount request and compares it with the entries in the configuration file /etc/exports. If access is allowed, the data is delivered to the client.

Because rpc.nfsd can start several kernel threads, the start script interprets the variable USE_KERNEL_NFSD_NUMBER in the file /etc/sysconfig/nfs. This variable determines the number of threads to start. By default, 4 server threads are started.

NFS Configuration Overview

All configuration settings for the NFS server are stored in the file /etc/exports Client-side configuration takes place using the file /etc/fstab. Both will be covered in detail later.

Both the NFS server and the clients can be configured with YaST modules. You can also modify the configuration files directly.

For the NFS server to start automatically when the computer is booted, the corresponding symbolic links in the runlevel directories must be generated. If you configure the NFS server with YaST, this is done automatically; otherwise, you need to generate them with **insserv nfsserver**.

Configure an NFS Server Manually

You can configure the server from the command line by doing the following:

- **Check for service (daemon) availability.** Make sure the following are available on your NFS server:
  - RPC portmapper (portmap)
  - RPC mount daemon (rpc.mountd)
  - RPC NFS daemon (rpc.nfsd)
- **Configure the services to be available at bootup.** For these services to be started by the scripts /etc/init.d/portmap and /etc/init.d/nfsserver when the system is booted, enter the following commands:

  insserv /etc/init.d/portmap (activated by default)
  insserv /etc/init.d/nfsserver
- **Define exported directories in /etc/exports**. For each directory to export, one line is needed to set which computers can access that directory with what permissions. All subdirectories of this directory are automatically exported as well.

  The following is the general syntax of the file /etc/exports:

directory **[***host***[(***option1***,***option2***,***option3***,...)]] ...**

Do not put any spaces between the host name, the parentheses enclosing the options, and the option strings themselves.

A *host* can be one of the following:

- A standalone computer with its name in short form (it must be possible to resolve this with name resolution), with its Fully Qualified Domain Name (FQDN), or with its IP address.
- A network, specified by an address with a netmask or by the domain name with a prefixed placeholder (such as ***.digitalairlines.com**).

Authorized computers are usually specified with their full names (including domain name), but you can use wildcards like * or ?.

If you do not specify a host, or use *****, any computer can import the file system with the given permissions.

- **Set permissions for exported directories in /etc/exports.** You need to set permission options for the file system to export in parenthesis after the computer name. The most commonly-used options include the following:

Table 4-5

Option	Meaning
ro	File system is exported with read-only permission (default).
rw	File system is exported with read-write permission. The local file permissions are not overridden.
root_squash	(Default) This ensures that the user root of the given machine does not have root permissions on this file system. This is achieved by assigning user ID 65534 to users with user ID 0 (root). This user ID should be set to nobody (which is the default).

Table 4-5 *(continued)*

Option	Meaning
no_root_squash	Does not assign user ID 65534 to user ID 0, keeping the root permissions valid.

The following is an example of an edited /etc/exports file that includes permissions:

```
#
# /etc/exports
#
/home          da10(rw,sync) da20(rw,sync)
/usr/X11       da10(ro,sync) da20(ro,sync)
/usr/lib/texmf da10(ro,sync) da20(rw,sync)
/srv/ftp       *(ro,sync)
```

Whenever you want to specify different permissions for a subdirectory (such as /home/geeko/pictures/) from an already exported directory (such as /home/geeko/), the additional directory needs its own separate entry in /etc/exports.

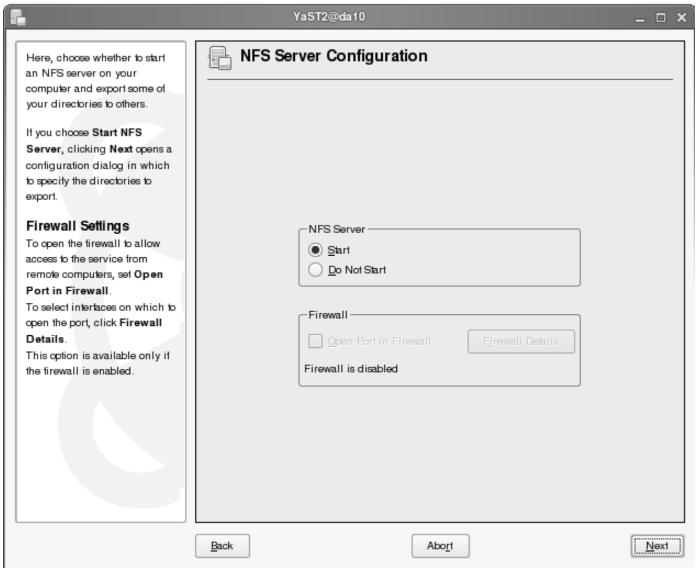
- **Restart mountd and nfsd.** The /etc/exports is read by mountd and nfsd. If you change anything in this file, you need to restart mountd and nfsd for your changes to take effect. You can do this by entering **rcnfsserver restart**.

Configure an NFS Server with YaST

To use YaST to configure the NFS server, start YaST and then select **Network Services > NFS Server**. You can also start the NFS Server module directly by entering in a terminal window as root **yast2 nfs-server**.

The following appears:

Figure 4-8



Here, choose whether to start an NFS server on your computer and export some of your directories to others.

If you choose **Start NFS Server**, clicking **Next** opens a configuration dialog in which to specify the directories to export.

Firewall Settings

To open the firewall to allow access to the service from remote computers, set **Open Port in Firewall**.
To select interfaces on which to open the port, click **Firewall Details**.
This option is available only if the firewall is enabled.

NFS Server Configuration

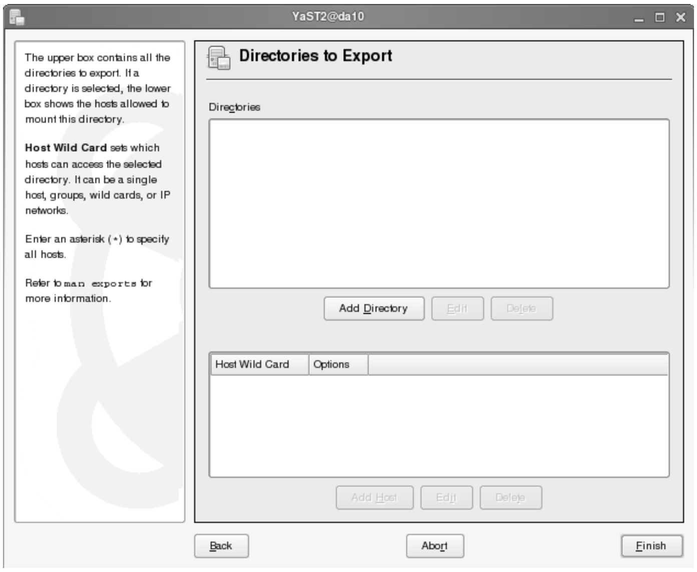
NFS Server
[✓ Start]
Do Not Start

Firewall
Open Port in Firewall
Firewall is disabled

Select **Start** in the upper part of the dialog. The lower part is only active if the firewall is activated. In this case you can open the ports necessary for NFS by selecting **Open Port in Firewall**.

Continue by selecting **Next**. A **Directories to Export** dialog appears:

Figure 4-9



The upper box contains all the directories to export. If a directory is selected, the lower box shows the hosts allowed to mount this directory.

Host Wild Card sets which hosts can access the selected directory. It can be a single host, groups, wild cards, or IP networks.

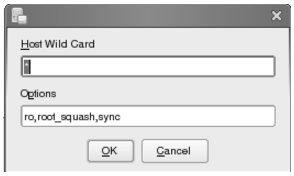
Enter an asterisk (*) to specify all hosts.

Refer to `man exports` for more information.

Add a directory for export by selecting **Add Directory**; then enter or browse to and select a ***directory***.

The following dialog appears:

Figure 4-10



Host Wild Card: *

Options: ro,root_squash,sync

Under **Host Wild Card,** this dialog lets you configure the host or hosts that should have access to the directory. You can define a single host, netgroups, wildcards, and IP networks.

For details on configuring the host settings, see “Configure an NFS Server Manually” on 4-53.

Add further hosts, edit, or delete a host entry for a directory by selecting the directory; then select **Add host**, **Edit**, or **Delete**.

When you finish, save the configuration by selecting **Finish**.

Export a Directory Temporarily

You can export a directory temporarily (without editing the file /etc/exports) by using the command exportfs.

For example, to export the directory /software to all hosts in the network 192.168.0.0/24, you would enter the following command:

exportfs -o ro,root_squash,sync 192.168.0.0/24:/software

To restore the original state, all you need to do is enter the command **exportfs -r**. The file /etc/exports is reloaded and the directory software is no longer exported.

The directories that are currently exported are listed in the file /var/lib/nfs/etab. The content of this file is updated when you use the command exportfs.

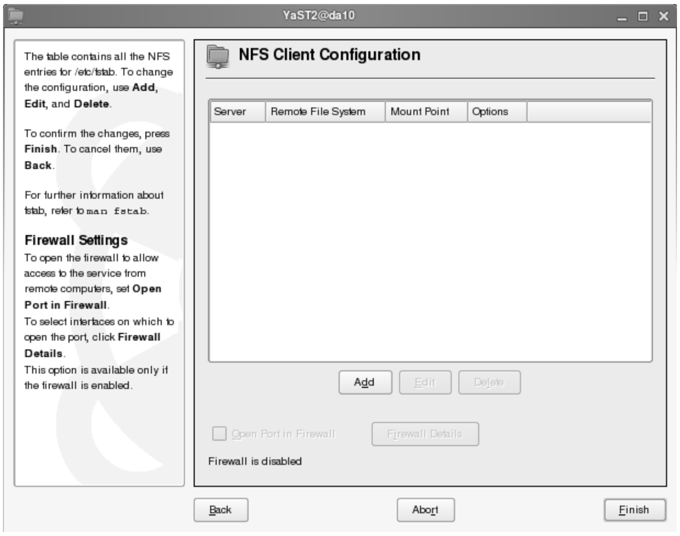
Configure NFS Client Access with YaST

NFS directories exported on a server can be mounted into the file system tree of a client. The easiest way to do this is to use the YaST NFS Client module.

To use YaST to configure the NFS client, start the YaST Control Center and then select **Network Services > NFS Client**. You can also start the NFS Client module directly by entering in a terminal window as root **yast2 nfs**.

The **NFS Client Configuration** dialog appears:

Figure 4-11



The table contains all the NFS entries for /etc/fstab. To change the configuration, use **Add**, **Edit**, and **Delete**.

To confirm the changes, press **Finish**. To cancel them, use **Back**.

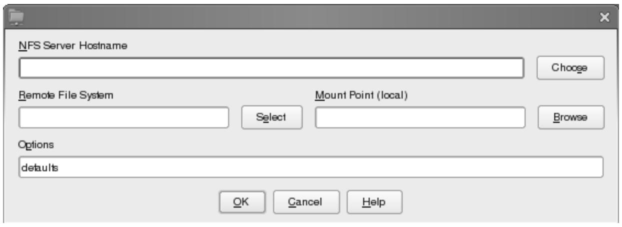
For further information about fstab, refer to `man fstab`.

Firewall Settings
To open the firewall to allow access to the service from remote computers, set **Open Port in Firewall**.
To select interfaces on which to open the port, click **Firewall Details**.
This option is available only if the firewall is enabled.

Firewall is disabled

Add a directory to the list by selecting **Add**. The following appears:

Figure 4-12



Options: defaults

From this dialog, you can configure the directory to mount in your file system tree. Configure the directory by doing the following:

1. Enter the NFS server's hostname, or find and select the NFS server from a list of NFS servers on your network by selecting **Choose**.

2. In the **Remote File System** field, enter the ***exported directory*** on the NFS server you want to mount, or find and select the available directory by selecting **Select**.

3. In the **Mount Point (local)** field, enter the ***mountpoint*** in your local file tree to mount the exported directory, or browse to and select the mountpoint by selecting **Browse**.
4. In the Options field, enter any ***options*** you would normally use with the mount command. For a list of these options, enter **man mount**.
5. When you finish configuring the directory, select **OK**. You are returned to the NFS client configuration dialog.

Save the NFS client settings by selecting **Finish**. The settings are saved, services restarted, and the exported directories are mounted in your local file tree.

Configure NFS Client Access from the Command Line

To configure and mount NFS directories, you need to know how to do the following:

- Import Directories Manually from an NFS Server
- Mount NFS Directories Automatically

Import Directories Manually from an NFS Server

You can import a directory manually from an NFS server by using the command mount. The only prerequisite is a running RPC port mapper, which you can start by entering (as root) the command **rcportmap start**.

The command **mount** automatically tries to recognize the file system (such as ext2, ext3, or ReiserFS). However, you can also use the mount option -t to indicated the file system type.

In the following example, the file system type NFS is specified:

mount -t nfs -o *options host:/directory /mountpoint*

Instead of a device file, the name of the NFS server together with the directory to import is used within the mount command.

The following are the most important mount options (**-o**)used with NFS:

- **soft (opposite: hard).** If the attempt to access the NFS server extends beyond the preset time frame (major timeout is 60 seconds), the mount attempt will be aborted.

  Otherwise, the client attempts to mount the exported directory until it receives feedback from the server that the attempt was successful.

  If a system tries to mount an NFS file system at boot time, this can cause the boot process to hang because the process will stop at this point when it attempts to mount the NFS directory.

  For directories that are not essential for the system to function, you can use the option soft. For directories that must be mounted (such as home directories), you can use the option hard.

- **bg (default: fg).** If you use this option, and the first attempt is unsuccessful, all further mount attempts are run in the background.

  This prevents the boot process from hanging when NFS exports are automatically mounted, with attempts to mount the directories continuing in the background.

- **rsize=*n*.** This option lets you set the number of bytes (***n***) that NFS reads from the NFS server at one time.

  Because older NFS versions have a limitation of 1024 bytes, this is the default value. Current versions of NFS (Version 3 and 4) can process larger amounts of data.

  For quicker access, we recommend resetting the rsize to **8192**.

- **wsize=*n*.** This option lets you set the number of bytes (***n***) that can be written to the NFS server.

  The default value is set to 1024. For faster write access, we recommend setting wsize to **8192**.

- **retry=*n*.** This option lets you set the number of minutes (**n**) an attempt can take to mount a directory through NFS. The default value is **10000** minutes (approximately 1 week).

- **nosuid.** This option lets you disable any interpretation of the SUID and SGID bits on the corresponding file system.

  For security reasons, always use this option for any file system that might be susceptible to tampering.

  If you do not use this option, there is a possibility that a user can obtain root access to the local file system by putting a SUID root executable on the imported file system.

- **nodev.** This option lets you disable any interpretation of device files in the imported file system. We recommend that you use this option for security reasons.

  Without setting this option, someone could create a device such as /dev/hda on the NFS export, then use it to obtain write permissions for the hard disk as soon as the file can be accessed from the client side.

- **exec (opposite: noexec).** Permit or do not allow the execution of binaries on the mounted file system.

You can use the command umount to unmount a file system. However, you can only do this if the file system is currently not being accessed.

For additional information on nfs, mount options and on the file /etc/fstab, enter **man 5 nfs**, **man 8 mount**, or **man 5 fstab**.

Mount NFS Directories Automatically

To mount directories automatically when booting (such as the home directories from a file server), you need to make corresponding entries in the file /etc/fstab.

When the system is booted, the start script /etc/init.d/nfs loads the file /etc/fstab, which indicates which file systems are mounted where and with which options.

The following is an example of an entry for an NFS mountpoint in the file /etc/fstab:

```
da1:/training/home /home nfs soft,rsize=8192,wsize=8192 0 0
```

In this entry, the first value indicates the host name of the NFS server (**da1**) and the directory it exports (**/training/home/**).

The second value indicates the mountpoint, which is the directory in the local file system where the exported directory should be attached (**/home/**).

The third value indicates the file system type (**nfs**). The comma-separated values following the file system type provide NFS-specific mounting options.

At the end of the line, there are 2 numbers (**0 0**). The first indicates whether to back up the file system with the help of dump (first number) or not. The second number configures whether the file system check is disabled (0), done on this file system only at a time (1), or parallelized when multiple disks are available on the computer (2).

In the example, the system does neither, as both options are set to 0.

After modifying an entry of a currently mounted filesystem in the file /etc/fstab, you can have the system read the changes by entering **mount -o remount /*mountpoint***. To mount all file systems that are not currently mounted and do not contain the option noauto, enter **mount -a**. (noauto is used with devices that are not automatically mounted, like floppy disks.)

You also need to activate the start script of the NFS client by entering **insserv nfs** (which sets the symbolic links in the respective runlevel directories).

Monitor the NFS System

Some tools are available to help you monitor the NFS system.

Enter **rpcinfo -p** to display information about the portmapper. The option **-p** displays all the programs registered with the portmapper, similar to the following:

```
da10:~ # rpcinfo -p
   Program Vers Proto   Port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    4   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    100024    1   udp   1026  status
    100021    1   udp   1026  nlockmgr
    100021    3   udp   1026  nlockmgr
    100021    4   udp   1026  nlockmgr
    100024    1   tcp  29301  status
    100021    1   tcp  29301  nlockmgr
    100021    3   tcp  29301  nlockmgr
    100021    4   tcp  29301  nlockmgr
    100005    1   udp    967  mountd
...
```

The NFS server daemon registers itself to the port mapper with the name nfs. The NFS mount daemon uses the name mountd.

You can use the command showmount to display information about the exported directories of an NFS server.

showmount -e da1 displays the exported directories of the machine da1. The option -a shows which computers have mounted which directories.

Exercise 4-4 Set Up and Manage Network File System (NFS)

In the first part of this exercise, you create a directory /export/sles10 and use it as mountpoint to import the directory /export/sles10 from da1 using NFS. Create an /etc/fstab entry to mount the directory automatically at boot time.

In the second part, you export the directory /export/data2 to others using NFS.

You can find this exercise in the workbook.

(End of Exercise)

Summary

Objective	Summary
1. Enable the Extended Internet Daemon (xinetd)	The Extended Internet Daemon (xinetd) is used to start various network services like ftp or POP3 when a connection is made to the respective port. The configuration file of xinetd is /etc/xinetd.conf and files for the various services in /etc/xinetd.d/. Configuration can be done with the YaST Network Services Module, an editor, and, to a certain extent, with chkconfig.
2. Enable an FTP Server	FTP is a widely used protocol to transfer files. It uses two TCP connections, one for control commands and one for the data transfer. There are various FTP servers available. PureFTPd has the advantage of a flexible configuration and the reputation of being secure. It can be configured via a configuration file or via command line options

Objective	Summary
3. Configure Time on SUSE Linux Enterprise Server 10	In order to implement a uniform time on all computers in a network, all computers must have access to at least one time server. There are 2 ways of synchronizing the time on a SUSE Linux Enterprise Server: netdate and NTP. The package xntp contains the ntpd time server to get the time from another time server as well as providing time to other machines on the network via NTP.
4. Configure NFS (Network File System)	Network File System (NFS) lets you configure an NFS file server that gives users transparent access to programs, files, or storage space on the server. Directories to export are specified in /etc/exports. NFS is an RPC-based service and thus needs the portmapper to function properly. /etc/init.d/nfssserver is the script to start the NFS server. Directories from other servers can be imported using the command mount or during boot according to entries in the file /etc/fstab.

SECTION 5 Create Shell Scripts

In this section, you learn about the basic scripting elements and structures of the shell programing language.

Objectives

1. Understand the Course Project
2. Use Basic Script Elements
3. Understand Variables and Command Substitution
4. Use Control Structures
5. Use Arithmetic Operators
6. Read User Input
7. Use Arrays
8. Finalize the Course Project
9. Use Advanced Scripting Techniques
10. Learn About Useful Commands in Shell Scripts

Introduction

The Linux shell can control the system with commands and perform file operations or start applications. You can also create a file that includes several shell commands and start this file like an application.

This type of file is called a *shell script*. The following are several reasons why you need to understand and create shell scripts:

- You can automate many daily tasks with shell scripts. In many cases this increases speed and convenience in everyday work.
- The boot procedure and many other system functions are controlled by shell scripts. To understand and manipulate the system behavior, you need a basic understanding of shell programming.
- Shell programming is relatively easy to learn compared to other programming languages.
- A shell script runs on almost every UNIX-like operating system and does not need to be adapted to other platforms.

There are also some disadvantages:

- Shell scripts are rather slow compared with other scripting languages.
- Shell scripts can use a lot of CPU power.

However, in most cases these disadvantages are not significant.

As you might have noticed, a Linux system offers different shell types. Shell scripts that are developed for one shell can usually be executed with a different shell, but this cannot be guaranteed.

This section focuses on the Bash shell, which is the default shell in SUSE Linux Enterprise Server 10.

As with all programing languages, shell scripting is learned best by actually writing code.

The exercises in this section include a description of a script that needs to be written. We recommend attempting to create the script, and then comparing your script to the solution to understand the scripting concepts covered.

You can find all these scripts on the *3073 Course CD* in the directory /exercises/section_5.

Objective 1 Understand the Course Project

Shell scripting is understood and learned best by doing it. Therefore we will work on an example project in this section. Our shell script will start with just a single line of code and evolve as we learn more scripting techniques.

The goal is to develop a request tracker, that can be used to manage user requests sent to a system administrator. The project consists of two parts:

- **Web Page.** The users can enter requests through a web interface. For each request, the web page generates a mail, which is sent to geeko@localhost.
- **Command Line Tool.** This tool opens the mail box of the user geeko and displays all requests. Requests can be viewed and deleted.

The development of the web page is already completed. You can find the file **/exercises/section_5/form_mailer.php** on the course CD. The files need to be copied into the htdocs directory of a PHP enabled web server.

In section 2 of course 3073 you learned how to setup a web server with PHP support.

The course project will be to develop the command line tool. If you don't want to setup a web server for the web page, you can also copy the mailbox file **geeko** from **/exercises/section_5** on the course CD to **/var/spool/mail** on your system.

If you would like to use the web interface, submit some example requests and check if they appear in geekos mailbox file under **/var/spool/mail/geeko** before you start with the project.

When writting shell scripts, you usually have many differrent options to solve a problem. Please note, that our project will not always use the most efficient way of coding. Otherwise the examples would be less readable and much harder to understand.

The script developed in this section is intended to process a dedicated mailbox file. Never run this script on your normal mailbox. The file might be corrupted afterwards and can not be read with a mail program anymore.

Exercise 5-1 Prepare your Environment

In this exercise, you prepare your system for the following exercises in this section.

You can find this exercise in the workbook.

(End of Exercise)

Objective 2 Use Basic Script Elements

The shell programming language is a powerful and complete programing language. Before you can start to create scripts, you need to become familiar with basic scripting techniques and elements.

Before writing your first shell script, you should consider a few points about scripting in general.

A shell script is basically an ASCII text file containing commands to be executed in sequence. To allow this, it is important that permissions for the script file are set to “r” (readable) and “x” (executable) for the user that runs it.

However, the execute permission is not granted by default to newly created files. To assign this permission, you need to use a command such as the following:

chmod +x script.sh

You can also run the script from another shell with a command such as the following:

sh script.sh

In this example, it is not necessary to make the script executable. On SUSE Linux Enterprise Server 10, /bin/sh is a link to /bin/bash. It doesn't really matter whether you call the script with sh script.sh or bash script.sh.

Another important point is that the directory where the script is located must actually be in the user´s search path for executables.

A good way to deal with this is to create a **/bin** directory for scripts under each user´s home directory. Then you can add this directory to the user's search path by adding a line such as the following to your ~.bashrc:

```
export PATH=$PATH:~/bin
```

Otherwise, shell scripts must be started with the full pathname.

When naming script files, it is a good idea to add an **.sh** extension to the filename. This ensures that the file can easily be recognized as a shell script.

If you do not add the suffix, you need to make sure the filename is not identical to existing commands. For example, a common mistake is to name a script ***test*** which interferes with the command line tool test.

Before we start with our project, let's have a look at the mailbox file we are going to process with the Request Tracker. The following is a part of the file:

```
From wwwrun@linux-rwke.site  Tue May  9 11:06:32 2006
Return-Path: <wwwrun@linux-rwke.site>
X-Original-To: geeko@localhost
Delivered-To: geeko@localhost.linux-rwke.site
Received: by linux-rwke.site (Postfix, from userid 30)
id 048002105F; Tue,  9 May 2006 11:06:31 -0400 (EDT)
To: geeko@localhost.linux-rwke.site
Subject: Ticket
Message-Id: <20060509150632.048002105F@linux-rwke.site>
Date: Tue,  9 May 2006 11:06:31 -0400 (EDT)
From: wwwrun@linux-rwke.site (WWW daemon apache)

#Ticket|Can not Print!|tux@novell.com|Workstation|The Printer in office is not
working. Can you please check...

From wwwrun@linux-rwke.site  Tue May  9 11:07:47 2006
Return-Path: <wwwrun@linux-rwke.site>
X-Original-To: geeko@localhost
Delivered-To: geeko@localhost.linux-rwke.site
Received: by linux-rwke.site (Postfix, from userid 30)
id CAD1C2105F; Tue,  9 May 2006 11:07:46 -0400 (EDT)
To: geeko@localhost.linux-rwke.site
Subject: Ticket
Message-Id: <20060509150746.CAD1C2105F@linux-rwke.site>
Date: Tue,  9 May 2006 11:07:46 -0400 (EDT)
From: wwwrun@linux-rwke.site (WWW daemon apache)

#Ticket|Web Server Broken!|websupport@novell.com|Server|Our web server is not
reachable anymore. Please fix!
```

The example shows two mail messages from the mailbox file. Each mail message starts with the line `From wwwrun@linux-rwke.site...` and ends with a line `#Ticket`. The #Ticket line is the actual content of the mail, all other lines are headers which carry meta data and transportation details of the mail.

The #Ticket line includes the information of the request sent through the web interface. It contains multiple fields separated by the | character. The fields have the following meaning (right of #Ticket):

Subject of Request | Email Address of Sender | Request Category | Description of Request

We start with a script that simply outputs the mailbox file to the terminal:

```
#!/bin/bash
#basic_script1.sh

cat /var/spool/mail/geeko
```

This first line tells the Linux kernel that this file has to be passed to the bash shell in order to be executed. Every shell script that should be directly executable needs such a line. The path to the bash binary is /bin/bash.

The second line is a comment. Every line starting with a # is not recognized by the interpreting shell. In this case the comment is used for the filename under which you can find the script on the course CD. All following script examples have such a comment.

The fourth line uses the **cat** command to open the mailbox file **/var/spool/mail/geeko** and to print it’s content to the terminal. You can also use any commands you know from the command line in a shell script.

When you execute this script, the output does not look very well-formated. We are only interested in the content of the mails, but not the headers. Look at the following improvement:

```
#!/bin/bash
#basic_script2.sh

cat /var/spool/mail/geeko | grep '#'
```

The output of cat is passed to grep using the the pipe | character. Grep prints only those lines that include a ’#’ which makes the output much clearer.

As a last improvement for the moment we are adding a start message to the script.

```
#!/bin/bash
#basic_script3.sh

echo -e "Welcome to the Request Tracker.\nThe following are open requests:\n"
cat /var/spool/mail/geeko | grep '#'
```

The echo command can be used to output text to the terminal, which is enclosed in double quotes. The option **-e** let’s echo interpret backslash sequences. In our script the sequence **\n** is used, which creates a line break (new-line) in the output.

The following is a list of other backslash sequences that can be used with echo:

- \\ Outputs a backslash
- **\a** Outputs an alert (beep tone)
- **\b** Outputs a backspace
- **\c** Outputs a suppress trailing new-line
- **\f** Outputs a form feed
- **\n** Outputs a new-line
- **\r** Outputs a carriage return
- **\t** Outputs a horizontal tab

- **\v** Outputs a vertical tab

After the modifications, the output of our script looks like the following:

```
geeko@linux-rwke:~/Shell_Scripting> ./basic_script3.sh
Welcome to the Request Tracker.
The following are open requests:

#Ticket|London Office|florian@fluvian.de|Telephone|Im trying to call the London
office, but the line is busy all day!
#Ticket|Can not Print!|tux@novell.com|Workstation|The Printer in office is not
working. Can you please check...
#Ticket|Web Server Broken!|websupport@novell.com|Server|Our web server is not
reachable anymore. Please fix!
```

Exercise 5-2 Create a Basic Shell Script

In this exercise, you create your first shell script.

You can find this exercise in the workbook.

(End of Exercise)

Objective 3 Understand Variables and Command Substitution

Variables are an important component of all programming languages. You can understand variables as containers that hold data. Instead of the data itself, the variable is used in the program code.

Look at the following example:

```
#!/bin/bash
#variable_1.sh

a="Geeko"
echo "Hello, my name is $a"
```

The string “Geeko” is assigned to the variable a. Then the variable a is used in the echo command. Two things are important here:

1. When you assign a variable, you just use the name of the variable. When you access the data of a variable, you put a $ before the variable name.
2. When you assign data to a variable, there must be no spaces between the variable name, the = character and the data.

The following is the output of the example script:

```
geeko@linux-rwke:~/Shell_Scripting> ./variables_1.sh
Hello, my name is Geeko
```

We use $a in the echo line and the variable is replaced with it’s content.

Variables can not only contain strings, but also numbers. By default a variable in a shell script can hold any kind of data. However, it is possible to limit a variable to a specific type (for example a string) with the **declare** command.

So far we have only assigned static values to variables, but it’s also possible to assign the output of a command to a variable or to use a command directly where the output is needed. This is called command substitution.

The term *command substitution* basically means that the output of a command is used in a shell command line or a shell script.

In the following example, the output of the command date is used to generate the output of the current date:

```
#!/bin/bash
#command_subs_1.sh

echo "Today is `date +%m/%d/%Y`"
```

Note that the command date +%m/%d/%Y is included in backticks (` ... `).

Instead of printing the output of a command to the screen with echo, it can also be assigned to a variable, as in the following:

```
#!/bin/bash
#command_subs_2.sh

TODAY=`date +%m/%d/%Y`
echo "Today is $TODAY"
```

In this case, the output of date is assigned to the variable TODAY, and then TODAY is printed to the screen with echo. Again make sure that there are no spaces before or after the equal sign.

Let’s improve our Request Tracker with what we’ve learned. In the start message we are going to include the number of open requests:

```
#!/bin/bash
#command_subs_3.sh

tinum=`cat /var/spool/mail/geeko | grep '#' | wc -l`
echo -e "Welcome to the Request Tracker.\nThere are $tinum open requests:\n"
cat /var/spool/mail/geeko | grep '#'
```

The command line **cat /var/spool/mail/geeko | grep '#' | wc -l** counts all lines from /var/spool/mail/geeko which include a **#**. The result is stored in the variable **tinum.** Then this variable is used in the echo command to display the number of open requests.

The output of the script looks like the following now:

```
geeko@linux-rwke:~/Shell_Scripting> ./command_subs_2.sh
Welcome to the Request Tracker.
There are 3 open requests:

#Ticket|London Office|florian@fluvian.de|Telephone|Im trying to call the London
office, but the line is busy all day!
#Ticket|Can not Print!|tux@novell.com|Workstation|The Printer in office is not
working. Can you please check...
#Ticket|Web Server Broken!|websupport@novell.com|Server|Our web server is not
reachable anymore. Please fix!
```

Exercise 5-3 Use Variables and Command Substitution

In this exercise, you learn how to use variables and command substitution.

You can find this exercise in the workbook.

(End of Exercise)

Objective 4 Use Control Structures

With the scripting techniques you have learned so far, you can only develop scripts that run sequentially from the beginning to the end.

In this objective, you learn how to use control structures to make the execution of parts of your script dependent on certain conditions or to repeat script parts.

In this objective we will cover the following;

- Create Branches
- Create Loops

Create Branches

A branch in a script means that a part of your script is only executed under a certain condition. A very common control structure for this uses the **if** command:

```
if condition
then
  commands
fi
```

If a condition is true, then one or more commands are executed.

The **if** statement can be extended with an optional **else** statement, as shown in the following example:

```
if condition
then
  command1
else
  command2
fi
```

In this case command2 is executed when the if condition is not true.

Let’s add an if structure to our Request Tracker. We are adding some code that modifies the start messages. If there are more than 10 open requests, the user sees a warning message.

```
#!/bin/bash
#control_struc1.sh

tinum=`cat /var/spool/mail/geeko | grep '#' | wc -l`

if test $tinum -gt 10
then
    echo -e "Welcome to the Request Tracker.\nThere are $tinum open requests. You
have to do something!\a"
else
    echo -e "Welcome to the Request Tracker.\nThere are $tinum open requests:\n"
fi

cat /var/spool/mail/geeko | grep '#'
```

To check the condition, the program **test** is used. `test $tinum -gt 10` checks if $tinum is greater (-gt) than 10. If this is true, test returns the value 0 otherwise the value 1.

Almost all command line tools have a return value. 0 always means something like “true”or “everything is OK”. Otherwise a value different from 0 is returned. An **if** condition is true when the program used for testing returns 0.

Test can also be used for many other things than checking if one number is greater than the other. The following is an overview of the most important the test options:

- **test STRING1 = STRING2**. The strings are equal
- **test STRING1 != STRING2**. The strings are not equal
- **test INTEGER1 -eq INTEGER2**. INTEGER1 is equal to INTEGER2
- **test INTEGER1 -lt INTEGER2**. INTEGER1 is less than INTEGER2
- **test -e FILE.** FILE exists

For a complete list of all test options, look at the test man page.

When you look at scripts written by someone else, you will also see a different syntax of test. Instead of **test 12 -eq 14** you can also leave out the test command and put the expression in square brackets like **[12 -eq 14]**.

One other thing you might have noticed, is that the lines after **then** and **else** are indented. This is not required but a very common method to identify logical blocks and to make the code more readable.

With **if** you can also create more complex structures in your script:

```
if test $number -eq 10
then
    echo "The value is 10"
elif test $1 -eq 20
then
    echo "The value is 20"
else
    echo "I don't know"
fi
```

With **elif**, you add more conditions in case the one in the initial if statement was not true.

Another way to create multiple branches is **case**. In a case statement, the expression contained in a variable is compared with a number of expressions. Commands are executed for the first expression that matches.

A case statement has the following syntax:

```
case $variable in
  expression1) command1;;
  expression2) command2;;
esac
```

The elif example from above looks like the following when using a case structure:

```
case $number in
    10) echo "The value is 10";;
    20) echo "The value is 20";;
    *) echo "I don't know"
esac
```

The variable $number is compared with 10, 20 and *. * matches for every value and is therefore the default action of the case statement.

Exercise 5-4 Use an if Control Structure

In this exercise, your learn how to use an if control structure.

You can find this exercise in the workbook.

(End of Exercise)

Create Loops

Another common control structure is the loop. A loop is often used when a certain task has to repeated more than once. Instead of repeating the same code in the script, a loop structure can be implemented.

There are a few options for implementing a loop in shell scripts. A very common one is the **for** loop:

```
for variable in element1 element2 element3
do
  commands
done
```

The line starting with **for** defines how many times the code between **do** and **done** has to be execueted. For each pass of the loop, the variable **"variable"** has one of the values defined in the list after **in.**

Lets have a look at an example:

```
#!/bin/bash
#for_loop_1.sh

for i in 1 2 3
do
   echo $i
done
```

The list after **n** contains three elements, the numbers 1, 2 and 3 separated with spaces. This means that the code between **do** and **done** is executed three times and each time the variable **i** has a different value from 1 to 3. When you run this script, it simply outputs 1 2 3.

```
geeko@linux-rwke:~/Shell_Scripting> ./for_loop_1.sh
1
2
3
```

The list defined after **in** is not necessarily static. The **for** loop is very often used to go through a list of files. An easy way to do this is to use * after in.

```
#!/bin/bash
#for_loop_2.sh

for i in *
do
   lower=`echo $i | tr [:upper:] [:lower:]`
   mv $i $lower
done
```

This script loops through all files in the current working directory and renames the files from upper to lower case. * is expanded to a list of all these files by the shell.

For every pass of the loop, the variable **$i** contains one file name. The file name is converted to lower case and stored in the variable **lower**. Then the original file is renamed with **mv** to lower case.

This is just a demo script. For a production script you would have to add some code that makes sure that an existing lower case file is not accidentally overwritten.

Another way of creating a list is a command substitution:

```
#!/bin/bash
#for_loop_3.sh

for i in `find -name "*mp3"`
do
   rm $i
done
```

This script uses **find** (note the backticks) to create a list of all .mp3 files in the current directory and all sub-directories. These files are deleted in the **for** loop.

Another loop construct is the **while** loop:

```
while condition
do
  commands
done
```

Very similar to the while loop is the **until** loop:

```
until condition
do
  commands
done
```

Both loop types depend on a condition. In a **while** loop the commands are executed as long as the condition **is** true and in an **until** loop the commands are executed until the condition becomes true.

We will use a **while** loop to make the output of the Request Tracker more readable:

```
#!/bin/bash
#while_loop_1.sh

tinum=`cat /var/spool/mail/geeko | grep '#' | wc -l`

if test $tinum -gt 10
then
    echo -e "Welcome to the Request Tracker.\nThere are $tinum open requests. You
have to do something!\a"
else
    echo -e "Welcome to the Request Tracker.\nThere are $tinum open requests:\n"
fi

while read line
do
    if echo $line | grep '#' > /dev/null
    then
        echo $line | grep '#' |cut -d '|' --output-delimiter=', ' -f '2 3 4'
    fi
done  < /var/spool/mail/geeko
```

The file **/var/spool/mail/geeko** is fed into a **while** loop using input redirection (<). In the loop the command **read** is used to process the data line by line. For every cycle of the loop, the variable **$line** contains one line of the file. The loop is terminated when all lines are processed (**read** returns a value which is not 0).

In the loop, grep tests if a line contains a "**#**". If yes, **cut** is used to display only the field's subject, email address and category of each line. The following is a description of the cut options in the example:

- **-d '|'.** This option determines the delimiter (in our case the | character), which is used to spilt the output into fields.
- **--output-delimiter=', '**. This option determines, that a "," is used to separat output fields.
- **-f '2 3 4'.** This option finally lets cut only output the fields 2,3 and 4 (subject, email and category).

The output of the Request Tracker looks like the following now:

```
geeko@linux-rwke:~/Shell_Scripting> ./while_loop_1.sh
Welcome to the Request Tracker.
There are 3 open requests:

London Office, florian@fluvian.de, Telephone
Can not Print!, tux@novell.com, Workstation
Web Server Broken!, websupport@novell.com, Server
```

Exercise 5-5 Use a while Loop

In this exercise, you practice how to iterate through a file with a while loop.

You can find this exercise in the workbook.

(End of Exercise)

Objective 5 Use Arithmetic Operators

Shell scripts often use values assigned to variables for calculation. There are several ways to implement this.

The Bash shell comes with built-in support for arithmetic operations, but there are some limitations to this. Specifically, the arithmetic capabilities of Bash are limited in the following ways:

- Only operations with whole numbers (integers) can be performed.
- All values are signed 64-bit values. Thus, possible values range from -2^{63} to $+2^{63}$ -1.

So when using Bash, you might need to use external commands, such as **bc** for floating-point calculations.

The following paragraphs list all possible methods and formats for arithmetic operations. All of them are based on this sample operation:

```
A=B+10
```

- **Use the external command expr (Bourne shell compatible)**

```
A=`expr $B + 10`
```

  Since an external command is used, this method will also work with the traditional Bourne shell. Scripts using external commands will always perform slower than those relying on built-in commands.

- **Use the Bash built-in command let**

```
let A="$B + 10"
```

  In Bash, you can use the let command to perform an arithmetic expression.

- **Use arithmetic expressions inside parentheses or brackets (two different formats)**

  ```
  A=$((B + 10))
  ```

  or

  ```
  A=$[B + 10]
  ```

  Arithmetic expressions can be enclosed in double parentheses or in brackets for expansion by Bash. Both $((. . .)) and $[. . .] are possible, but the latter is considered deprecated and should be avoided.

- **Use the built-in command declare**

  ```
  declare -i A
  declare -i B
  A=B+10
  ```

  This declares a variable as an integer.

  If all variables involved in a calculation have previously been declared as integers through **declare -i**, arithmetic evaluation of these variables happens automatically when a value is assigned to them.

  This means that the variable B, for example, does not have to be prefixed with the $ to be evaluated.

With the **expr** command, only the following five operators are available: + , - , * , / , and % (modulo, remainder of a division). Additional operators (which are identical to those of the C programming language) can be used with all of the above Bash formats.

For a complete list, consult the man page for bash.

We are using an arithmetic operator to assign a number to each request in the Request Tracker:

```
#!/bin/bash
#arithmetic_1.sh

tinum=`cat /var/spool/mail/geeko | grep '#' | wc -l`

if test $tinum -gt 10
then
    echo -e "Welcome to the Request Tracker.\nThere are $tinum open requests. You
have to do something!\a"
else
    echo -e "Welcome to the Request Tracker.\nThere are $tinum open requests:\n"
fi

number=1
while read line
do
    if echo $line | grep '#' > /dev/null
    then
        echo -e "$number|$line" | grep '#' |cut -d '|' --output-delimiter=', ' -f
'1 3 4 5'
        ((number++))
    fi
done  < /var/spool/mail/geeko
```

We create the variable ***$number*** and assign the value 1 to it. In the **while** loop $number is added to the echo output. The expression `((number++))` increments with every loop cycle the value of $number by 1. This way the requests are numbered starting with 1.

The following is the output of the Request Tracker with line numbering:

```
geeko@linux-rwke:~/Shell_Scripting> ./arithmetic_1.sh
Welcome to the Request Tracker.
There are 3 open requests:

1, London Office, florian@fluvian.de, Telephone
2, Can not Print!, tux@novell.com, Workstation
3, Web Server Broken!, websupport@novell.com, Server
```

Exercise 5-6 Use Arithmetic Operators

In this exercise, you learn how to use arithmetic operators.

You can find this exercise in the workbook.

(End of Exercise)

Objective 6 Read User Input

One way to read user input is to use the command **read**. The read command takes a variable as an argument and stores the read input in the variable. The variable can then be used to process the user input.

The following example reads user input into the variable with the name VARIABLE:

```
read VARIABLE
```

The script pauses at this point, waiting for user input, until the **Enter** key is pressed. To tell the user to enter something, you need to print (echo) a line with some information, such as the following:

```
echo "Please enter a value for the variable:"
read VARIABLE
```

Now we are extending the Request Tracker and make it a bit more interactive. We are using read so that the user can enter a command after the open requests have been displayed.

```
#!/bin/bash
#read_user_1.sh

tinum=`cat /var/spool/mail/geeko | grep '#' | wc -l`

if test $tinum -gt 10
then
    echo -e "Welcome to the Request Tracker.\nThere are $tinum open requests. You
have to do something!\a"
else
    echo -e "Welcome to the Request Tracker.\nThere are $tinum open requests:\n"
fi

input='blank'

until test $input = "quit"
do

    number=1
    while read line
    do
        if echo $line | grep '#' > /dev/null
        then
           echo -e "$number|$line" | grep '#' |cut -d '|' --output-delimiter=', '
-f '1 3 4 5'
            ((number++))
        fi
    done  < /var/spool/mail/geeko

read input
done
```

First the variable input is initialized with the string “blank”. Then we enter the while loop. At the end of the loop we read user input into the variable $input. In case that the user enters “quit”, the script is terminated. In case of any other input, the requests are listed again.

At the moment this does make much sense yet, but soon we will add more input options to the script.

Exercise 5-7 Read User Input

In this exercise, you learn how to read user input and process the input in your script.

You can find this exercise in the workbook.

(End of Exercise)

Objective 7 Use Arrays

Arrays are basically variables that can hold more than one value. To identify a value in an array, a numerical index is used. The index is written in square brackets after the array name.

```
lines[1]=”Hello World”
```

This line assigns the string “Hello World” to the index **1** of the array with the name **lines**.

To access a value in an array, you have to specify an index and put curly brackets around the array name:

```
echo ${lines[1]}
```

Arrays are very usefull to store list data like a list of files or the requests in our Resquest Tracker.

Have a look at the following modifications: (From now on we will only list those partsof the code, that have been modified. You can find the full version on the course CD.)

```
#!/bin/bash
#array_1.sh
#Full version on CD

input='blank'
until test $input = "quit"
do
    reqnr=1

    while read line
    do
        if echo $line | grep '#' > /dev/null
        then
                lines[$reqnr]="$reqnr|$line"
                ((reqnr++))
        fi
    done < /var/spool/mail/geeko

    for ((i=1;i<=${#lines[@]};i++))
    do
        echo -e ${lines[$i]} | cut -d '|' --output-delimiter=', ' -f '1 3 4 5'
    done

    unset lines
    read input
done
```

In the while **loop**, the requests are stored into the array **$lines**. The variable *$reqnr* which is used as index, is incremented in every cycle of the **while** loop.

In the **for** loop, the content of the array is echoed to the terminal. For this a different syntax of the for loop is used, which is similar to the **for** loop in the C programming language.

for ((i=1;i<=${#lines[@]};i++))** means that the loop runs as long as the variable *$i* is smaller or equal (<=) to the number of elements in the array $lines. **${#lines[@]} is a way to access the number of elements in an array.

The index variable $i is initially set to 1 and incremented with every cycle of the for loop.

At the end of the while loop, the array lines are deleted with the **unset** command. Otherwise the array cannot be reused in the next run of the while loop.

Right now these changes do not make much sense, as we are not adding any new functionality. However, the usage of an array will make further changes much easier.

Exercise 5-8 Use Arrays

In this exercise, you learn how to use arrays.

You can find this exercise in the workbook.

(End of Exercise)

Objective 8 Finalize the Course Project

In this objective we would like to add the following features to the Request Ticker:

- View Request Details
- Delete Requests

View Request Details

Right now only the subject, the sender and the category of a request is displayed. Now we are adding an option which lets the user view the description of a request:

```
#!/bin/bash
#finalize1.sh
#Full Version on CD

input='l'
until test $input = "quit"
do
    reqnr=1
    while read line
    do
        if echo $line | grep '#' > /dev/null
        then
                lines[$reqnr]="$reqnr|$line"
                ((reqnr++))
        fi
    done < /var/spool/mail/geeko

    com=`echo $input | cut -d ':' -f 1`
    req=`echo $input | cut -d ':' -f 2`

    case $com in
        'i') echo -e `echo ${lines[$req]} | cut -d '|' --output-delimiter='\n' -f
'3 4 5 6'` ;;
        'l') for ((i=1;i<=${#lines[@]};i++))
             do
                echo -e ${lines[$i]} | cut -d '|' --output-delimiter=', ' -f '1 3
4 5'
             done
             ;;
          *) echo "Unknown Command"
    esac

    unset lines
    read input
done
```

The user can enter commands in the form **<command>:<request number>**. The command is stored in the variable $com and the request number is stored in $req.

Then a case statement for $com is used to determine what to do. If the command is just “l”, the requests are listed like before. If the command i (information) is entered, all details of the specified request are listed. For any other input, “Unknown Command” is echoed.

Delete Requests

Now we are going to add a command that lets the user delete requests. We are using the existing case structure:

```
#!/bin/bash
#finalize2.sh
#Full version on course CD
input='l'
until test $input = "quit"
do
    reqnr=1
    while read line
    do
        if echo $line | grep '#' > /dev/null
        then lines[$reqnr]="$reqnr|$line"
             ((reqnr++))
        fi
    done < /var/spool/mail/geeko

    com=`echo $input | cut -d ':' -f 1`
    req=`echo $input | cut -d ':' -f 2`

    case $com in
        'i') echo -e `echo ${lines[$req]} | cut -d '|' --output-delimiter='\n' -f
'3 4 5 6'` ;;
        'l') for ((i=1;i<=${#lines[@]};i++))
            do
            echo -e ${lines[$i]} | cut -d '|' --output-delimiter=', ' -f '1 3 4 5'
            done
             ;;
        'd') rm /var/spool/mail/geeko
             for ((i=1;i<=${#lines[@]};i++))
             do
                 if test $i -ne $req
                 then
                 echo ${lines[$i]} | cut -d '|' --output-delimiter='|' -f '2 3 4 5
6' >> /var/spool/mail/geeko
                 else
                     echo "Request $req deleted."
                 fi
             done
             ;;
          *) echo "Unknown Command"
    esac
    unset lines
    read input
done
```

When the user enters a command like **d:3** the specified request is deleted from the mailbox file. First the existing file (/var/spool/mail/geeko) is deleted and then a new one is created from the data in the $lines array. All lines in the array are appended to the file, except for the one that has to be deleted.

Exercise 5-9 Add more Commands

In this exercise, you add more command options to the Request Tracker.

You can find this exercise in the workbook.

(End of Exercise)

Objective 9 Use Advanced Scripting Techniques

In this objective, you learn about the following advanced scripting techniques, that will help you solve common problems of script development:

- Use Shell Functions
- Read Options with getopts

In this objective we are not making any more changes to the Request Tracker project. However, it would be great if you would go on improving the Tracker by using functions or command line options.

Use Shell Functions

Sometime you need to perform a task multiple times in a shell script. Instead of writing the same code again and again, you can use functions.

Shell functions act like script modules because they make an entire script section available under a single name. Shell functions are normally defined at the beginning of a script. You can store several functions in a file and include this file whenever the functions are needed.

There are two ways to declare a function in a script.The following is the basic syntax of a function:

```
functionname () {
  commands
  commands
}
```

The following generates a function with the **function** command:

```
function functionname {
  commands
  commands
}
```

The function name can be composed of any regular character string.

The following is a simple function that creates a directory and then changes to that directory:

```
# mcd: mkdir + cd; creates a new directory and
# changes into that new directory right away

mcd (){
  mkdir $1
  cd $1
}
```

After having been created, this function can be called in a shell script, as in the following:

```
...
mcd /tmp/new_directory
...
```

The parameter /tmp/new_directory is called an *argument*. Within a function, arguments can be accessed with the variables $1, $2, $3, and so on, depending on the number of arguments passed to the function.

The following function can be used to create a pause in a script. The script resumes only after the Enter key is pressed:

```
# pause: causes a script to take a break

pause (){
  echo "To continue, hit RETURN."
  read q
}
```

You can also create functions that stop their processing from within, similar to exiting a loop (iteration), with the commands break and continue.

To exit a function, use the command return. If return is called without an argument, the return value of the function is identical to the exit status of the last command executed in that function.

Otherwise, the return value is identical to the one supplied as an argument to return.

Exercise 5-10 Use Shell Functions

In this exercise, you learn how to use shell functions.

You can find this exercise in the workbook.

(End of Exercise)

Read Options with getopts

With the shell built-in command **getopts**, you can extract the options supplied to a script on the command line. The shell interprets command line arguments as command options only if they are prefixed with a **-** (the default when using the shell interactively).

This makes it possible to place options in different positions on the command line and to supply them in an arbitrary order.

This means that the following command:

cp -dpR *.txt texts/

achieves the same thing as the command

cp -R *.txt -d texts/ -p

getopts recognizes options in the same way. The following is the getopts syntax:

getopts *optionstring variable*

The ***optionstring*** describes all options to be recognized. For instance, getopts abc declares a, b, and c as the options to be processed.

If a parameter is expected for the option (such as **-m maxvalue**), the corresponding option must be followed by a : in the string (as in **getopts m:**).

The option string is followed by a variable to which all the command-line options specified are assigned as a list.

The getopts command is mostly frequently used in a while loop together with case to define which command to execute for a given option, as in the following:

```
while getopts abc: variable
do
  case $variable in
    a ) echo "The option -a was used." ;;
    b ) echo "The option -b was used." ;;
    c ) option_c="$OPTARG"
        echo "Option c has been set." ;;
  esac
done

echo $option_c
```

If the option **-a** or **-b** is used, the script prints out a message that the corresponding option was used. If the option **-c** *value* is used, the value is assigned to the variable option_c, which is printed to the screen at the end of the script.

The parameter of an option can be accessed with the variable OPTARG.

Objective 10 Learn About Useful Commands in Shell Scripts

The following objective gives you an overview of useful commands that can be used in shell scripts. It's intended to be a reference and not something you have to read from the beginning to the end.

We will talk about the following:

- Use the cat Command
- Use the cut Command
- Use the date Command
- Use the grep and egrep Commands
- Use the sed Command
- Use the test Command
- Use the tr Command

Use the cat Command

When combined with the here operator (**<<**), the **cat** command is a good choice to output several lines of text from a script. In interactive use, the command is mostly run with a filename as an argument, in which case cat prints the file contents on standard output.

Use the cut Command

You can use the **cut** command to cut out sections of lines from a file, so only the specified section is printed on standard output.

The command is applied to each line of text as available in a file or on standard input. You can use **cut -f** to cut out text fields. **cut -c** works with the specified characters.

You can specify single sections (characters or fields) or several sections. The default delimiter to separate fields from each other is a tab, but you can specify a different field separator with the **-d** option.

The following are some examples of using cut:

```
cut -d : -f1 /etc/passwd
root
bin
daemon
lp
mail
news
```

The above command specifies that the field separator should be a colon. In every line of /etc/passwd, the field that comes before the first colon is taken and printed to stdout:

```
ls -l somedir/ | cut -c 35- | sort -n
687 Sep 20 17:06 file2
2199 Sep 20 17:05 file1
6593 Sep 20 17:06 file3
```

The above command takes the output of the **ls** command and cuts out everything from the thirty-fifth character. This is piped to sort, so the final output is sorted according to file size.

Use the date Command

You can use the **date** command whenever there is a need to obtain a date or time string for further processing by a script. Without any options specified, the command´s output looks like the following:

```
date
Fre Sep 03 14:18:12 CEST 2004
```

The date command lets you change the output format in almost every detail. With the **-I** option (as in the following), date prints the date and time in ISO format (which is the same as if the options had been +%Y-%m-%d):

```
date -I
2004-09-03

date +%m-%d %H:%M
09-03 14:19

date +%D, %r
09/03/02, 02:19:58 PM

date +%d.%m.%y
03.09.02

date +%d.%m.%Y
03.09.2004

date +%e.%-m.%y,   %l.%M   %p
3.9.02, 2.20 PM

date +%A, %e. %B   %Y
Friday, 3. September 2004
```

To view a list with all the possible format options for date, see **man date**. In any case, you should be able to customize the output to exactly match the requirements of your script.

Use the grep and egrep Commands

The command **grep** and its variant **egrep** are used to search files for certain patterns, and use the following syntax:

grep *searchpattern filename* ...

The command prints lines that contain the given search pattern. You can specify several files, from which **grep** will print the matching line and the corresponding filenames.

Several options are available to specify that only the line number should be printed, for instance, or that the matching line should be printed together with leading and trailing context lines.

Search patterns can be supplied in the form of regular expressions, although the **bare grep** command is limited in this regard.

To search for more complex patterns, use the **egrep** command, which accepts extended regular expressions. As a simple way to deal with the difference between the two variants, make sure you use **egrep** in all of your shell scripts.

The regular expressions used with egrep need to be in accordance with the standard regex syntax.

To avoid having special characters in search patterns interpreted by the shell, enclose the pattern in quotation marks, as in the following:

```
tux@DA1:~> egrep (b|B)lurb file*
bash: syntax error near unexpected token  |

tux@DA1:~> egrep "(b|B)lurb" file*
file1:blurb
file2:Blurb
```

Use the sed Command

The sed program is a stream editor, an editor used from the command line rather than interactively. sed performs text transformations on a line-by-line basis.

You can specify sed commands either directly on the command line or in a special command script loaded by the program on execution.

The following is the syntax for the sed command:

sed *editing-command filename*

The available editing commands are single-character arguments such as the following:

- **d:** Delete
- **s:** Substitute (replace)
- **p:** Output line
- **a:** Append after

As with other commands, the output of sed normally goes to standard output, but it can also be redirected to a file.

Each sed command must be preceded by an exact address or address range specifying the lines to which the editing command applies.

Apart from the single-character commands for text transformations, you can also specify options to influence the overall behavior of the sed program.

The following are some important command line options for sed:

- **-n, --quiet, --silent.** By default, sed will print all lines on standard output after they have been processed. This option suppresses the output so sed only prints those lines for which the **p** editing command has been given to explicitly re-enable printing.
- **-e *command1* -e *command2* ....** This option is necessary when specifying two or more editing commands. It must be inserted before each additional editing command.
- **-f *filename*.** With this option, you can specify a script file from which sed should read its editing commands.

For many editing commands, it is important to specify the exact line or lines that should be processed by the command. One of the more frequently used address labels is $, which stands for the last line.

The following are 2 examples of the sed command:

- **sed -n ‘1,9p’ somefile**

  This command prints only lines 1 through 9 on stdout.

- **sed ‘10,$d’ somefile**

  This command deletes everything from line 10 to the end of the file and also prints the first 9 lines of somefile.

You can use a regular expression to define the address or address range for an editing command. Regular expressions must be enclosed in forward slashes. If an address is defined with such an expression, sed processes every line that includes the given pattern.

The following is an example of using regular expressions:

sed -n ‘/Murphy.*/p’ somefile

This example prints all lines that have the pattern Murphy.* in them.

If you want **sed** to perform several editing commands for the same address, you need to enclose the commands in braces, as in the following:

sed ‘1,10{*command1* ; *command2*}’

The following lists the most important editing commands available for sed:

Table 5-1

Command	Exampleo	Editing Action
d	sed 10,$d *file*	Delete line.
a	sed ‘a\text\text’ *file*	Insert text before the specified line.
i	sed ‘i\text\text’ *file*	Replace specified lines with the text.

Table 5-1 *(continued)*

Command	Exampleo	Editing Action
c	sed ‘2000,$c\text ‘ *file*	Replace specified lines with the text.
s	sed s/*x*/*y*/*option*	Search and replace. The search pattern *x* is replaced with pattern *y*. The search and the replacement pattern are regular expressions in most cases and the search and replace behavior can be influenced through various options.
y	sed y/abc/xyz/	(yank) Replace every character from the set of source characters with the character that has the same position in the set of destination characters.

You can use the following options with the **s** command (search and replace):

- **I.** Do not distinguish between uppercase and lowercase letters.
- **g.** Replace globally wherever the search pattern is found in the line (instead of replacing only the first instance).
- **n.** Replace the nth matching pattern only.
- **p.** Print the line after replacing.
- **w.** Write the resulting text to the specified file rather than printing it on stdout.

The following are some examples of using the **s** command:

- **sed ‘s/:/ /’ /etc/passwd**

  This command replaces the first colon in each line with a space.

- **sed ‘s/:/ /g’ /etc/passwd**

  This command replaces all colons in all lines with a space.

- **sed ‘s/:/ /2’ /etc/passwd**

  This command replaces only the second colon in each line with a space.

- **sed -n ‘s/\([aeiou]\)/\1\1/Igp’**

  This command replaces all single vowels with double vowels. The example shows how matched patterns can be referenced with “**\1**” if the search pattern is given in parentheses (which have to be escaped). The **I** option ensures that sed ignores the case.

  The **g** option causes characters to be replaced globally. The **p** option tells sed to print all lines processed in this way.

Use the test Command

The **test** command exists both, as a built-in command and as an external command. It is used to compare values and to check for files and their properties (whether a file exists, whether it is executable, and so on).

If a tested condition is true, test returns an exit status of 0; if the condition is not true, the exit status is 1. In shell scripts, ttest is used mainly to declare conditions to influence the operation of loops, branches, and other statements.

The following is the test syntax:

test *condition*

You can use the test command to do the following:

- Test whether a file exists. Some of the available options are:

Table 5-2

Option	Description
-e	File exists
-f	File exists and is a regular file
-d	File exists and is a directory
-x	File exists and is an executable file

- Compare 2 files. Some of the available operators are:

Table 5-3

Option	Description
-nt	Newer than
-ot	Older than
-ef	Refers to the same inode (such as a hard link)

- Compare 2 integers. The available operators are:

Table 5-4

Option	Description
-eq	Equal
-ne	Not equal
-gt	Greater than
-lt	Less than
-ge	Greater than or equal
-le	Less than or equal

- Test strings. The available operators are:

Table 5-5

Option	Description
test -z *string*	Exit status is 0 (true) if the string has zero length (is empty).
test *string*	Exit status is 0 (true) if the string has nonzero length (consists of at least one character).
test *string1* = *string2*	Exit status is 0 (true) if the strings are equal.
test *string1* != *string2*	Exit status is 0 (true) if the strings are not equal.

- Combine tests. The available operators are:

Table 5-6

Option	Description
test ! *condition*	Exit status is 0 (true) if the condition is not true.
test *condition1* -a *condition2*	Exit status is 0 (true) if both conditions are true.
test *condition1* -o *condition2*	Exit status is 0 (true) if either condition is true.

For more detailed information about test, enter **help test** or **man test** (the built-in test command and the external one have identical features).

Use the tr Command

The **tr** command translates (replaces) or deletes characters. It reads from standard input and prints the result on standard output. With tr, you can replace regular characters or sequences of such characters and special characters like **\t** (horizontal tab) or **\r** (return).

A complete list of all special characters handled by tr is included in the man page of the program.

The following is the standard syntax of **tr**:

tr ***set1 set2***

The characters included in ***set1*** are replaced with the characters included in **set2**.

The following is an example of using the **tr** command:

cat text-file | tr a-z A-Z

This a command causes all lowercase characters in a file to be changed to uppercase, and the result is printed to stdout.

You can use **tr** to delete characters from the first set by entering the following:

tr -d set1

This will not translate anything; it only deletes the ones included in **set1**, printing the rest to standard output.

The following is another example of using the **tr** command:

VAR='echo $VAR | tr -d %'

In this example, tr deletes the percent sign from the original value of VAR and the result is assigned as a new value to the same variable.

By entering a command like

tr -s ***set1 char***

you can also use **tr** to replace a set of characters with a single character.

Summary

Objective	Summary
1. Understand the Course Project	As course project we are developing a shell script that acts as a Request Tracker. Requests can be entered through a web interface. The shell script reads the requests from a mailbox file and lets the user manage them.
2. Use Basic Script Elements	A shell script is basically an ASCII text file containing commands to be executed in sequence. To allow this, it is important that permissions for the script file are set to "**r**" (readable) and "**x**" (executable) for the user that runs it. Any command you know from the command line can also be used in a shell script. A shell script always starts with a line like **#!/bin/bash** to indicate the interpreter of the script.

Objective	Summary
3. Understand Variables and Command Substitution	Variables are an important component of all programming languages. You can understand variables as containers that hold data. Instead of the data itself, the variable is used in the program code. When you assign a variable, you just use the name of the variable. When you access the data of a variable, you put a $ before the variable name. The term *command substitution* basically means that the output of a command is used in a shell command line or a shell script. The commands are included in backticks (` ... `).
4. Use Control Structures	With the scripting techniques you have learned so far, you can only develop scripts that run sequentially from the beginning to the end. In this objective, you learned how to use control structures to make the execution of parts of your script dependent on certain conditions or to repeat parts of a script. Branches can be created with **if** or **case**. Loops are implemented with **while** or **until**.

Objective	Summary
5. Use Arithmetic Operators	Shell scripts often use values assigned to variables for calculation. There are several ways to implement this. The Bash shell comes with built-in support for arithmetic operations, but there are some limitations to this.The arithmetic capabilities of Bash are limited in the following ways: Only operations with whole numbers (integers) can be performed. All values are signed 64-bit values. Thus, possible values range from -2^{63} to $+2^{63}$ -1. So when using Bash, you might need to use external commands, such as **bc** for floating-point calculations.
6. Read User Input	One way to read user input is to use the command **read**. The read command takes a variable as an argument and stores the read input in the variable. The variable can then be used to process the user input. The following example reads user input into the variable with the name VARIABLE: read VARIABLE

Objective	Summary
7. Use Arrays	Arrays are basically variables that can hold more than one value. To identify a value in an array, a numerical index is used. The index is written in square brackets after the array name. lines[1]=”Hello World” This line assigns the string “Hello World” to the index **1** of the array with the name **lines**. To access a value in an array, you have to put braces around the array name: cho ${lines[1]}
8. Finalize the Course Project	In this objective, we added a detailed request view and the option to delete requests.
9. Use Advanced Scripting Techniques	In this objective, you learned how to create a use shell functions and how to evaluate command line options.
10. Learn About Useful Commands in Shell Scripts	This objective gives you an overview of useful commands that can be used in shell scripts.

SECTION 6 Compile Software from Source

In this section, you learn how to compile and install software that is available as source code.

Objectives

1. Understand the Basics of C Programming
2. Understand the Concept of Shared Libraries
3. Understand the GNU Build Tool Chain
4. Perform a Standard Build Process

Introduction

Although SUSE Linux Enterprise Server 10 is shipped with software packages for almost all purposes, you might want to install software from other sources.

Sometimes OpenSource projects or third-party vendors provide RPM packages that are made for SUSE Linux Enterprise Server 10 and can be installed with the RPM command line tool or with YaST.

In many cases, however, open source projects provide only tar archives with the source code of an application. In this section you learn how to compile and install software from these source archives.

Objective 1 Understand the Basics of C Programming

Most applications (and the Linux kernel) are written in the C or C++ programming language. To compile and install software from source achives, it is useful to have a basic understanding of C and the steps which are necessary to turn program code into an executable file.

The C++ language is considered the successor of C. The syntax of C++ is very similar to C, but C++ lets you create object-oriented code. However, many applications and the Linux kernel are still written in C.

In this objective, you learn the following C programming basics:

- The Difference Between Source Code and an Executable
- The Structure of a Simple C Program
- Compile a Simple C Program

The Difference Between Source Code and an Executable

There are basically 2 different types of programing languages:

- **Script languages.** Applications written in a script language can be easily developed with just a text editor. The script files can be directly executed with the help of an interpreter program.

  Examples of script languages are Perl, PHP, Python, and the Shell scripting language.

- **Compiler languages.** Applications written with these programing languages can also be created with a text editor, but normally there is no interpreter software available.

  Before the code can be executed, it needs to be converted into a binary format that can be directly executed by the CPU. This conversion is done by a special program called the *compiler.*

Examples of compiler languages include C, C++, and Fortran.

The following are advantages and disadvantages of each programing language type:

Table 6-1

Language Type	Advantages	Disadvantages
Script language	■ Most script languages are relatively easy to learn. ■ The development process in a script language is rather fast. ■ Script programs can basically run on all platforms where the interpreter is available, without changing the program code.	■ The execution of script application is rather slow. ■ It´s not possible to do system programming with scripting languages. (such as operating systems or device drivers).
Compiler language	■ The execution is very fast compared with scripting languages. ■ It is possible to do system programming (such as operating systems or device drivers).	■ Compiler languages can be pretty difficult to learn. ■ The development process usually takes longer. ■ The source code needs to be recompiled before it can be executed on a different platform.

The summary in this table is not complete. There are programming languages like Java that are classified between the described programming language types.

The Structure of a Simple C Program

The following is the source code of a simple C program:

```
#include <stdio.h>

int main(void)
{
  char name[80];

  printf("Please enter your name: ");
  scanf("%s", name);

  printf("Your name is: %s\n", name);

  return(0);
}
```

This program prompts the user to enter his name, and then it prints out Your name is: and the name the user has entered.

The following describes each line:

- **#include <stdio.h>**

  This line is a *preprocessor directive***.** Before the actual source code is compiled into a binary file, the file is processed by the preprocessor. In this example, the directive instructs the preprocessor to include the file stdio.h.

  A file such as stdio.h contains information about functions that are used later in the program. This is a typical characteristic of C. Every variable or function needs to be declared before it can be used.

  Files that contain function declarations are called *header files* and their filenames typically end in .h.

- **int main(void)**

  This line starts the main function of the program. This is the function that is initially called when the program is started. A program can consist of more functions, but it must have at least a main function.

  The function head normally consists of 3 parts:

  - The type of the return value, in this case, int, an integer.
  - The function name. The main function always has the name main.
  - The arguments of the function, listed in parentheses. The keyword void means the function does not expect an argument.

- **{**

  Every thing that belongs to a function is included in curly brackets. This line starts the block of the main function.

- **char name[80];**

  This line declares a variable name. The type of the variable is char, a variable type for a single character. In this example, since you need to store more that just one character in the variable, declare 80 char variables [80].

  This shows another characteristic of C. It is not enough to declare a variable before you can use it, you need to specify the type of variable. In the example above, it would not be possible to store other data like integers in the variable.

  Another important element is the semicolon at the end of the line. Every declaration and every function call in C must end with a semicolon.

- **printf("Please enter your name: ");**

  This line prints out the message Please enter your name:. It uses the function printf for this purpose. This line also shows a typical function call in C. The arguments are given in parentheses after the function name.

- **scanf("%s", name);**

  In this line the function scanf is used to read the input of the user. The function takes two arguments: a format string %s, which determines the way the input should be handled by the function, and the variable name, in which the input should be stored.

- **printf("Your name is: %s\n", name);**

  This line uses the function printf again, this time to print out the entered name of the user. The function takes 2 arguments in this case, the string Your name is: and the variable name, which holds the name of the user.

  The content of name is output at the position of the %s format string, which acts like a placeholder in this case.

- **return(0);**

  This line uses the return function to return 0 as the value of the function. Because this is the main function, it also determines the return value of whole program.

- **}**

  This curly bracket closes the main function and the whole program.

  The code of C programs should be saved in a file ending in .c. In this example, the filename **my_name.c** is used.

Compile a Simple C Program

To execute the previously described program, it needs to be compiled into a binary file. For this you need a C compiler. The standard C compiler in Linux is the gcc, the Gnu C Compiler.

To compile the simple example program, you invoke the compiler with the following command:

gcc my_name.c -o my_name

First, the name of the file that contains the source code is passed to gcc, followed by the -o option and the name of the output binary file.

After the compilation has finished, the binary can be started like any other command line program, as in the following:

```
./my_name
Please enter your name: Florian
Your name is: Florian
```

Exercise 6-1 Compile a Simple C Program

In this exercise, you compile a simple C program.

You can find this exercise in the workbook.

(End of Exercise)

Objective 2 Understand the Concept of Shared Libraries

Many tasks on a system are performed by more than one application. For example, the opening and displaying of PNG image files is performed by the web browser, the graphic program, and other applications.

It would not make sense to implement the functionality of opening PNG files again and again in every application. Therefore, program functionality can be stored in *shared libraries*.

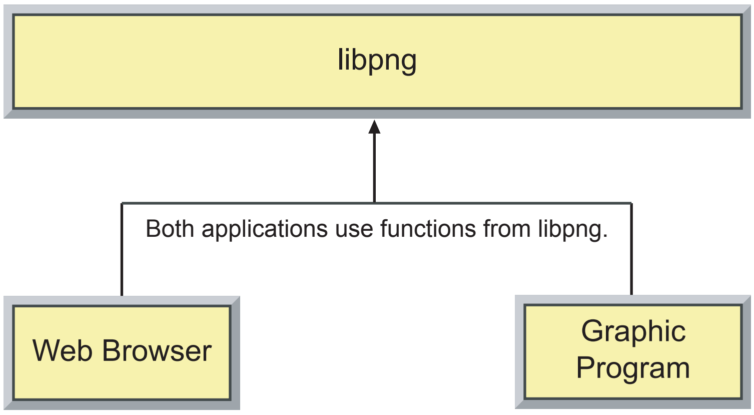
In the case of opening and displaying PNG files, the functionality is provided by the shared library libpng.

Physically, a shared library is a file on the hard disk that is loaded into the main memory when an application is started that requires the functionality of the library.

The task of finding and loading the required libraries is performed by the program **ld**.

The following illustrates how shared libraries work:

Figure 6-1



A shared library consists of 2 parts:

- The shared library file itself, which is loaded when a program requires a function of the corresponding library. The filenames for these files start with lib and end in .so (shared object). For example: **libpng.h**
- The header files of the library, which contain the functions declaration of the library. The filenames of these files end in .h (header file). For example: **png.h**

As described at the beginning of this section, the C programing language requires that every function used in a program has to be declared.

This means that the header files of a library need to be installed on a system to compile software that uses functions of that library. To run already compiled software, only the shared library files are necessary.

Because the software that ships with SUSE Linux Enterprise Server 10 is already compiled, the header files of some libraries are not installed by default. These libraries are split into two packages: the software packages that contain the header files have the extension **-devel** attached to the package name.

For example, the package libpng contains the shared library, and the package **libpng-devel** contains the corresponding header files.

When you run the configure script, it sometimes prompts you about missing libraries that should be installed on the system. If you install the required packages with YaST, you have to make sure that you select both the shared library and the corresponding devel package.

The text file **ARCHIVES.gz** at the top level of the installation media contains a list of all files that are shipped with SUSE Linux Enterprise Server 10. This file can be used to find out, which library file is included in which RPM package.

The following command line searches for the header file **png.h** of the PNG library.

```
zcat ARCHIVES.gz | grep png.h
```

zcat is used because the file ARCHIVES.gz is compressed with Gzip. In the output of the command line, a list matching filenames and their corresponding RPM package is listed. The following line shows, that png.h can be found in the package libpng-devel-1.2.8-19.2.i586.rpm (output shortened):

```
[..] libpng-devel-1.2.8-19.2.i586.rpm: [..] /usr/include/libpng12/png.h
```

Objective 3 Understand the GNU Build Tool Chain

In most cases programs use more than one source code file. In order to structure the source, developers tend to spread the code over multiple files.

It would be very difficult to compile a program with multiple source code files manually on the command line. Fortunately some tools are available to manage the compilation process.

In this objective, you learn how to do the following to perform a standard build process:

- Use configure to Prepare the Build Process
- Use make to Compile the Source Code
- Use make install to Install the Compiled Program
- Install the Required Packages for a Build Environment

Use configure to Prepare the Build Process

Before the actual compilation process can be started, you must prepare the source code with a **configure** script. This needs to be done for the following reasons:

- Many applications can be compiled on different UNIX systems, Linux distributions, and hardware platforms. To make this possible, the build process needs to be prepared for the actual environment.
- The build process itself is controlled by a program called make. The instructions for how to compile the different source files are read from Makefiles. The configure script generates these Makefiles depending on the system environment.
- You can use configure to enable or disable certain features of an application.

To run the configure script, you need to use the following command at the top of the source directory:

./configure

To enable or disable certain features of an application, configure takes additional arguments. The available arguments depend on the application that will be compiled.

You can use the following command to list all available configure options:

./configure --help

Use make to Compile the Source Code

You use the tool **make** to compile multiple source files in the correct order. Make is controlled by Makefiles. Normally, these Makefiles are generated by the configure script, but you can also create them manually.

You can also use make to install the program files on the hard disk.

The following is a simple Makefile that shows how make works:

```
# Makefile for my_name

all: my_name

my_name: my_name.c
gcc my_name.c -o my_name

install: my_name
install -m 755 my_name /usr/local/bin/my_name

uninstall: /usr/local/bin/my_name
rm -f /usr/local/bin/my_name

clean:
rm -f my_name
```

This Makefile can perform the following tasks:

- Compile the program from source
- Install the program
- Uninstall the program
- Clean up the directory where the compilation is performed

Every Makefile consists of targets, dependencies, and commands for the targets. Targets and dependencies are separated by a colon. The commands must be placed under the target, indented with one tab space. A **#** introduces comments.

If you execute the command make while you are in the respective directory, the program make will search this directory for the files GNUMakefile, Makefile, or makefile.

If make is executed without any parameters, the first target of a Makefile is used. In the example above, this is all. This target is associated with the target my_name, which specifies the step to take: compile the file my_name.c with gcc.

The command make can also be used with individual targets. For example, the command **make install** (as root) installs the binary file at the specified location and **make uninstall** removes the binary file.

Even large software projects are created in the same way, but the Makefiles are much more extensive and complex. If the software will be compiled to a functional program on multiple architectures, things are much more complicated.

For this reason, the Makefile is usually generated by the configure script.

Use make install to Install the Compiled Program

The last step when installing a program from source is to install the binary file and additional files belonging to the application.

This step is usually done with make and an install target in the corresponding Makefile.

You can perform the installation with the following command:

make install

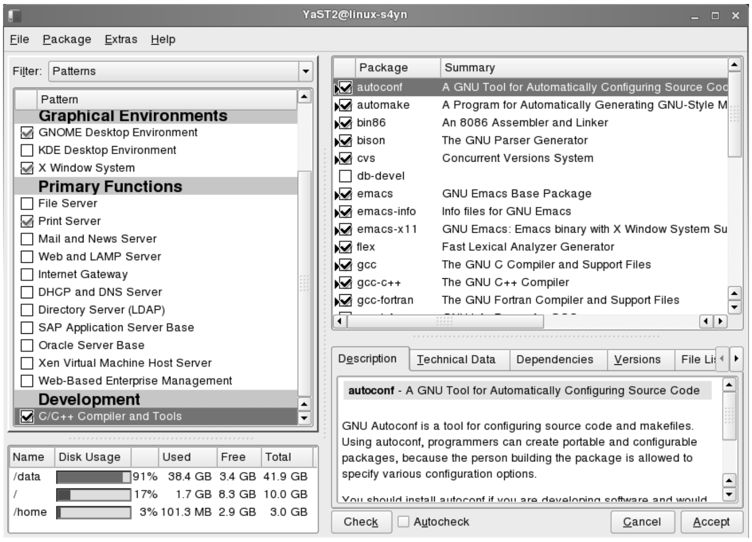
You must enter this command as root at the top level of the source directory.

Install the Required Packages for a Build Environment

A lot of different software packages are required to perform the described build process. The easiest way to install all required packages is to select the pattern C/C++ Compiler and Tools in the YaST package manager.

To access the predefined selections, select **Patterns** from the filter drop-down list as shown in the following:

Figure 6-2



YaST2@linux-s4yn

Filter: Patterns

Pattern
Graphical Environments
✓ GNOME Desktop Environment
☐ KDE Desktop Environment
✓ X Window System
Primary Functions
☐ File Server
✓ Print Server
☐ Mail and News Server
☐ Web and LAMP Server
☐ Internet Gateway
☐ DHCP and DNS Server
☐ Directory Server (LDAP)
☐ SAP Application Server Base
☐ Oracle Server Base
☐ Xen Virtual Machine Host Server
☐ Web-Based Enterprise Management
Development
✓ C/C++ Compiler and Tools

	Package	Summary
✓	autoconf	A GNU Tool for Automatically Configuring Source Coc
✓	automake	A Program for Automatically Generating GNU-Style M
✓	bin86	An 8086 Assembler and Linker
✓	bison	The GNU Parser Generator
✓	cvs	Concurrent Versions System
☐	db-devel	
✓	emacs	GNU Emacs Base Package
✓	emacs-info	Info files for GNU Emacs
✓	emacs-x11	GNU Emacs: Emacs binary with X Window System Su
✓	flex	Fast Lexical Analyzer Generator
✓	gcc	The GNU C Compiler and Support Files
✓	gcc-c++	The GNU C++ Compiler
✓	gcc-fortran	The GNU Fortran Compiler and Support Files

autoconf - A GNU Tool for Automatically Configuring Source Code

GNU Autoconf is a tool for configuring source code and makefiles. Using autoconf, programmers can create portable and configurable packages, because the person building the package is allowed to specify various configuration options.

You should install autoconf if you are developing software and would

Name	Disk Usage	Used	Free	Total
/data	91%	38.4 GB	3.4 GB	41.9 GB
/	17%	1.7 GB	8.3 GB	10.0 GB
/home	3%	101.3 MB	2.9 GB	3.0 GB

Objective 4 Perform a Standard Build Process

In this objective, the program xpenguins is compiled and installed from a source archive as an example of a standard build process.

The xpenguins program can be downloaded in a tar archive with the name xpenguins-2.2.tar.gz.

Before you start the build process, you need to extract the tar archive by entering the following command:

tar xzf xpenguins-2.2.tar.gz

Some tar archives end in .bz. In this case, the archive is compressed with bzip and needs to be extracted with the options xjf.

After the archive is extracted, you need to change to the source directory which has been created by entering the following:

cd xpenguins-2.2/

In the source directory, you need to run the configure script with the following command:

./configure

The output looks like the following:

```
creating cache ./config.cache
checking for a BSD compatible install... /usr/bin/install
-c
checking whether build environment is sane... yes
checking whether make sets ${MAKE}... yes
checking for working aclocal... found
checking for working autoconf... found
checking for working automake... found
checking for working autoheader... found
checking for working makeinfo... missing
checking for gcc... gcc
checking whether the C compiler (gcc  ) works... yes
checking whether the C compiler (gcc  ) is a
cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking for a BSD compatible install... /usr/bin/install
-c
checking how to run the C preprocessor... gcc -E
checking for X... libraries /usr/X11R6/lib, headers
/usr/X11R6/include
checking for dnet_ntoa in -ldnet... no
checking for dnet_ntoa in -ldnet_stub... no
checking for gethostbyname... yes
checking for connect... yes
checking for remove... yes
checking for shmat... yes
checking for IceConnectionNumber in -lICE... yes
checking for XpmReadFileToData in -lXpm... yes
checking for XpmFree in -lXpm... yes
checking for ANSI C header files... yes
checking for unistd.h... yes
checking whether time.h and sys/time.h may both be
included... yes
checking return type of signal handlers... void
checking for select... yes
checking for strdup... yes
updating cache ./config.cache
creating ./config.status
creating Makefile
creating src/Makefile
creating themes/Makefile
...
```

In the last lines of the output you can see that the Makefiles are created.

If the configure script does not report any errors, you can start the compilation process by entering the following:

make

The output of make for xpenguins looks like the following:

```
make  all-recursive
make[1]: Entering directory `/tmp/xpenguins-2.2'
Making all in src
make[2]: Entering directory `/tmp/xpenguins-2.2/src'
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c main.c
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c
xpenguins_config.c
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c
xpenguins_core.c
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c
xpenguins_theme.c
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c
toon_associate.c
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c toon_draw.c
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c
toon_globals.c
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c toon_query.c
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c toon_set.c
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c toon_core.c
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c toon_end.c
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c toon_init.c
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c toon_root.c
gcc -DHAVE_CONFIG_H -I. -I. -I..     -I/usr/X11R6/include -g -O2
-DPKGDATADIR=\""/usr/local/share/xpenguins"\" -c toon_signal.c
gcc  -I/usr/X11R6/include -g -O2 -DPKGDATADIR=
""/usr/local/share/xpenguins"\"  -o xpenguins  main.o
xpenguins_config.o xpenguins_core.o xpenguins_theme.o
toon_associate.o toon_draw.o toon_globals.o toon_query.o
toon_set.o toon_core.o toon_end.o toon_init.o toon_root.o
[...]
```

The most important part in the output of make are the compiler calls starting with:

```
gcc -DHAVE_CONFIG_H -I ...
```

When the compilation process has finished without any errors, you can install the software by entering the following (as root):

make install

Exercise 6-2 Compile Software from a Source Package

In this exercise, you compile software from a source package.

You can find this exercise in the workbook.

(End of Exercise)

Summary

Objective	Summary
1. Understand the Basics of C Programming	There are basically 2 different programming language types: ■ **Script languages.** The source code is executed by an interpreter software. ■ **Compiler languages.** The source code needs to be converted into a binary file, which can be directly executed by the CPU. The C and C++ programming languages are the most important compiler languages. C has the following basic characteristics: ■ A preprocessor processes the source code before compiling. ■ Every program has at least a main function. ■ Functions and variables need to be declared before they can be used in the code. ■ The declaration must include the definition of variable types. The name of C source files normally end in .c. The basic command to compile a source file looks like the following: **gcc *sfile*.c -o *bfile***

Objective	Summary
2. Understand the Concept of Shared Libraries	Shared libraries contain certain functions that are needed by many programs. These files are loaded when an application needs a function from the corresponding library. A shared library consists of 2 basic parts: ■ The shared object ■ The header file Some libraries are split into 2 software packages on SUSE Linux Enterprise Server 10. To run applications, you just need the base library package. To compile software, you also need the header files in the package with the extension -devel.
3. Understand the GNU Build Tool Chain	The standard build process consists of the following steps: ■ The build process must be prepared with the configure script. ■ The make command is used to compile the source code. ■ The make program is used again to install the application. The easiest way to install all necessary software packages for a build environment is to select the C/C++ Compiler and Tools selection in the YaST package manager.

Objective	Summary
4. Perform a Standard Build Process	The following are the command lines that are needed to build a software from source, shown by example of the xpenguins game: ■ **tar xzf xpenguins-2.2.tar.gz** This extracts the source archive. ■ **cd xpenguins-2.2/** ■ This changes to the source directory. ■ **./configure** This runs the configure script. ■ **make** This starts the compilation process. ■ **make install** This installs the program.

SECTION 7 Perform a Health Check and Performance Tuning

In this section, you learn to analyze performance on a SUSE Linux Enterprise Server 10 system and what you can do to prevent these bottlenecks.

Objectives

1. Find Performance Bottlenecks
2. Reduce System and Memory Load
3. Optimize the Storage System
4. Tune the Network Performance
5. Use Powertweak

Introduction

As with any system, sometimes the performance of a SUSE Linux Enterprise Server 10 system is not sufficient.

Because of the complexity of today's IT systems and infrastructure, performance bottlenecks are sometimes not easy to find. All components interact with each other, and different kinds of server types require different measures to improve system performance.

In this section, you learn about monitoring utilities that help you find the component having performance problems.

You also learn some hints for solving performance problems. Remember that the solutions for your problems need to be based on the result of your performance analysis and depend on your system type.

No matter what measures you choose, make sure that all changes are well tested before you enable them on the actual production system. Changes to the kernel parameters need to be tested very carefully.

A good source for more information about Linux performance tuning is: http://www.redbooks.ibm.com/abstracts/redp3862.html?Open.

Objective 1 Find Performance Bottlenecks

If you need to tune system performance, it is usually because the system is somehow too slow. Before you make any changes, you need to identify the bottleneck that is causing the performance problem.

Complaints from users or customers about a slow system are normally of a general character and do not provide detailed information about the cause of a problem.

Before you start to troubleshoot a system, you should ask for more information to gain a better overview of the situation. The following is a list of questions that can help you to find the performance bottleneck:

- **What kind of server is affected?** This includes information about the hardware and the purpose of the server.
- **What are the exact symptoms of a problem?** The more information you have, the more likely you are to determine the cause of a problem.
- **Does the problem occur at specific times of the day or the week?** For example, performance problems might occur in the morning when people start to work or after lunch break when people return to work.
- **When and how did the problem start?** Did the problem occur quickly or slowly over several days or months?
- **Who is experiencing the problems?** Does just one person have the problem, or is it a group of people who are using the same file server?
- **Can the problem be reproduced?** This can be very helpful when you are analyzing the system.

When you have gathered enough information, you can start to analyze the system by doing the following;

- Analyze Processes and Processor Utilization
- Analyze Memory Utilization and Performance
- Analyze Storage Performance
- Analyze Network Utilization and Performance

Analyze Processes and Processor Utilization

When you have a performance problem, you should look at the processor utilization first. If the processor is not fast enough to run all of your applications at a reasonable speed, this is the bottleneck you have to work on.

One way to measure processor utilization is the system load. The load value can be displayed with various monitoring tools such as **top** or **uptime**.

On a multiprocessing operating system like Linux, multiple processes can run virtually simultaneously. Since one processor can run only one process at a time, the Linux kernel splits the available processing time of a CPU into short slices that are assigned to the running processes.

To assign the CPU time, the kernel puts the running processes into a queue. Depending on the priority of a process and the time since it was executed last, the kernel decides which process should be executed next.

The load value is the average number of waiting processes in the process queue in a specific amount of time. Therefore programs like **top** or **uptime** display load values for the last 1, 5, and 15 minutes.

On a system with a single processor, an average load value of 1 means that the full processing capacity is used by applications and the operating system.

If the value is lower than 1, some capacity is not used. If the average value is higher than 1, the processor is not fast enough to handle all currently running processes.

On a multiprocessor system, the load value can be higher. As a rule of thumb, the load value should not be higher than the number of processors installed in the system.

A process that is started on a system does not always require CPU time. Depending on the kind of the process it is running, the CPU spends quite a lot of time to waiting for I/O processes to be finished. For example, an I/O process can be user input or data that is read from or written to the hard disk.

During these times the processes are not waiting in the kernel's process queue and do not influence the load value of a system. This means that an application can be slow, but CPU time is not the reason for it.

The following is a list of monitoring utilities that can be used to display the current CPU utilization and the average load values:

Table 7-1

Program	Description
top	Displays a sorted list of applications and the three values for the average load values in the last 1, 5, and 15 minutes. When you find that your system has a high load value, **top** can also be very helpful to find out which application is actually producing it.
uptime	**uptime** can also be used to display the system load in the last 1, 5, and 15 minutes.
mpstat	On multiprocessor systems, **mpstat** can be used to display the utilization of each installed processor.

Table 7-1 *(continued)*

Program	Description
KDE System Guard	**KDE System Guard** displays a graphical representation of the system load.

Analyze Memory Utilization and Performance

Another bottleneck for system performance can be caused by system memory. Applications have to be loaded into memory before they can be executed by the CPU. The memory is also used by the Linux kernel itself and for caching I/O operations like network or storage access.

The memory is controlled by the memory management system of the Linux kernel. Every application has to ask the kernel to allocate memory, and every application is only allowed to write into its own memory space.

There are 2 different kinds of memory available on a Linux system:

- **Physical memory.** This is memory that is actually installed in the system in the form of memory bars or chips. Access to this kind of memory is usually very fast.
- **Swap memory.** A Linux system should have access to at least one swap partition. The space on this partition is used to free parts of the physical memory by copying temporarily unused memory pages. Access to swap memory is very slow compared to physical memory.

You can view the utilization of the physical and the swap memory with the **free** program by entering the following:

free

The output looks like the following:

```
           total       used       free     shared    buffers   cached
Mem:      516204     502080      14124          0      29356   154920
-/+ buffers/cache:   317804     198400
Swap:    1036152     143320     892832
```

The output contains a headline with 3 lines of information:

- **Mem.** This line contains information about the physical memory. It contains the following details:
    - **total.** This entry displays the total amount of available physical memory in KBs. The number is lower than the installed physical memory, since the kernel itself uses a small part of the memory.
    - **used.** This entry displays the amount of memory that is used for applications cached data.
    - **free.** This entry displays the memory that is not used and available at the moment.
    - **Shared/buffers/cached.** These columns display more detailed information about how the memory is used.
- **-/+ buffers/cache.** Some of the memory on a Linux system is used to cache data for applications or devices. Parts of this memory can be freed when it is needed for other purposes.

  The free column displays the buffer adjusted line, which shows the memory that would be used and available if the buffer and the cache were freed.
- **Swap.** This line shows informations about the utilization of the swap memory. The information includes the amount of total, used, and free available memory.

As accessing the hard disk is much slower than accessing physical memory, the performance of the whole system is affected when a lot of swap space has to be used.

Usually this happens when there is not enough physical memory to perform the desired functionality of a system. It can also happen if an application requests much more memory than it actually needs.

One reason for this could be an application crash, but it also happens during normal operation, when the implementation of a program is faulty. In this case, the application has a *memory leak*.

You can use the **top** command to find programs that use a lot of memory. By default, **top** sorts the process list by CPU utilization. By typing **F**, **n**, and then pressing the **Enter** key, you can change the sorting column memory utilization. This way the top memory consumers can be found at the top of the list.

If a lot of used swap memory is displayed in **free**, this can indicate a performance bottleneck caused by a lack of physical memory. But this is not always the case. Sometimes a lot of memory is copied to the swap partition but is never touched again. The performance of the system is only affected when the swap memory is actually accessed.

You can use the command **vmstat** to display the activity of swap memory, as in the following:

vmstat 1

The option 1 lets vmstat repeat its output every second. This way the usage of swap memory can be displayed over a period of time. You can terminate the program pressing **Ctrl+C**.

The output of vmstat looks like the following:

```
procs  --------memory---------- -swap- --io-- -system- ----cpu----
 r  b  swpd  free  buff  cache  si  so  bi bo   in  cs us sy id wa
 0  0     4  6728 34464 244744   0   0 447 42 1216 384 15  3 74  7
 0  0     4  6728 34464 244744   0   0   0  0 1186 222  1  1 98  0
 0  0     4  6760 34464 244744   0   0   0  0 1282 299  3  0 97  0
 0  0     4  6696 34532 244744   0   0   0 68 1139 147  1  1 97  1
 0  0     4  6696 34532 244744   0   0   0  0 1105 123  0  0 98  0
 0  0     4  6696 34532 244744   0   0   0  0 1117 131  0  0 98  0
```

The output in the columns **si** and **so** are of interest in this case. **si** stands for swap in, which means that data is transfered to the main memory from the swap space. **so** stands for swap out, which means that data is transfered to the swap space from the main memory. In the example above, there is no activity for the swap space.

The first line of the output displays the average values since the system was started. The lines that follow show the average values since the last output.

The following output of vmstat is captured on a different system which ran out of memory and shows a lot of activity in swap memory:

```
procs  --------memory--------   -- -swap- --io-- -system- -------cpu------
 r  b    swpd free  buff  cache  si   so   bi   bo    in    cs  us sy id wa
 0  3  167880  608  4592  93400  340  188 2588  196  1223 1315   7  3  0 90
 1  3  169316 1072  4044  90352  300 1768 5968 1868  1233 1222  36  5  0 59
 1  2  170268 2520  4088  89416  288 1104 1388 1224  1260  442  23  2  0 75
 0  3  170652 1484  4020  90136  364  668 1844  808  1260 1142  12  3  0 85
 0  4  171380 1848  3544  92424  100  868 4400  940  2491 2458  11  8  0 81
 0  5  171576 1352  3504  91984  552  388 1592  388  1248 1195  15  3  0 82
```

In this example, there is much more activity in the **si** and **so** columns than before. The number displayed represents the amount of memory that is copied to or from swap memory.

A system that shows a constant vmstat output like this has a performance bottleneck caused by a lack of physical memory.

The following are commands and an application you can use to display memory utilization:

Table 7-2

Program	Description
free	Displays the current utilization of the physical and swap memory.
vmstat	Monitors the activity of swap memory and can also be used to display other system parameters.
KDE System Guard	Offers the capability to display memory usage. Choose the signal plotter visualization to follow the memory usage over a period of time.

Analyze Storage Performance

The performance of the storage system can be an issue, especially on systems that face heavy hard disk utilization like ftp, web, or other kinds of file servers.

Before you analyze the hard disk performance and utilization, you should make sure that you don't have any problems with a too-high system load or a lack of physical memory.

Systems with disk performance problems, usually show a low network and CPU utilization but a high activity of the installed disks, which is not caused by memory paging or swapping.

In this case, you can use the command vmstat to display the activity of the disk subsystem. You start vmstat by entering the following:

vmstat 1

The program should be started on the system when the performance problem occurs. The following is the output of a system with almost no disk operations:

```
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id  wa
 0  0      4   6728  34464 244744    0    0   447    42 1216   384 15  3 74   7
 0  0      4   6728  34464 244744    0    0     0     0 1186   222  1  1 98   0
 0  0      4   6760  34464 244744    0    0     0     0 1282   299  3  0 97   0
 0  0      4   6696  34532 244744    0    0     0    68 1139   147  1  1 97   1
 0  0      4   6696  34532 244744    0    0     0     0 1105   123  0  0 100  0
 0  0      4   6696  34532 244744    0    0     0     0 1117   131  0  0 100  0
```

In this example, the columns of interest are **bi** and **bo**. They display the number of blocks that are read from (**bi**) or written to (**bo**) the disk subsystem.

The following shows a system with a high utilization of the disk subsystem:

```
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 1  2     52   5680   6100 221688    0    0     0 36160 1273  1655 42 58  0  0
 0  3    304   6896   1232 225672    0  256     4 22160 1586  1127 31 40  0 28
 1  2    304   5936   1252 226540    0    0     0 28400 1487   460 15 23  0 62
 1  0    304   7792   1276 224404    0    0     0 43328 1342   408 20 29  0 51
 1  2    304   6256   1624 224648    0    0     0 88260 1205   439 24 42  0 35
 0  2    476   6648   1672 224112    0  172     4 45452 1149  8015 29 54  0 17
 0  2    476   7672   1720 223184    0    0     8 36940 1168  8310 23 44  0 33
```

As you can see in this column, the system has to deal with a lot of writing activity to the disk subsystem.

However, a lot of data read from or written to the disk does not necessarily mean that the disk subsystem is too slow. Depending on the available disk types and the disk configuration, a disk load that totally blocks one system can be easily handled by another system.

A performance problem that is caused by the disk subsystem usually occurs when a process has to wait for data being delivered from or written to the disk.

You can use the command **iostat** to determine the average time a program has to wait for data from the disk.

The **iostat** command is not part of the SUSE Linux Enterprise Server 10 default installation. You need to install the package **sysstat** to use it.

The following command displays information about the disk device /dev/hda:

iostat -x 1 /dev/hda

The option **-x** enables the output of some additional information. 1 sets the interval in which **iostat** repeats its output to 1 second. The device name specifies the disk that should be monitored. If no disk is specified on the command line, all disks that are used by the system are monitored.

The output of **iostat** looks like the following:

```
avg-cpu:  %user   %nice    %sys %iowait   %idle
           8.08    0.04    1.73    1.70   88.45

Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
hda         3.18  17.90  3.37  1.32  146.73  153.78    73.36    76.89    64.11
0.25   53.50   4.57   2.14

avg-cpu:  %user   %nice    %sys %iowait   %idle
           4.90    0.00    0.98    0.00   94.12

Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
hda         0.00   0.00  0.00  0.00    0.00    0.00     0.00     0.00     0.00
0.00    0.00   0.00   0.00

avg-cpu:  %user   %nice    %sys %iowait   %idle
           5.05    0.00    0.00    0.00   94.95

Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
hda         0.00   0.00  0.00  0.00    0.00    0.00     0.00     0.00     0.00
0.00    0.00   0.00   0.00
```

Every output contains two blocks of information. The first block displays information of the CPU utilization, like **top** or **uptime**. The second block shows the information about the requested disk device.

The first output represents the average values since the system was started. All following lines show the average values since the last update period.

The block that displays the device information shows first some details about the amount of data that is read from or written to the device. To find out if the disk subsystem has a performance bottleneck, focus on the following 2 columns:

- **await.** This column displays the average time in milliseconds an application has to wait till its I/O request is performed.
- **svctm.** This column displays the average time in milliseconds that an I/O request needs to be performed.

As you can see in the output above, the concerned system is not really busy. The average **await** time since the system was booted is 53.50 milliseconds and the average **svctm** time is 2.14 milliseconds.

As you can see in the following lines, the current disk utilization is even far below the average with **await** and **svctm** times of 0 milliseconds.

Compare this with the following output of a system with a higher I/O load:

```
avg-cpu:  %user   %nice    %sys %iowait   %idle
          26,00    0,00   45,00   29,00    0,00

Device:    rrqm/s wrqm/s    r/s   w/s  rsec/s  wsec/s    rkB/s       wkB/s
avgrq-sz avgqu-sz   await  svctm  %util
hda          0,00 9198,00  4,00 39,00   32,00 73872,00   16,00  36936,00
1718,70   103,83 1430,33  23,28 100,10

avg-cpu:  %user   %nice    %sys %iowait   %idle
          20,79    0,00   39,60   39,60    0,00

Device:   rrqm/s wrqm/s    r/s   w/s   rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
hda        0,00 9105,94  0,00 44,55    0,00 73140,59   0,00 36570,30  1641,60
99,97 2441,89  22,24  99,11

avg-cpu:  %user   %nice    %sys %iowait   %idle
          26,26    0,00   45,45   28,28    0,00

Device:   rrqm/s wrqm/s    r/s   w/s   rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
hda        0,00 10313,13 0,00 41,41    0,00 82828,28   0,00 41414,14  2000,00
93,90 2529,10  24,41 101,11

avg-cpu:  %user   %nice    %sys %iowait   %idle
          24,00    0,00   48,00   28,00    0,00

Device:   rrqm/s wrqm/s    r/s   w/s   rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
hda        0,00 9293,00  0,00 41,00    0,00 74640,00   0,00 37320,00  1820,49
92,70 2447,00  24,41 100,10
```

As you can see, the average **await** time on this system far beyond 2000 milliseconds, and the **svctm** time is much higher than before. Such a system cannot fulfill the requested I/O operation at an adequate speed.

The following is an overview of commands that you can use to analyze disk utilization:

Table 7-3

Command	Description
vmstat	Monitors the amount of data that is read from or written to disk.
iostat	Displays how long I/O requests from applications take.

Analyze Network Utilization and Performance

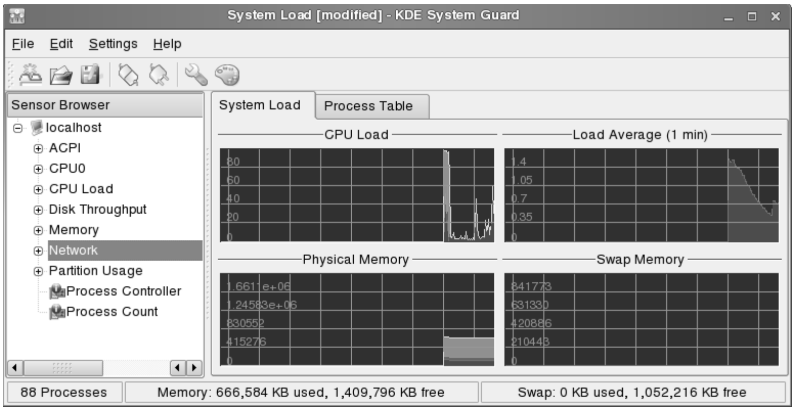
On server systems, the network connection can be a performance bottleneck. There are many different parameters that can interfere with the network connection.

You can monitor these parameters with the KDE System Guard. It can be started from the command line by typing **ksysguard**.

Although the KDE System Guard is a KDE application, it can also be used with the Gnome Desktop. The application is included in the package **kdebase3**.

The following is a screenshot of the KDE System Guard:

Figure 7-1



88 Processes | Memory: 666,584 KB used, 1,409,796 KB free | Swap: 0 KB used, 1,052,216 KB free

On the left side of the window, you can browse the available monitoring sensors. Browse to **Network > Interfaces >** ***Interface_you_want_to_monitor***.

Two different blocks of sensors are available:

- **Receiver.** These sensors display information about the received network data.
- **Transmitter.** These sensors display information about the sent network data.

The following describes some of the available sensors you can use to analyze network problems:

Table 7-4

Sensor	Description
Data/Packets	The amount of data or packets sent or received by the interface. If performance problems occur during a high network load, the network connection or type might be too slow for the purpose of the server.
Collisions	This sensor is only available for the transmitter. Collisions usually occur more frequently when too many hosts share the same Ethernet domain (such as hosts that are connected with a hub instead of a switch). Too many collisions can have a negative impact on the overall network performance.

Table 7-4 *(continued)*

Sensor	Description
Dropped Packets	This sensor displays the number of packets that are either dropped when they are received by the host or by other network components like routers on their way to the destination. Too many dropped packets can have a bad influence on the network performance. The following are some reasons for dropped packets: ■ Network components are running at a different speed. For example, the server runs at 100 Mbps, but the router at only 10 Mbps. ■ The network or system load of a server is too high to handle all received network packets properly. ■ A network component runs with a misconfigured packet filter that drops network packets.
Errors	An error occurs when a packet is transmitted but the content of the packet is corrupted. This can be caused by a bad physical connection or faulty network adapters.

There is also protocol specific information under **Network > Sockets**.

Besides problems that are caused by the network or network setup itself, some network services can interfere with the overall system performance. These network services might not even be running on the same host that actually experiences performance problems.

The following are 3 examples of this:

- **DNS.** Many applications or services rely on the name resolution of the DNS system. If a DNS server is not working properly, the application is waiting for the response, which slows down its operation.
- **Proxy.** Applications that connect to a service using a proxy server suffer from bad performance of this system.
- **NFS.** Applications or services that access data that is mounted using NFS can be blocked completely if the NFS service is not available.

The following are tools that you can use to monitor the network:

Table 7-5

Program	Description
KDE System Guard	Displays network utilization and different kinds of transmission errors.
Traffic-vis	Analyzes network connections to specific hosts. You need to install the package traffic-vis in order to use this tool
ip	Displays the status of an interface as well as transmission errors.

Exercise 7-1 Analyze System Performance

In this exercise, you analyze system performance.

You can find this exercise in the workbook.

(End of Exercise)

Objective 2 Reduce System and Memory Load

If you have determined that your performance problem is caused by a high system load, you do the following to reduce the load:

- Analyze CPU-Intensive Applications
- Run Only Required Software
- Keep Your Software Up to Date
- Optimize Swap Partitions
- Change Hardware Components

Analyze CPU-Intensive Applications

A high system and memory load is often caused by single application. You can use the **top** utility to find out which process uses the most resources on your system.

Sometimes a process uses a lot of system resources because of a faulty implementation. Usually you can determine this by restarting the process. If the process does not use the same amount of resources after it has been restarted, a likely cause is a faulty implementation.

In this case, you should try to get more information about the issue by searching the Internet and the web site of the vendor or the OpenSource project.

If the process starts to utilize the same amount of system resources after it has been restarted, the system is probably not fast enough to run the process. Refer to “Run Only Required Software” below for details on how to solve this issue.

Run Only Required Software

The easiest but most effective way to reduce the system load is to run only the software that is required to fulfill the purpose of a system. This includes the following methods:

- Run a Server System without X
- Reduce the Number of Daemon Processes

Run a Server System without X

Usually, it's not necessary to run an X-Server on a server system. Most administrative tasks including those done in YaST can be done on the text console or remotely with SSH or SUSE Linux Remote Administration.

Preventing the X-Server from being started saves memory and CPU utilization. To do so, you can switch to runlevel 3 manually by entering the following:

init 3

You can also set the default runlevel to 3 to boot the system to runlevel 3 automatically.

To change the default runlevel, you need to open the file /etc/initab with a text editor. In the file, look for a line like the following:

```
id:5:initdefault:
```

By changing 5 to 3, you can change the default runlevel from 5 (multiuser, network, graphical login) to 3 (multiuser, network).

After the change, the line looks like the following:

```
id:3:initdefault:
```

Reduce the Number of Daemon Processes

In most cases, a server offers only a few services but a lot more daemons are actually running. By reducing the number of running daemon processes, you can reduce the processor and the memory load.

To get an overview of the current service configuration, you can use the **chkconfig** command by entering the following:

chkconfig -l

The **-l** option lists all services and their configuration in each runlevel. For example, the following is the output of the Apache web server:

```
apache2      0:off  1:off  2:off  3:on   4:off  5:on   6:off
```

As you can see, apache2 is enabled for runlevels 3 and 5.

Review the list and make sure that the only services that are running are those needed in the default runlevel of your server. If you find a service that is not necessary, you can prevent it from starting up at boot time by removing its start script from the init process.

Use a command like the following to remove a service from the init process:

chkconfig apache2 off

In this example, apache2 is disabled in all runlevels. To re-enable a service, use a command like the following:

chkconfig apache2 3

In this example, apache2 is enabled in runlevel 3.

Changing the runlevel configuration does not affect the currently running instance of a service. If you don't want to reboot your system with the new configuration, you need to stop a running service by calling its **rc script** manually.

The command in the following example stops a running instance of apache2:

rcapache2 stop

Keep Your Software Up to Date

There are many reasons to keep your software up to date. Beside possible security issues caused by outdated software, up to date software can improve performance.

Implementation errors that lead to a high utilization of system resources might be fixed in a newer release. And newer, faster algorithms might be used.

However, there might be exceptions to the rule. For this reason, you should test new releases carefully before using them in a production environment.

Optimize Swap Partitions

On a system with a lot of swapping, you should usually add more main memory to enhance the performance. However, if you can't do so, optimizing the swap partitions can help.

First, you should make sure that you have enough available swap space. The old rule–that you should have double the size of the physical memory as swap space–is a bit outdated but still a reasonable starting point.

The key to speeding up the swap space is to spread it over several disks. This works only on systems that have more than one installed disk.

Every swap partition has an entry in the file /etc/fstab that looks like the following:

```
/dev/hda1      swap       swap     0 0
```

You can use more than one swap partition by creating partitions and adding these to /etc/fstab, as in the following:

```
/dev/hda1       swap       swap   pri=1   0 0
/dev/hdb1       swap       swap   pri=1   0 0
/dev/hdc1       swap       swap   pri=1   0 0
```

In this example, 3 partitions are used on 3 different disks. The additional parameter **pri=1** assigns the same priority to all swap partitions.

With a priority 1 assigned to all swap partitions, the kernel can use the partitions in parallel. This leads to a higher overall performance of swapping operations.

The drives that hold swap partitions should run at the same speed.

Change Hardware Components

If the above methods to reduce the system load do not lead to a lower resource utilization, you should consider upgrading the following hardware:

- Upgrade the CPU
- Upgrade the Memory

Upgrade the CPU

If your system shows a high system load but all other parameters such as memory, network and storage load or utilization are not significantly high, you should consider upgrading the CPU.

However, you need to consider the following before upgrading the CPU:

- Are there significantly faster CPUs available for the type of system you are using (socket type, BIOS support)?

- Are the rest of the system components fast enough for the new CPU? Otherwise, you could work on one bottleneck and create a new one.
- Is the system going to be replaced in the near future?
- Are other, faster systems available in your organization that could be used instead of the current system?

Depending on the answers to these questions, you might decide to replace the whole system instead of just the CPU. In some cases, this might be even more economical in the long run than just a CPU upgrade.

Upgrade the Memory

Upgrading the memory usually means installing more physical memory. The first question you might ask is how much additional memory you should install.

A way to answer this question is to look at the amount of swap space that is used by the system when the performance problems occur. Adding double the amount of used swap space might be a good starting point.

However, you should also compare the cost of a memory upgrade with the cost of installing a new system.

Remember that if you add additional physical memory, you should also add additional swap space. However, in most cases, more than 1 GB of swap space does not increase performance significantly.

Exercise 7-2 Reduce Resource Utilization

In this exercise, you learn how to reduce the resource utilization of a SUSE Linux Enterprise Server 10 system.

You can find this exercise in the workbook.

(End of Exercise)

Objective 3 Optimize the Storage System

There are many different ways to optimize the performance of your storage systems, including the following:

- Configure IDE Drives with hdparm
- Tune Kernel Parameters
- Tune File System Access
- Change Hardware Components

Configure IDE Drives with hdparm

You can use the tool **hdparm** to tune some settings of IDE hard drives. Entering the following command displays the current settings of a drive:

hdparm -i /dev/hda

In this example, the settings of the device hda are listed.

The most important setting you can change with hdparm is DMA (direct memory access). With DMA, data from a disk can be written directly to the main memory of a system without CPU utilization. This enhances performance in 2 ways:

- The transfer itself is much faster than with disabled DMA.
- The CPU is not utilized and can be used for other tasks.

By default, DMA should be enabled for IDE hard disks. However, if you experience a weak disk performance, you should check the settings. DMA can also be enabled for CD/DVD drives, which increases performance, especially for large data transfers.

You can use following command to check the current status of the DMA configuration:

hdparm -d /dev/hda

In this example, the DMA settings for the device hda are checked, with an output similar to the following:

```
/dev/hda:
 using_dma    =  1 (on)
```

In this example, DMA is enabled for the device hda; otherwise, the variable **using_dma** would have the value 0.

You can enable DMA with a command such as the following:

hdparm -d 1 /dev/hda

In this example, DMA is enabled for the device hda.

With hdparm, you can also use command line options that affect a drive's performance. The following lists the most important options:

Table 7-6

Parameter	Description
-c 1	Enables 32-bit transfers of disk data over the PCI bus.
-u1	A setting of 1 permits the driver to unmask other interrupts during processing of a disk interrupt, which greatly improves Linux's responsiveness and eliminates serial port overrun errors.
-X *value*	Configures the drive to use a specific transfer mode.
-A 1	Enables read-ahead, which increases performance when dealing with large, sequential file operations.

Before you change any settings with hdparm, you should make sure that important files on your system are saved and backed up. Improper settings can lead to system crashes or data loss. For more information, see man hdparm.

hdparm also provides an option to measure the transfer performance of a hard disk, as in the following command for the device hda:

hdparm -t /dev/hda

The output for this command might look like the following:

```
/dev/hda:
 Timing buffered disk reads:  156 MB in  3.01 seconds =
51.75 MB/sec
```

In this example, the disk offers a sequential transfer rate of 51.75 Mbps. To achieve valid results, you should repeat the test several time and compare the results. In general, the test should be run at a low system and storage load.

All changes that are made with hdparm are active only until the next reboot. To make sure hdparm commands are executed every time the system boots, you can add them to the file /etc/init.d/boot.local.

Tune Kernel Parameters

The components of the Linux kernel that are responsible for hard disk access offer some parameters that can be changed at runtime.

None of these parameters are saved permanently. If you want to set them every time the system starts up, you can enter a command to set a parameter in the file /etc/init.d/boot.local.

Tunable parameters let you do the following:

- Tune the IO Scheduler
- Change the Read-Ahead Parameter
- Change the Swappiness Parameter

Tune the IO Scheduler

Because Linux is a multitasking operating system, more than one process at a time might need to access the hard disk.

For this reason, the Linux kernel contains a component called the I/O Scheduler. This scheduler collects requests from the processes and hands them over to the hardware driver that is responsible for the drive.

The SUSE Linux Enterprise Server 10 I/O Scheduler has one parameter that you can used to tune the I/O performance. The parameter is stored in the file: /sys/block/*device*/queue/iosched/quantum

The parameter determines how many I/O requests are stored in a queue before they are handed over to the driver. By queuing the requests, the scheduler can optimize the order of the requests.

When you use this parameter, there is a tradeoff between data throughput and latency. Use the following rule:

- Lower value = Shorter latency but lower data throughput
- Higher value = Longer latency but higher data throughput

The default value for SUSE Linux Enterprise Server 10 is 4 requests.

You can set the value of the parameter with a command similar to the following:

echo 6 > /sys/block/hda/queue/iosched/quantum

When you change the value, you should always benchmark your application to measure the success of the change.

Changes to the I/O Scheduler parameters might not lead to performance enhancements on general purpose servers. However, on systems with a high disk utilization like database servers, it can be useful to experiment with theses settings.

Change the Read-Ahead Parameter

Another kernel parameter lets you determine how much data should be used for the read-ahead. Read-ahead basically means that more data from a file is read than requested by an application.

This is done because an application usually wants to read all data from a file, not just the data at the beginning. You can set the read-ahead parameter in the file /sys/block/*device*/queue/read_ahead_kb.

The value determines how much data (in KB) is read ahead from file. The default value on SUSE Linux Enterprise Server 10 is 512 KB. Larger values can lead to a better overall throughput, with the drawback of a higher latency.

You can set the value with the following command:

echo 256 > /sys/block/*device*/queue/read_ahead_kb

Change the Swappiness Parameter

The swappiness parameter affects both the memory and the I/O performance. It basically determines when a system starts to swap out data to the disk, and can be set in the file /proc/sys/vm/swappiness.

You can set the parameter value from 0 and 100. The higher the value, the more the system will swap. The default value for SUSE Linux Enterprise Server 10 is 60.

You can set the parameter with a command like the following:

echo 40 > /proc/sys/vm/swappiness

The parameter determines how much you value the page cache over program memory.

Tune File System Access

To achieve a performance advantage for an application, you can control the way the kernel accesses the file system by doing the following:

- Disable atime Update
- Implement File System Dependent Tuning Options

Disable atime Update

For every file Linux stores the following information:

- When the file was created (ctime)
- When the file was modified the last time (mtime)
- When the file was accessed the last time (atime)

To keep the atime information up to date, the kernel needs to update the atime attribute every time a file is accessed. Updating the atime means that the kernel needs to perform a write process, which causes additional load for the hard disk.

If the atime attribute is not important to you, you can mount a data partition with the noatime option.

The following shows an **fstab** entry for the partition /dev/hda2 that uses the **noatime** option:

```
/dev/hda2      /data   reiserfs    noatime       0 0
```

Implement File System Dependent Tuning Options

Beside the general disk tuning options, you can also configure the file system to

- Mount a Reiser File System with the notail Option
- Configure the Journaling Mode of Ext3

Mount a Reiser File System with the notail Option

On traditional UNIX files systems, small files or the rest of a big file (the tail) use a full block of the file system, although they don't really fill the block.

Reiserfs can store this data much more efficiently in the file system internal structure. However, this costs some performance. You can use the mount option **notail** to disable this feature. The drawback is a less space-efficient data storage.

You can use the **notail** option either with the **-o** option of mount or in the **/etc/fstab** file, as in the following:

```
/dev/hda2      /data   reiserfs    notail        0 0
```

Configure the Journaling Mode of Ext3

The ext3 file system offers journaling functionality. In journaling, every file system transaction is logged in a special area of a partition, called the *journal*. The data in the journal helps to restore a consistent file system in case of a system crash or a power failure.

The ext3 file system offers 3 journaling modes that also affect the disk performance:

- **data=journal.** If you use this mode, the data of a transaction and the file metadata are logged in the journal. This is the most secure option for data security.
- **data=ordered.** When an ext3 file system is mounted with this option, only the file metadata is stored in the journal. However, it forces the file data to be written to disk before the metadata.

  This option is a good compromise between speed and reliability, and is the default for SUSE Linux Enterprise Server 10.
- **data=writeback.** This is the fastest journaling option. Metadata is logged to the journal, but file data is not treated in a special way. However, you still have the advantages of a journaling file system when a crash or a power failure occurs.

You can use these options with the **-o** option of the mount command, or add them to the **/etc/fstab**, as in the following:

```
/dev/hda2      /data   ext3    data=writeback    0 0
```

Change Hardware Components

If all of the above mentioned options do not improve disk performance, you might need to consider upgrading your hardware.

From a performance perspective, a true SCSI hardware RAID system might be the best choice. But upgrading to a newer IDE or SCSI disk can produce some of the same results.

However, you have to compare the costs and the estimated advantages of an upgrade with the purchase of a new system. A hardware upgrade always has the risk of creating a new performance bottleneck somewhere else in the system.

Exercise 7-3 Tune an IDE Hard Drive with hdparm

In this exercise, you tune your IDE hard drive.

You can find this exercise in the workbook.

(End of Exercise)

Objective 4 Tune the Network Performance

There are several different approaches to tuning the network performance of your Linux system. Because of the nature of networks, this sometimes includes not only your system but the whole network infrastructure.

The following are 2 ways you can tune network performance:

- Change Kernel Network Parameters
- Change Your Network Environment

Change Kernel Network Parameters

The Linux kernel lets you change some network parameters during runtime. This makes sense on systems that have to deal with a lot of parallel connections (such as web servers).

The parameters can be set with the **sysctl** command. To use this command, you have to be the root user, because changing kernel parameters is not permitted for normal users.

The most important command line parameter of sysctl is **-w**. With this option, you can write a value into a kernel configuration parameter.

You can also access the kernel parameters from the proc file system, which is mounted under /proc. You change the parameters by writing them into the corresponding files in the **/proc** directory.

The following lists several sysctl commands and their effect on network performance:

Table 7-7

sysctl command	Effect
sysctl -w net.ipv4.tcp_tw_reuse=1 sysctl -w net.ipv4.tcp_tw_recycle=1	When a TCP connection has been closed, the corresponding socket stays in the TIME-WAIT status for a while. Setting these 2 parameters, enables the reuse of these sockets for new connections. On a system with many TCP connections, this can reduce the number of open connections and the utilization of system resources.
sysctl -w net.ipv4.tcp_keepalive_time=900	TCP connections are usually kept alive for a specific amount of time. After this time period, a system probes to see if the connection partner is still reachable. If not, the connection is closed and the used resources are freed. The default time for SUSE Linux Enterprise Server 10 is 900 seconds. By reducing this time, you can reduce the number of opened but unused connections.

Change Your Network Environment

Because networking involves more than one system, you should consider which changes to other hosts or your network infrastructure can improve the network performance.

The following are some suggestions for improving network performance:

- **Monitor all other system components.** Before you change your network infrastructure, you should make sure that your problem is really caused by the network connection.

  Monitor all other components carefully over a longer period of time, especially the CPU and memory utilization.

- **Limit the collision domain.** If you see a lot of collisions when you monitor your system's network interface, there are probably too many systems that share the same Ethernet collision domain.

  In this case, you should restructure your network or use switches instead of hubs.

- **Check cable quality.** If you see a lot of transmission errors when you monitor a network interface, you might have a problem with your network cable. Replace the network cable and monitor the interface again.

- **Check both sides of a connection.** If your server has connectivity problems with a specific client and all other clients are working correctly, you should check the connection from the client side.

- **Change network adapters.** In some cases, a driver for a network adapter can be faulty and cause a performance bottleneck. Try switching to an adapter from a different vendor and monitor the system to see if performance improves.
- **Upgrade to a faster network type.** If other measures do not lead to improved performance, upgrading to a faster network technology (such as Gigabit Ethernet) might help.

  However, you must make sure that the other components of your system (such as the chipset) can handle this speed.

Objective 5 Use Powertweak

The files and directories under the directory /proc/ contain a wealth of information about various aspects of the running system. This includes the files under /proc/sys/, which you can view and modify during operation to change the system settings.

SuSE Linux Enterprise Server offers a special tool for configuring kernel and hardware parameters called *Powertweak*. This tool includes the daemon powertweakd and a graphical YaST front-end.

A significant advantage of using Powertweak to set kernel and hardware parameters is that a short description is provided for every parameter.

To start Powertweak, do the following:

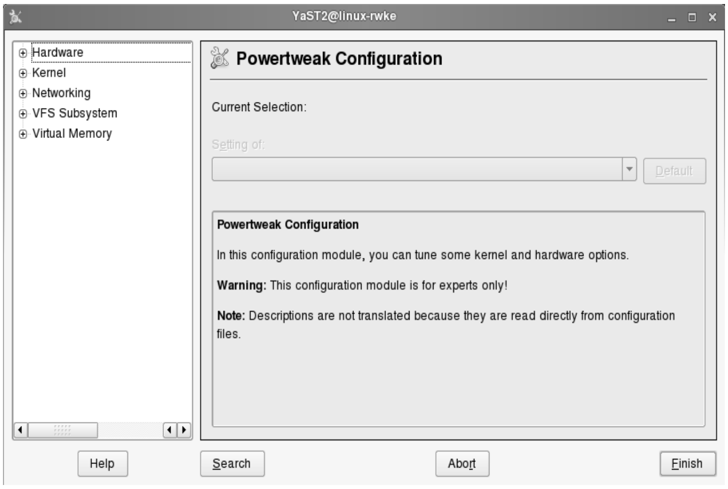
1. Start the YaST Powertweak Configuration module by doing one of the following:
   - ❑ Start **YaST** and select **System > Powertweak**.

     or
   - ❑ Open a terminal window and enter **sux -** and the root ***password***; then enter **yast2 powertweak**.

When you start the module, the powertweak packages need to be installed. By selecting the check box, **Install powertweak-extra Package** you can install an extra powertweak module, which gives more configuration options.

1. The following is a screenshot of the Powertweak module:

Figure 7-2



Powertweak Configuration

In this configuration module, you can tune some kernel and hardware options.

Warning: This configuration module is for experts only!

Note: Descriptions are not translated because they are read directly from configuration files.

2. To find a parameter, do one of the following:
   - ❑ From the left frame, expand a category and subcategories until you find the parameter you want to change; then select the ***setting***.

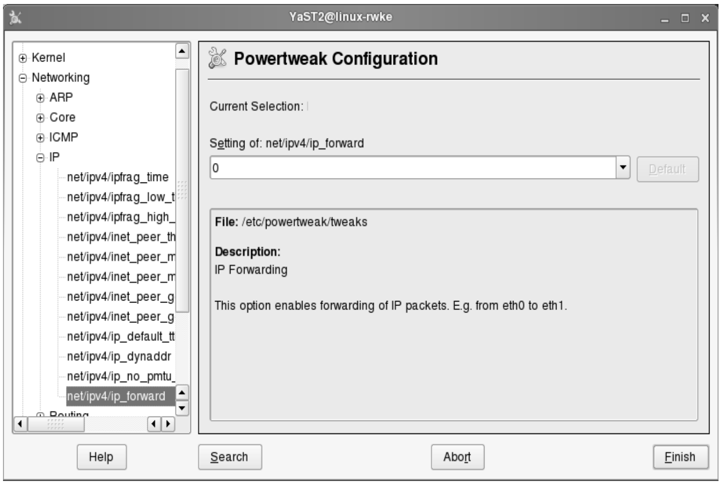
     or

   - ❑ Select **Search** and enter a ***keyword***; then select **OK**.

   Once you select a parameter, information appears in the right frame.

For example, if you find and select **Networking > IP > net/ipv4/ip_forward**, the following appears:

Figure 7-3



Powertweak Configuration

Current Selection:

Setting of: net/ipv4/ip_forward

0

File: /etc/powertweak/tweaks

Description:
IP Forwarding

This option enables forwarding of IP packets. E.g. from eth0 to eth1.

3. From the right frame, read the information (file, possible values, default value, and description); then select or enter the ***setting***.

   For example, you can activate routing by entering **1**.

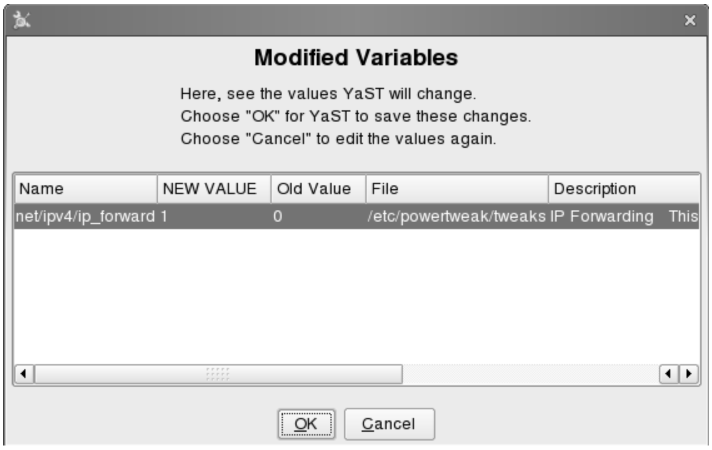
4. (Optional) Find and configure other settings.

   If you want to return a setting to its default value, select **Default**.

5. When you finish, select **Finish**.

The following appears:

Figure 7-4



Modified Variables

Here, see the values YaST will change.
Choose "OK" for YaST to save these changes.
Choose "Cancel" to edit the values again.

Name	NEW VALUE	Old Value	File	Description
net/ipv4/ip_forward	1	0	/etc/powertweak/tweaks	IP Forwarding This

This dialog lists a summary of all the changes you have indicated that you want made.

6. When you finish reviewing the list, save the changes by selecting **OK**.

   The changes are saved and activated by SuSEConfig.

When you have modified parameters with Powertweak, the Powertweak daemon (powertweakd) is configured to start in runlevel 3 and 5. The daemon restores the parameters when the system is rebooted.

Exercise 7-4 Use Powertweak

In this exercise, you use Powertweak to adjust the tcp keepalive time.

You can find this exercise in the workbook.

(End of Exercise)

Summary

Objective	Summary
1. Find Performance Bottlenecks	To find performance bottlenecks, you should monitor the following components of your system: ■ **CPU.** The value of the CPU load is measured by the average number of process that are waiting to be executed. The load can be displayed with **uptime** or **top**. top can also be used to display the processes that cause the highest CPU utilization. ■ **Memory.** A lack of physical memory is a very common performance bottleneck. When the system needs to page out memory pages to swap memory, the overall system performance is affected. You can display the paging and swapping activities with the tool **vmstat**. ■ **Storage System.** A good indicator for the storage load of a system is the time that an application needs to wait for an I/O request and the amount of time an average I/O request takes. Both values can be displayed with the tool **iostat**.
1. Find Performance Bottlenecks (continued)	■ **Network components.** KDE System Guard displays various parameters of network utilization such as packets, errors, or collisions.

Objective	Summary
2. Reduce System and Memory Load	To reduce the system and memory load, you can do the following: ■ Determine which processes utilize most of the processing power. Determine whether this is a failure or part of normal operation. ■ Run only software that is required to fulfill the purpose of the system. ■ Keep your software up to date. ■ Optimize swap memory by spreading it over multiple disks. ■ Upgrade the CPU and the physical memory.
3. Optimize the Storage System	To enhance the performance of the storage system, you can do the following: ■ Use **hdparm** to ensure an optimal configuration of your hard disks. ■ Set kernel parameters to optimize disk access. ■ Tune access to the file systems on your disks. ■ Exchange slow components of your storage system.

Objective	Summary
4. Tune the Network Performance	■ Adapt the network parameters of the Linux kernel for your needs. ■ Reconfigure your network environment. This includes the following: ■ Reduce the collision domain of Ethernet networks. ■ Check the physical quality of the connection (such as cables and plugs) ■ Check both sides of a faulty network connection. ■ Replace or upgrade your network equipment.
5. Use Powertweak	YaST offers a module for Powertweak. Powertweak allows you to make changes to kernel parameters and restores these changes when the system is rebooted.

SECTION 8 Manage Hardware

In this section, you learn how SUSE Linux Enterprise Server 10 handles hardware and device drivers. You also learn how to add and replace certain types of hardware.

Objectives

1. Describe the Differences between Devices and Interfaces
2. Describe how Device Drivers Work
3. Describe how Device Drivers Are Loaded
4. Manage Kernel Modules Manually
5. Use the hwup Command
6. Describe the sysfs File System
7. Describe how udev Works
8. Obtain Hardware Configuration Information from YaST

Introduction

Although most hardware devices can be configured with YaST or are even automatically detected when plugged into the system, it is sometimes helpful to understand how things work in the background.

In the this section, you are introduced to the SUSE Linux Enterprise Server 10 hardware management and how device drivers are loaded.

Objective 1 Describe the Differences between Devices and Interfaces

This objective uses the terms "device" and "interface." These terms are often confused, not only by users and administrators but also by developers of operating systems and related tools.

This course uses the following definitions for device and interface:

- **Device.** A device is a real, physical piece of hardware. This can be a PCI network card, an AGP graphic adapter, a USB printer, or any kind of hardware that you can hold, feel, or break if you want to.
- **Interface.** An interface is a software component associated with a device. To use a physical piece of hardware, it needs to be accessed by a software interface.

  A device can have more than one interface.

Interfaces are usually created by a driver. In Linux, a driver is usually a software module that can be loaded into the Linux kernel. Therefore, a driver can be seen as the glue between a device and its interfaces.

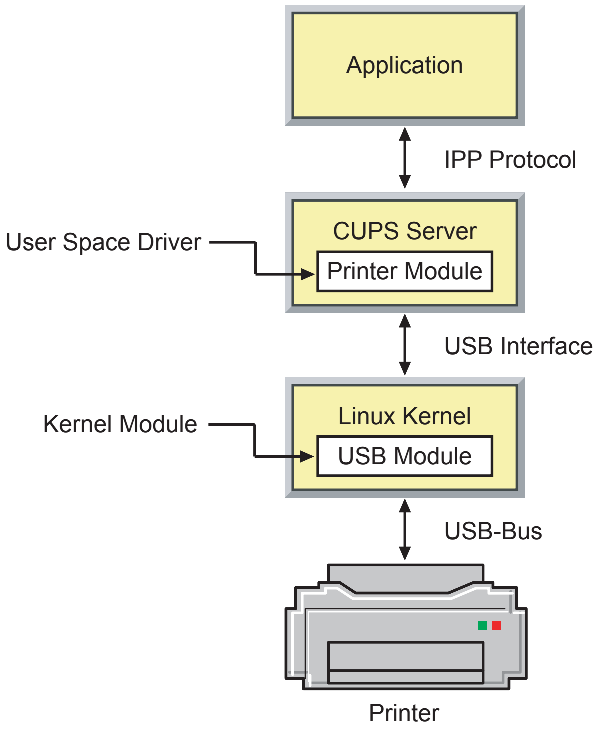
Objective 2 Describe how Device Drivers Work

As described before, device drivers access and use a device. There are 2 basic kinds of device drivers:

- **Kernel modules.** The functionality of the Linux kernel can be extended by kernel modules. These modules can be loaded and removed during runtime. They allow the kernel to provide access to hardware.
- **User space drivers.** Some hardware needs additional drivers that work in user space. Examples of this kind of hardware are printers or scanners.

The following illustrates the roles of kernel and user space drivers:

Figure 8-1



While the handling of user space drivers depends on the framework they are used in, you can manage kernel modules with the following commands:

- **lsmod.** This command lists all loaded kernel modules. For example:

  ```
  lsmod
  ```

- **modprobe.** This command loads kernel modules. Because kernel modules can depend on each other, modprobe automatically resolves these dependencies and loads all required modules. For example:

  ```
  modprobe usb-storage
  ```

  In this example, modprobe loads the usb-storage module which is needed to access storage devices connected with the USB bus.

  Because this module requires other USB modules, modprobe also loads these modules.

- **rmmod.** This command removes loaded kernel modules. For example:

  ```
  rmmod usb-storage
  ```

  Only modules that are not needed can be removed. In this example, the USB device first has to be disconnected before the usb-storage module can be removed.

Kernel modules are files that are stored in the directory /lib/modules/*kernel-version*/.

Because modules normally work only with the kernel version they are built for, a new directory is created for every kernel update you install.

Modules are stored in several subdirectories with a filename extension of **.ko** for kernel object. When loading a module with modprobe, you can leave out the extension and use just the module name.

Objective 3 Describe how Device Drivers Are Loaded

Because it would be very inconvenient to load all kernel modules manually after every system start, there are several methods to perform this task automatically.

The following is an overview of how device drivers are loaded in SUSE Linux Enterprise Server 10:

- **initrd.** Important device drivers that are necessary to access the root partition are loaded from initrd. initrd is a special file that is loaded into memory by the boot loader. Examples of such modules are the SCSI host controller and file system drivers.
- **initscripts.** Some initscripts are dedicated to loading and setting up hardware devices, such as the ALSA sound script for sound cards.
- **udev.** udev also loads kernel modules with the hwup command.
- **X Server.** Although the graphics card drivers are not kernel modules, X Server loads special drivers to enable hardware 3D support.
- **manually.** You can always load kernel modules on the command line or in scripts with the modprobe command or with the hwup or hwdown commands.

Objective 4 Manage Kernel Modules Manually

Although SUSE Linux Enterprise Server 10 initializes most hardware devices automatically, it can be very helpful to know how kernel modules are managed manually.

To manage kernel modules, you need to understand the following:

- Kernel Module Basics
- Manage Modules from the Command Line
- modprobe Configuration File (/etc/modprobe.conf)

For the latest kernel documentation, see /usr/src/linux/Documentation.

Kernel Module Basics

The kernel that is installed in the directory /boot/ is configured for a wide range of hardware. It is not necessary to compile a custom kernel, unless you want to test experimental features and drivers.

Drivers and features of the Linux kernel can either be compiled into the kernel or be loaded as kernel modules. These modules can be loaded later, while the system is running, without having to reboot the computer.

This is especially true of kernel modules that are not required to boot the system. By loading them as components after the system boots, the kernel can be kept relatively small.

The kernel modules are located in the directory /lib/modules/*version*/kernel/.

For example, the modules for the 2.6 kernel can be found in the following directory:

/lib/modules/2.6.16-0.12-default/kernel/

Manage Modules from the Command Line

The following are commands you can use from a command line when working with modules:

- **lsmod.** This command lists the currently loaded modules in the kernel.

  The following is an example:

```
DA50:~ # lsmod
Module                  Size  Used by
quota_v2               12928  2
edd                    13720  0
joydev                 14528  0
sg                     41632  0
st                     44956  0
sr_mod                 21028  0
ide_cd                 42628  0
cdrom                  42780  2 sr_mod,ide_cd
nvram                  13448  0
usbserial              35952  0
parport_pc             41024  1
lp                     15364  0
parport                44232  2 parport_pc,lp
ipv6                  276348  44
uhci_hcd               35728  0
intel_agp              22812  1
agpgart                36140  1 intel_agp
evdev                  13952  0
usbcore               116572  4 usbserial,uhci_hcd
```

  The list includes information about the module name, size of the module, how often the module is used, and by which other modules use it.

- **insmod *module*.** This command loads the indicated *module* into the kernel.

  The module must be stored in the directory /lib/modules/*version_number*/. However, it is recommended to use modprobe for loading modules.

- **rmmod *module*.** This command removes the indicated *module* from the kernel. However, it can only be removed if no processes are accessing hardware connected to it or corresponding services.

  We recommend that you use **modprobe -r** for removing modules.

- **modprobe *module*.** This command loads the indicated *module* into the kernel or removes it (with option -r).

  Dependencies of other modules are taken into account when using modprobe. In addition, modprobe reads in the file /etc/modprobe.conf for any configuration settings.

  This command can only be used if the file /lib/modules/*version*/modules.dep created by the command depmod exists. This file is used to add or remove dependencies.

  The kernel daemon (Kmod since kernel version 2.2.x) ensures that modules needed in the running operation are automatically loaded using modprobe (such as accessing the CD-ROM drive).

For more detailed information, enter **man modprobe**.

- **depmod.** This command creates the file /lib/modules/*version*/modules.dep. This file contains the dependencies of individual modules on each other.

  When a module is loaded (such as with modprobe), modules.dep ensures that all modules it depends on are also loaded.

  If the file modules.dep does not exist, it is created automatically when the system starts by the start script /etc/init.d/boot. For this reason, you do not need to create the file manually.

  On SUSE Linux Enterprise Server 10, depmode also creates the file modules.aliases, which is used by hwup and modprobe to determine which driver needs to be loaded for which device. Learn more about this file in the hwup objective in this section.

- **modinfo *option module*.** This command displays information (such as license, author, and description) about the module indicated on the command line.

  The following is an example:

```
DA50:~ # modinfo isdn
license:        GPL
author:         Fritz Elfert
description:    ISDN4Linux: link layer
depends:        slhc
supported:      yes
vermagic:       2.6.5-7.21-default 586 REGPARM gcc-3.3
```

For more detailed information, enter **man modinfo**.

modprobe Configuration File (/etc/modprobe.conf)

The file /etc/modprob.conf is the configuration file for the kernel modules. For example, it contains parameters for the modules that access hardware directly.

The file plays an important role in loading modules. Various command types can be found in the file, such as the following:

- **install.** These instructions let modprobe execute commands when loading a specific module into the kernel.

  The following is an example:

```
install     eth0     /bin/true
```

- **alias.** These instructions determine which kernel module will be loaded for a specific device file.

  The following is an example:

  ```
  alias     eth0     nvnet
  ```

- **options.** These instructions are options for loading a module.

  The following is an example:

  ```
  options     ne     io=0x300 irq=5
  ```

For more detailed information, enter **man 5 modprobe.conf**.

Exercise 8-1 Manage the Linux Kernel Modules

In this exercise, you load and unload kernel modules.

You can find this exercise in the workbook.

(End of Exercise)

Objective 5 Describe the sysfs File System

sysfs is a virtual file system that is mounted under /sys. In a virtual file system, there is no physical device that holds the information. Instead, the file system is generated virtually by the kernel.

The directory and file structure under /sys, provides information about the hardware which is currently connected with a system. Under **/sys**, there are 4 main directories:

- **/sys/bus** and **/sys/devices.** These directories contain different representations of system hardware. Devices are represented here.

  For example, the following represents a digital camera connected to the USB bus:

  ```
  /sys/bus/usb/devices/1-1/
  ```

  This directory contains several files that provide information about the device. The following is a listing of the files in this directory:

  ```
  1-1:1.0               bMaxPower         manufacturer
  bcdDevice             bNumConfigurations  maxchild
  bConfigurationValue   bNumInterfaces      power
  bDeviceClass          detach_state        product
  bDeviceProtocol       devnum              serial
  bDeviceSubClass       idProduct           speed
  bmAttributes          idVendor            version
  ```

  For example, by reading the content from the manufacturer file, you can determine the manufacturer of the device:

  ```
  cat manufacturer
  OLYMPUS
  ```

  In this case, an Olympus digital camera is connected with the system.

- **/sys/class** and **/sys/block.** The interfaces of the devices are represented under these 2 directories.

  For example, the interface belonging to the Olympus digital camera is represented by the following directory:

  `/sys/block/sda/`

  The directory named /sda is the digital camera accessed like a SCSI hard disk.

  The following is the content of the /sda directory:

  ```
  dev     queue  removable  size
  device  range  sda1       stat
  ```

  The subdirectory /sda1 represents the interface to the first partition on the cameras memory card. For example, by reading the content of /sda1/size, you can determine the size of the partition:

  ```
  cat sda1/size
  31959
  ```

  The partition has a size of 31959 512-byte blocks, which is about 16 MB.

To connect an interface with a device, file system links are used. In the Olympus digital camera example, a link exists from the file /sys/block/sda/device to the corresponding device:

```
ll device
lrwxrwxrwx  1 root root 0 Aug 17 14:03 device ->
../../devices/pci0000:00/0000:00:1d.0/usb1/1-1/1-1:1.0/hos
t0/0:0:0:0
```

In this way, all interfaces of the system are linked with their corresponding devices.

Beside the representation in **sysfs**, there are also the device files in the **/dev** directory.

These files are needed for applications to access the interfaces of a device. The name “device file” is a bit misleading, the name “interface file” would be more suitable.

Objective 6 Describe how udev Works

Before you can use a hardware device, you need to load the appropriate driver module and set up the corresponding interface. For most devices in SUSE Linux Enterprise Server 10, this is done by udev.

To get an overview about udev, do the following:

- Understand the Purpose of udev
- Understand how udev Works
- Understand Persistent Interface Names

Understand the Purpose of udev

udev has three main purposes:

- **Create device files.** The most obvious task of udev is to create device files under **/dev** automatically when a device is connected to the system. Before, the /dev directory was populated with every device, that might appear in the system. This led to a very complex and confusing /dev directory, as most of the device files were actually not used.
- **Persistent device names.** Another advantage of udev is that it provides a mechanism for persistant device names.
- **Hotplug replacement.** In SUSE Linux Enterprise Server 10, udev also replaces the hotplug system, which was responsible for the initialization of hardware devices in previous versions. udev is now the central point for hardware initialization in SUSE Linux Enterprise Server 10.

Understand how udev Works

udev is implemented as a deamon (udevd), which is started at boot time through the script **/etc/init.d/boot.d/S03boot.udev**. udev communicates with the Linux kernel through the **uevent** interface. When the kernel sends out a uevent message that a device has been added or removed, udevd does the following, based on the udev rules:

- Initializes devices by calling hwup.
- Creates device files in **/dev**.
- Sets up network interfaces with ifup.
- Renames network interfaces, if necessary.
- Mounts storage devices which are identified as hotplug in **/etc/fstab**.
- Informs other applications through HAL (Hardware Abstraction Layer) about the new device.

To handle uevent messages, which have been issued before udevd was started, the udev start script triggers these missed events by parsing the sysfs filesystem. In previous SUSE Linux Enterprise Server versions, this part of the system initialisation was done by the coldplug script.

Everything udev does, depends on rules, defined in configuration files under **/etc/udev/rules.d/**, which are used to process a uevent.

A detailed description of udev rules is beyond the scope of this course. However, in the following you find the most important facts:

- udev rules are spread over several files, which are processed in alphabetical order. Each line in these files is a rule. Comments can be added with the # character.
- Each rule consists of multiple key value pairs. Example for a key value pair: kernel==”hda”

- There are two different key types:
    - **match keys**. These keys are used to determine, if rule should be used to process an event.
    - **assignment keys.** These keys determine what to do if a rule is processed.

  There always has to be at least one match and one assignment key in rule.
- For every uevent, all rules are processed. Processing does not stop when a matching rule is found.

You can monitor the activity of udev with the tool **udevmonitor**. When you start the tool and change the hardware configuration (e.g. plug or unplug an USB device) the udev activities are displayed on the screen. For more details, you can start the tool the option **--env**.

Understand Persistent Interface Names

The interface files in the directory **/dev** are created and assigned to the corresponding hardware device when the device is recognized and initialized by a driver. Therefore the assignment between device and interface file depends on:

- The order in which device drivers are loaded.
- The order in which devices are connected to a computer.

This can lead to situations, where it‘s not clear which device file is assigned to which device. Consider the following example:

You have two USB devices, a digital camera and a flash card reader. Both devices are accessed as storage devices through the device files **/dev/sda**, **/dev/sdb** ...

Which device is assigned to which device file, usually depends on the order in which they are plugged in. The first devices gets sda and so on.

udev can help to make this more predictable. With the help of sysfs, udev can find out, which device is connected to which interface file. The easiest solution for persistent device names would be to rename the interface files, for example from /dev/sda1 to /dev/camera.

Unfortunatly, interface files can not be renamed under Linux. The only exeption to this rule are network interfaces, which traditionally have no interface files under /dev.

Therefore udev uses a different approach. Instead of renaming an interface file, a link with a unique and persitent name is created to the assigned interface file.

By default udev is for example configured to create these links for all storage devices. For each device, a link is created in each of the following subdirectories under **/dev/disk/**:

- **by-id**. The name of the link is based on the vendor and on the name of a device.
- **by-path**. The name of the link is based on the bus position of a device.
- **by-uuid**. The name of the link is based on the serial number of a device.

The udev rules which create these links are located in the file /etc/udev/rules.d/60-persistent-storage.rules.

This means, that the association between devices and interface files still depends on the order in which the drivers are loaded or in which order devices are connected with the system.

With udev however, persistent links are created and adjusted everytime the device configuration changes.

As mentioned before, network interfaces are treated differently. They don’t have interface files and they can be directly renamed by udev.

Persistent network interfaces are configured as udev rule in the file **/etc/udev/rules.d/30-net_persistent_names.rules**

The following is an example rule:

```
SUBSYSTEM=="net", ACTION=="add", SYSFS{address}=="00:30:05:4b:98:85",
IMPORT="/sbin/rename_netiface %k eth0"
```

The matching key SYSFS{address} is used to identify a network device by it's MAC address. At the end of the rule, the name of the interface is given. In this example eth0.

In SUSE Linux Enterprise Server 9, it was possible to configure persistent network interface names in the interface configuration files under /etc/sysconfig/network. This is not supported anymore in SUSE Linux Enterprise Server 10. Interface names now have to be configured in the udev rule.

Exercise 8-2 Add a device symlink with udev

In this exercise, we will create a udev rule, that creates a symlink in /dev when a device in plugged in. To perform this exercise, you need a USB mouse.

You can find this exercise in the workbook.

(End of Exercise)

Objective 7 Use the hwup Command

The command hwup is used by udev to load driver modules and to initialize devices. In this objective, you learn how to use this command and how to interpret the corresponding configuration files.

To start a device, hwup is called with a hardware description as argument. The hardware description is a unique identifier for a specific device in the system. The following is an example hwup command line:

hwup bus-pci-0000:02:08.0

The device description consists of the following components:

- **bus.** This determines that the device is identified by the bus it is connected to.
- **pci.** This indicates that the device is connected to the PCI bus.
- **0000:02:08.0.** This is the address of the device in the PCI bus.

You can display the PCI address of a device with the **lspci** command, as in the following example:

```
...
0000:02:08.0 Ethernet controller: Intel Corp. 82801BD
PRO/100 VE (LOM) Ethernet Controller (rev 81)
...
```

As you can see, with the command **hwup bus-pci-0000:02:08.0**, the ethernet controller with the PCI adress 0000:02:08.0 would be started.

The command **hwdown <hardware_description>** can be used to stop a device. hwdown is a link to hwup and not a separate command.

hwdown unbinds a device from it's driver, but does not unload the driver module automatically.

There are two different ways how hwup gets information about devices and about the driver that needs to be loaded for a device:

- From Configuration Files
- From sysfs

From Configuration Files

When a device needs to be initialized, hwup first tries to read a device configuration from files in the directory **/etc/sysconfig/hardware/**.

In order to determine the correct configuration file, the configuration filenames follow a specific naming scheme.

The following is the filename for a PCI network adapter:

hwcfg-bus-pci-0000:02:08.0

The filename consists of the keyword **hwcfg** and the hardware description of a device.

The following lists the possible variables in a device configuration file:

Table 8-1

Variable	Description
STARTMODE	This determines when and how a device will be started: ■ **auto.** The device is automatically started at boot time or by udev when the device is connected to the system. ■ **manual.** The device *should not* be started automatically, but it *can* be started manually. ■ **off.** The device should never be started.

Table 8-1 *(continued)*

Variable	Description
MODULE	The value of this variable determines the name of the kernel module that should be loaded for the device. If multiple modules have to be loaded, you can use this variable multiple times with any suffix appended. You must then use the same suffixes for multiple MODULE_OPTIONS variables. Example: **MODULE_A=”foo”** **MODULE_B=”bar”** **MODULE_OPTIONS_A=”foo-opt”** **MODULE_OPTIONS_B=”bar-opt1=xyz”**
MODULE_OPTIONS	With this variable, options can be passed to the kernel module.
SCRIPT{UP,DOWN}_[type]	This specifies the script to be called for initialization and deconfiguration of a specific device type. This script is called if the type of the device to be initialized matches the type given in this parameter.
SCRIPT{UP,DOWN}	This specifies the script to be called for initialization and deconfiguration of the device. It will be called only if no matching type-specific scripts are configured.

The following is an example of a configuration file for a network adapter:

```
MODULE='e100'
MODULE_OPTIONS=''
STARTMODE='auto'
```

The module e100 is loaded, there are no options for this modul and the device is started automatically at boot time.

The hwup command is usually called by udev, but you can also use it manually. For example, the following command starts the network card shown in the previous:

hwup bus-pci-0000:02:08.0

The last 3 elements of the configuration filename specify the device.

You can use the command hwdown to deconfigure devices, as in the following:

hwdown bus-pci-0000:02:08.0

From sysfs

When there is no configuration file for a device under /etc/hardware, hwup uses **sysfs** to find out more about the required driver.

In the file **/sys/bus/devices/<pci_device_id>/modalias** a unique id for a device can be found. The following is the content of the file modalias for a network adapter:

```
pci:v00008086d00001039sv00001734sd00001001bc02sc00i00
```

Now udev calls the command modprobe and uses the modalias is as parameter.

The file **/lib/modules/<kernel_version>/modules.aliases** contains a list with modalias IDs and their corresponding kernel driver modules. The information about which driver module handles which modalias ID is included in the source files of the modules. The command depmod is used to extract the information from the modules and to write them into the file modules.aliases.

Therefore you can consider the file **modules.aliases** as a kind of driver database, which is used to find out, which device is handled by which driver.

When called from udev with the modalias parameter, modprobe looks for the driver in modules.aliases and load the asscociated driver.

To prevent udev/hwup/modprobe from loading a driver automatically in the described way, a driver can be entered in the file **/etc/modprobe.d/blacklist**. By default the ALSA sound drivers are for example entered here, because these drivers are loaded by the ALSA init script (alsasound).

Exercise 8-3 Explore Hardware Initialization

In this exercise, you learn how to shutdown a device with hwdown and how to start it again manually.

You can find this exercise in the workbook.

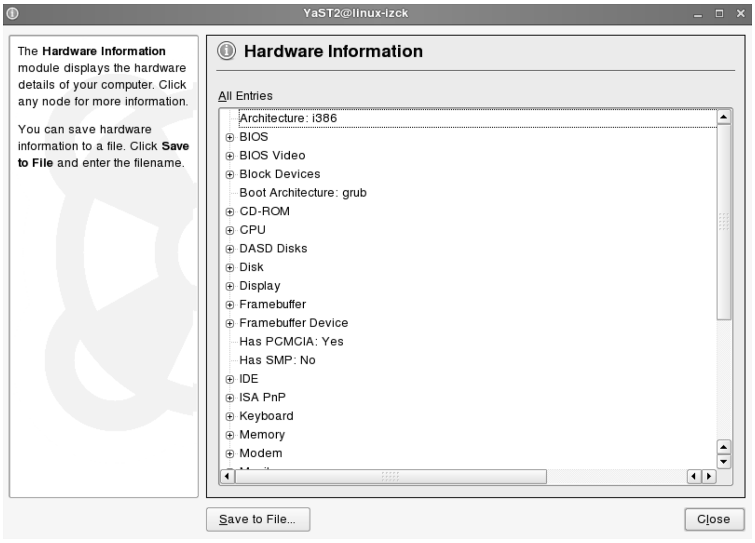
(End of Exercise)

Objective 8 Obtain Hardware Configuration Information from YaST

To obtain information about the configuration of your hardware, from the YaST Control Center select **Hardware > Hardware Information**.

After scanning the hardware for a few moments, YaST displays a dialog similar to the following that summarizes the information about the detected hardware:

Figure 8-2



The **Hardware Information** module displays the hardware details of your computer. Click any node for more information.

You can save hardware information to a file. Click **Save to File** and enter the filename.

```
Architecture: i386
⊕ BIOS
⊕ BIOS Video
⊕ Block Devices
  Boot Architecture: grub
⊕ CD-ROM
⊕ CPU
⊕ DASD Disks
⊕ Disk
⊕ Display
⊕ Framebuffer
⊕ Framebuffer Device
  Has PCMCIA: Yes
  Has SMP: No
⊕ IDE
⊕ ISA PnP
⊕ Keyboard
⊕ Memory
⊕ Modem
```

Exercise 8-4 Obtain Hardware Configuration Information

In this exercise, you learn how to obtain hardware configuration information about your computer.

You can find the exercise in the workbook.

(End of Exercise)

Summary

Objective	Summary
1. Describe the Differences between Devices and Interfaces	■ The terms device and interface are often confused. This section uses the following definitions: ■ **Device.** A device is a physical piece of hardware. ■ **Interface.** An interface is a software component that is used to access a device. ■ One device can have more than one interface. ■ An interface is created by a device driver.
2. Describe how Device Drivers Work	■ There are 2 basic kinds of device drivers: ■ **Kernel modules.** Kernel modules are loaded into the Linux kernel and extend its functionality. ■ **User space drivers.** These drivers run within user space applications. ■ Some devices require both: kernel modules and user space drivers. ■ You can use the following commands to manage kernel modules: ■ **lsmod.** Use lsmod to list loaded drivers. ■ **modprobe.** Use modprobe load kernel modules.

Objective	Summary
2. Describe how Device Drivers Work (continued)	■ **rmmod.** Use rmmod to remove loaded kernel modules. ■ The kernel modules are files that are stored in the directory /lib/modules/*kernel-version*/.
3. Describe how Device Drivers Are Loaded	In a SUSE Linux Enterprise Server 10 system, kernel modules are loaded in the following ways: ■ From initrd ■ By initscripts ■ By udev ■ By the X Server ■ Manually by the user root
4. Manage Kernel Modules Manually	■ The hwup command is used to start preconfigured devices. ■ The device configuration files are stored in the directory /etc/sysconfig/hardware/. ■ The filename of the configuration file contains a unique identifier for the corresponding device. In the configuration file, the following variables can be used: ■ STARTMODE ■ MODULE ■ MODULE_OPTIONS ■ SCRIPT{UP,DOWN}_[type] ■ SCRIPT{UP,DOWN}

Objective	Summary
5. Describe the sysfs File System	■ The sysfs file system provides a representation of all devices and interfaces of a system. ■ Devices are represented in the directories: /sys/bus and /sys/devices. ■ Interfaces are represented by the directories /sys/class and /sys/block. ■ A device and its interfaces are connected with file system links.
6. Describe how udev Works	In SUSE Linux Enterprise Server 10, udev also replaces the hotplug system, which was responsible for the initialization of hardware devices in previous versions. udev is the central point for hardware initialization in SUSE Linux Enterprise Server 10.
7. Use the hwup Command	hwup is used to load device drivers. Information about the required drivers for a device is either taken from a configuration file under /etc/sysconfig/hardware/ or from the file /lib/modules/<kernel_version>/ modules.aliases.
8. Obtain Hardware Configuration Information from YaST	To obtain information about the configuration of your hardware, select the YaST module **Hardware > Hardware Information**.

SECTION 9 Prepare for the Novell CLP 10 Practicum

In this section, you work through the following scenarios to prepare for the Novell CLP (Certified Linux Professional) practicum exam:

1. Install a Xen Environment
2. Configure a Web Server
3. Configure a Samba File Server
4. Automate System Tasks

Remember that skills from all 3 Novell CLP courses might be necessary to fulfill the required tasks.

There might be not enough time to complete all of the objectives in this section on the last day of the course. We recommend to complete the remaining parts at home.

Scenario

Digital Airlines is planning on deploying SUSE Linux in its IT infrastructure. During the first phase, SUSE Linux Enterprise Server 10 will be used on the back-end systems like file, web, and network-infrastructure servers.

As the network administrator for your Digital Airlines office, you (along with the management) have designed a migration plan which includes the following services to be migrated to SUSE Linux Enterprise Server 10:

- Intranet web server
- File and print services for Windows clients.

Both services should run on the same physical system in separat Xen domains.

You decide to start by installing and testing these services on a computer in a computer in the test lab.

Objective 1 Install a Xen Environment

To create a base for the samba and the apache system, it’s required to reinstall your physical machine and to setup 3 Xen domains. One domain0 and two domainUs. Setup your system according to the following guidelines:

- Delete the existing installation on your system and reinstall SUSE Linux Enterprise Server 10.
- During the installation, create a partition which has a enough space to hold the filesystem image files of two SUSE Linux Enterprise Server 10 Xen installations. Mount this partition under **/xen** in domain0.
- Install Xen and boot into domain0 of the Xen system.
- Create two Xen domains with YaST and install SUSE Linux Enterprise Server 10. Make sure, that the filesystem image files are stored under **/xen**.
- Make sure, that networking works in domain0 and in the two domainU systems.

Objective 2 Configure a Web Server

Your Digital Airlines office runs an internal web server which provides vital information for employees. The server hosts a general portal site and a virtual host for every department.

Because the web server needs to be migrated to SUSE Linux Enterprise Server 10, you decide to create a prototype system for the general portal site and 2 departments (accounting and marketing) in one of the Xen domains (domainU).

Set up the prototype system using the following guidelines:

- Install and configure an Apache web server that hosts the general portal site and 2 virtual hosts for the departments accounting and marketing.
- Use the Apache example pages as demo content.
- The virtual host from accounting should run under SSL, and should only be accessible for the users in the group accounting.
- Make additional entries in the file **/etc/hosts** to test the virtual host setup.
- From each department one user should be allowed to login using SSH on the server to change the content of the virtual host.

  Create the users **JNelson** and **SRife** on your system. JNelson should be responsible for the marketing department and SRife for the accounting department.
- All pages which you have to migrate end in **.htm**. Create a shell script which replaces the **.htm** with **.html**.

Objective 3 Configure a Samba File Server

As part of the SUSE Linux migration plan for your Digital Airlines office, you need to move file and print services to a Samba server running on SUSE Linux Enterprise Server 10.

You decide to test this migration for the marketing department on the other Xen domain (domainU).

Set up the Samba server using the following guidelines:

- Install the Samba server and client software.
- Configure a marketing workgroup.
- Create a UNIX group named **marketing**.
- Create 2 normal users (PSmith and JWattson) who are members of the accounting group and are included in the file **smbpasswd**.
- Create one shared folder for the group accounting.
- Export the home directories of **PSmith** and **JWattson** as personal shared folders.
- Test your shares (you can use **smbclient**).

Objective 4 Automate System Tasks

In order to make the administration of the SUSE Linux Enterprise Server 10 system as convenient as possible, certain task should be automated with shell scripts.

Do the following:

- On domain0, develop a shell script which can be used to start and stop the web server and the Samba server domains. The scripts should be easier to use as an xm command line and simply take the parameter **start** and **stop**.

  Every call of the scripts should be documented by sending a mail to the root user. **Note**: This can be done with the command **mail**. See the manpage of mail for details.

  First, develop a script for the web server domain. When this script works properly, make a copy and adjust it for the Samba server domain.

  Both scripts should be installed in the **~/bin** directory of the root user.

- On the Samba server, develop a script, that searches for Windows executables in the shared folder of the accounting department.

  Use the command **file** to determine whether a file is an windows executable or not.

  When a file is detected as Windows executable, the file should be moved to a quarantine directory in roots home directory. Additionally, a mail should be sent to the root user for every executable found. The mail should include information about the filename and the location where the file was found.

- On the web server domain, develop a backup script, that makes an incremental backup of the directory /srv/www/. Status information about the backup should be mailed to the root user when the backup has been completed.

Index