SYLLABUS

SYSTEM ADMINISTRATION

(TCS–701/TIT-701)

UNIT-I

Introduction: Duties of the Administrator, Administration tools, Overview of permissions. processes : Process status. Killing processes, process priority. Starting up and Shut down : Peripherals, Kernal loading, Console, The schedular, init and inittab file, Run-levels, Run level scripts.

Managing User Accounts: Principles, password file, Password security, Shadow file, Groups and the group file, Shells, restricted shells, user management commands, homes and permissions, default files, ofiles, locking accounts, setting passwords, Switching user, Switching group, Removing users.

UNIT-II

Managing Unix File Systems: Partitions, Swap space, Device files, Raw and Block files, Formatting disks, Making file system, Superblock, I-nodes, File system checker, Mounting file systems, Logical Volumes, Network File systems, Boot disks

Configuring the TCp/IP Networking: Kernal Configuration; Mounting the/proc File system, Installing the Binaries, Setting the Hostname, Assigning IP Addresses, Creating Subnets, Writing Hosts and networks Files, Interface Configuration for IP, ifconfig, netstat command, Checking the ARP Tables; Name service and resolver configuration.

UNIT-III

TCP/IP Firewall: Methods of Attack, What Is a Firewall? What Is IP Filtering? Setting Up Linux for Firewalling Testing a Firewell Configuration; A Sample Firewall Configuration: IP Accounting, Configurating the Kernel for IP Accounting, Configuring IP Accounting, Using IP Accounting Results.

IP Masquerade and Network and Address Translation: Side Effects and Frings Benefits, Configuring the Kernel for IP Masquerade, Configuring IP Masquerade.

UNIT-IV

The Network Information System: Getting Acquainted with NIS, NIS Versus NIS+, The Client Side of NIS, Running and NIS Server, NIS Server Security.

Network File Syetm: Preparing NFS, Mounting an NFS Volume, The NFS Daemons, The exports File.

System Backup and Recovery: Log files for system and applications; Backup schedules and methods (manual and automated).

UNIT-V

Active Directory, LDAP.

UNIT—I

INTRODUCTION AND MANAGING USER ACCOUNTS

Q.1. Explain the essential role of the system administrator. **[Important]**

Ans. Essential Duties of the System Administrator : The person who is responsible for setting up and maintaining the system is called as the system administrator. System administrators may be members of an information technology department. The duties of a system administrator are wide-ranging, and vary widely from one organization to another.

The system administrator is responsible for following things:

1. **Account provisioning :** The system administrator adds accounts for new users, removes the accounts of users that are no longer active, and handles all the account-related issues that come up in between (e.g., forgotten passwords). The process of adding and removing users can be automated, but certain administrative decisions (where to put a user's home directory, which machines to create the account on, etc.) must still be made before a new user can be added. When a user should no longer have access to the system, the user's account must be disabled. All the files owned by the account should be backed up and then disposed of so that the system does not accumulate unwanted baggage over time.
2. **Adding and removing hardware :** When new hardware is purchased or when hardware is moved from one machine to another, the system must be configured to recognize and use that hardware. Hardware-support chores can range from the simple task of adding a printer to the more complex job of adding a disk array. Now that virtualization has arrived in the enterprise computing sphere, hardware configuration can be more complicated than ever. Devices may need installation at several layers of the virtualization stack, and the system administrator may need to formulate policies that allow the hardware to be shared securely and fairly.
3. **Performing backups :** Performing backups is perhaps the most important job of the system administrator, and it is also the job that is most often ignored or sloppily done. Backups are time consuming and boring, but they are absolutely necessary. Backups can be automated and delegated to an underling, but it is still the system administrator's job to make sure that backups are executed correctly and on schedule (and that the resulting media can actually be used to restore files).
4. **Installing and upgrading software :** When new software is acquired, it must be installed and tested, often under several operating systems and on several types of hardware. Once the software is working correctly, users must be informed of its availability and location. As patches and security updates are released, they must be incorporated smoothly into the local environment. Local software and administrative scripts should be properly packaged and managed in a fashion that's compatible with the native upgrade procedures used on systems at your site. As this software evolves, new releases should be staged for testing before being deployed to the entire site.

5. **Monitoring the system :** Large installations require vigilant supervision. Don't expect users to report problems to you unless the issues are severe. Working around a problem is usually faster than taking the time to document and report it, so users often follow the path of least resistance.
   Regularly ensure that email and web services are working correctly, watch log files for early signs of trouble, make sure that local networks are properly connected, and keep an eye on the availability of system resources such as disk space. All of these chores are excellent opportunities for automation, and a variety of off-the shelf monitoring systems can help sysadmins with this task.
6. **Troubleshooting :** System failures are inevitable. It is the administrator's job to play mechanic by diagnosing problems and calling in experts if needed. Finding the problem is often harder than fixing it.
7. **Maintaining local documentation :** As a system is changed to suit an organization's needs, it begins to differ from the plain-vanilla system described by the documentation. Since the system administrator is responsible for making these customizations, it's also the sysadmin's duty to document the changes. This chore includes documenting where cables are run and how they are constructed, keeping maintenance records for all hardware, recording the status of backups, and documenting local procedures and policies.
8. **Vigilantly monitoring security :** The system administrator must implement a security policy and periodically check to be sure that the security of the system has not been violated. On low security systems, this chore might involve only a few basic checks for unauthorized access. On a high-security system, it can include an elaborate network of traps and auditing programs.
9. **Fire fighting :** Although helping users with their various problems is rarely included in a system administrator's job description, it claims a significant portion of most administrators' workdays. System administrators are bombarded with problems ranging from "It worked yesterday and now it doesn't! What did you change?" to "I spilled coffee on my keyboard! Should I pour water on it to wash it out?" In most cases, your response to these issues affects your perceived value as an administrator far more than does any actual technical skill you might possess. You can either howl at the injustice of it all, or you can delight in the fact that a single well-handled trouble ticket scores as many brownie points as five hours of midnight debugging.

Q.2. Discuss the various administration tools of Windows.

Ans. Window System Administration Tools : Administrative Tools is a folder in Control Panel that contains tools for system administrators and advanced users. The tools in the folder might vary depending on which version of Windows you are using.

Many of the tools in this folder, such as Computer Management, are Microsoft Management Console (MMC) snap-ins that includes their own help topics. To view specific help for an MMC tool, or to search for an MMC snap-in that you do not see in the following list, open the tool, click the Help menu, and then click Help Topics.

Open Administrative Tools by clicking the Start button, clicking Control Panel, clicking System and Maintenance, and then clicking Administrative Tools.

Some common administrative tools in this folder include :

- **Computer Management :** Manage local or remote computers by using a single, consolidated desktop tool. Using Computer Management, you can perform many tasks,

such as monitoring system events, configuring hard disks, and managing system performance.

- **Data Sources (ODBC) :** Use Open Database Connectivity (ODBC) to move data from one type of database (a data source) to another.
- **Event Viewer :** View information about significant events, such as a program starting or stopping, or a security error, that are recorded in event logs.
- **ISCSI Initiator :** Configure advanced connections between storage devices on a network.
- **Local Security Policy :** View and edit Group Policy security settings.
- **Memory Diagnostics Tool :** Check your computer's memory to see if it is functioning properly.
- **Print Management :** Manage printers and print servers on a network and perform other administrative tasks.
- **Reliability and Performance Monitor :** View advanced system information about the central processing unit (CPU), memory, hard disk, and network performance.
- **Services :** Manage the different services that run in the background on your computer.
- **System Configuration :** Identify problems that might be preventing Windows from running correctly.
- **Task Scheduler :** Schedule programs or other tasks to run automatically.
- **Windows Firewall with Advanced Security :** Configure advanced firewall settings on both this computer and remote computers on your network.

Q.3. Discuss the tools of Linux System Administration. **[Important]**

Ans. There are four major players in the world of Linux system administration tools: COAS, Linuxconf, Webmin and YaST. One of these, YaST, is tied specifically to SuSE Linux. The other three, COAS, Linuxconf and Webmin, come by default with some distributions but are independently available for download and installation.

1. Linuxconf : Linuxconf comes with Mandrake Linux and Red Hat Linux, but is also available for most modern Linux distributions. You've probably encountered this tool before if you use one of these distributions, either as the whole package or in one of its modular components. Multiple interfaces for Linuxconf have been available for years, but now we're up to four: GUI, Web, command-line and ncurses.

Linuxconf has actually been around for years, which means its bugs have had longer to shake out than the other distribution-neutral tools. There is a base package with the non-GUI components, and then there are various GUI front-end pieces from a more general X Window System version to one specifically built for GNOME.

Whether you stick with the command-line version or add a GUI front end, you run the tool by typing linuxconf. From here you navigate text or point and click menus to access a wide variety of system settings, everything from basic networking details to GRUB configuration. Linuxconf also plays well with people who refuse to use the root account for anything but the most vital tasks. If you try to run it as root, the tool simply asks for the root password--if this fact makes you nervous, then you may want to consider not using this tool, but this practice is fairly standard in modern administration tools. When you consider that anyone could just try to su to the root account at whim, you start to see why it is so important to have a secure root password in place.

2. Webmin : Webmin comes with, and was recently acquired by, Caldera Linux. This too is not only available for most modern Linux distributions, it also runs on most major flavors of UNIX and is available in around twenty languages (though some modules are not available in all of the languages). Webmin is purely a web-based application and a heavily modular one at that.

There is a set of core modules that handle the usual system administration functionality, and then there are the third-party modules available for administering a variety of packages and services. Where any user can install Linuxconf, Webmin must be installed by root. After that you can access this tool from any user account as long as you know the root password.

There are three separate rows of icons on this tool's front page. On the upper right, you have a pair of administrative links, one to log you out of the Webmin tool and another that allows you to fill out a feedback form that sends your comments back to the Webmin team. In the same top row on the upper left you can click on the word Webmin and go to the product home page. On the upper bar directly beneath those links, there are a series of menu icons, which are, from left to right :

- **Webmin :** takes you back to the main Webmin screen.
- **System :** a collection of configuration issues, such as user and group manipulation, disk quotas and cron jobs.
- **Servers :** configuration routines for a number of servers you may have installed on your system, such as Apache, WU-FTPD and sendmail.
- **Hardware :** configuration utilities for hardware issues such as RAID, printers and disk partitions.
- **Cluster :** a collection of cluster maintenance tools.
- **Others :** a set of tools that system administrators typically need, such as a command prompt, an alias manager and a file manager.

Finally, there is the Webmin tab, which has a series of Webmin management tools :

- **Webmin Actions Log :** if you've enabled Webmin logging, this function allows you to search through the logfiles for what you've utilized this tool to do to your system.
- **Webmin Configuration :** takes you to the amazing number of configuration options available for Webmin, everything from strengthening your Webmin authentication requirements to upgrading either the main package or individual modules.

3. Yet another System Tool (YaST) : YaST2 is the graphical version, and the older YaST is a great fallback if you aren't able to get into the GUI or have not installed a GUI. YaST2 is laid out in a standard file-manager format, with the menu of categories on the left and icons for the various configuration routines on the right, which change to correspond with your category choice. The categories are:

- **Software :** the selection of SuSE software-management utilities, such as the ability to update your system over the Internet or add and remove packages from the SuSE CD-ROM or DVD-ROM.
- **Hardware :** a selection of hardware configuration routines, including printers, sound cards and scanners.
- **Network/Basics :** the selection of configuration tools for modem and other connectivity devices, Ethernet cards and more.
- **Network/Advanced :** the section where you can configure many of your network services, such as sendmail, routing and NIS+.

- **Security&Users :** the selection of user and group-management tools, as well as a couple of useful security tools.
- **System :** the selection of overall system-configuration tools, including a boot script (rc-config) file editor, boot loader configuration editor and routines for changing the language, keyboard and so on.
- **Misc :** a collection of tools that couldn't be otherwise grouped into the other categories. Some of the items represented here involve alternate print setup tools, some for working with log files, and some for communicating with SuSE.

4. The Caldera Open Administration System (COAS) : COAS comes in a rudimentary form with Caldera OpenLinux but is available for most modern Linux distributions, and is open source and covered by the GPL. This tool comes in four different formats, so you can use the interface that's most comfortable: command-line, ncurses (command-line but menu driven), GUI and Web for remote use. This tool is still under development but will be included in its entirety in later versions of Caldera OpenLinux--though there are rumors that because Caldera now has a stake in Webmin, COAS may be on its way out.

This tool is modular, meaning that rather than loading its full set of features into memory when you start it, only the portions you're using come into play. Adding and removing modules is typically transparent as you work with the tool unless you require an external module, one made by a third party or one that for some reason is not included with the core release.

Q.4. Give an overview of various permissions and roles for a system administrator.

Ans. Overview of Permissions and roles : Roles may vary in name, depending on the application. However, in general, the creator of a project site, or the instructor of a course site, has full permissions, and can add or delete content within a worksite.

Permissions allow users to access certain features of a course or project worksite, depending on their roles, and on the decisions made by the site owner and the system administrator.

Roles are collections of permissions. Some roles allow users to simply access or read content, while other roles allow for advanced changes, such as adding participants, editing the site's content, and changing permissions for other roles.

When you create a worksite, or when one is created for you, you have the role with the most permissions and the broadest level of access. You can choose (within the limits established by the system administrator) which tools or functions (e.g., Forums, Schedule, Resources) you want the site to have. For many of these tools or functions, you can set permissions that allow or prevent users from seeing or performing certain tasks, depending on their roles.

The following roles are available :

1. **Course sites :**
   - **Instructor :** Instructors have full permissions throughout the site, including the ability to publish the site and set its global access. Instructors can read, revise, delete, and add both content and participants to a site.
   - **Teaching Assistant :** Teaching Assistants can read, add, and revise most contents in their sections.
   - **Student :** Students can read content, and add content to a site where appropriate.
2. **Project sites :**
   - **Maintain :** The Maintain role has full permissions throughout the site, including the ability to publish the site and set its global access. The Maintain role can read, revise, delete, and add both content and participants to a site.

- **Access :** The Access role can read content and add content to a site where appropriate.

Q.5. What do you mean by a process? What are the various components of a process? [Important]

Ans. A process is the abstraction used by UNIX and Linux to represent a running program. It's the object through which a program's use of memory, processor time, and I/O resources can be managed and monitored.

It is part of the UNIX philosophy that as much work as possible be done within the context of processes, rather than handled specially by the kernel. System and user processes all follow the same rules, so you can use a single set of tools to control them both.

Components of a Process : A process consists of an address space and a set of data structures within the kernel. The address space is a set of memory pages1 that the kernel has marked for the process's use. It contains the code and libraries that the process is executing, the process's variables, its stacks, and various extra information needed by the kernel while the process is running. Because UNIX and Linux are virtual memory systems, there is no correlation between a page's location within a process's address space and its location inside the machine's physical memory or swap space.

The kernel's internal data structures record various pieces of information about each process. Here are some of the more important of these :

- The process's address space map
- The current status of the process (sleeping, stopped, runnable, etc.)
- The execution priority of the process
- Information about the resources the process has used
- Information about the files and network ports the process has opened
- The process's signal mask (a record of which signals are blocked)
- The owner of the process

An execution thread, usually known simply as a thread, is the result of a fork in execution within a process. A thread inherits many of the attributes of the process that contains it (such as the process's address space), and multiple threads can execute concurrently within a single process under a model called multithreading.

Concurrent execution is simulated by the kernel on old-style uniprocessor systems, but on multicore and multi-CPU architectures the threads can run simultaneously on different cores. Multithreaded applications such as BIND and Apache benefit the most from multicore systems since the applications can work on more than one request simultaneously. Many of the parameters associated with a process directly affect its execution: the amount of processor time it gets, the files it can access, and so on.

A process has certain attributes that directly affect execution, these include :

- **PID :** The PID stands for the process identification. This is a unique number that defines the process within the kernel.
- **PPID :** This is the process Parent PID, the creator of the process.
- **UID :** The User ID number of the user that owns this process.
- **EUID :** The effective User ID of the process.
- **GID :** The Group ID of the user that owns this process.
- **EGID :** The effective Group User ID that owns this process.
- **priority :** The priority that this process runs at.

Q.6. Define the lifecycle of a process.

Ans. To create a new process, a process copies itself with the **fork** system call. **Fork** creates a copy of the original process; that copy is largely identical to the parent. The new process has a distinct PID and has its own accounting information. fork has the unique property of returning two different values. From the child's point of view, it returns zero. The parent receives the PID of the newly created child. Since the two processes are otherwise identical, they must both examine the return value to figure out which role they are supposed to play.

After a fork, the child process will often use one of the **exec** family of system calls to begin the execution of a new program. These calls change the program that the process is executing and reset the memory segments to a predefined initial state. The various forms of **exec** differ only in the ways in which they specify the command-line arguments and environment to be given to the new program.

When the system boots, the kernel autonomously creates and installs several processes. The most notable of these is **init**, which is always process number 1. **init** is responsible for executing the system's startup scripts, although the exact manner in which this is done differs slightly between UNIX and Linux. All processes other than the ones the kernel creates are descendants of **init. init** also plays another important role in process management. When a process completes, it calls a routine named **_exit** to notify the kernel that it is ready to die.

It supplies an exit code (an integer) that tells why it's exiting. By convention, 0 is used to indicate a normal or "successful" termination. Before a process can be allowed to disappear completely, the kernel requires that its death be acknowledged by the process's parent, which the parent does with a call to **wait**. The parent receives a copy of the child's exit code (or an indication of why the child was killed if the child did not exit voluntarily) and can also obtain a summary of the child's use of resources if it wishes.

This scheme works fine if parents outlive their children and are conscientious about calling wait so that dead processes can be disposed of. If the parent dies first, however, the kernel recognizes that no wait will be forthcoming and adjusts the process to make the orphan a child of **init. init** politely accepts these orphaned processes and performs the **wait** needed to get rid of them when they die.

Q.7. How can we kill a process using kill command? Also give its syntax.

[Important]

Ans. As its name implies, the **kill** command is most often used to terminate a process. **kill** can send any signal, but by default it sends a TERM. **kill** can be used by normal users on their own processes or by root on any process. The syntax is

```
kill [-signal] pid
```

where signal is the number or symbolic name of the signal to be sent and pid is the process identification number of the target process.

A kill without a signal number does not guarantee that the process will die, because the TERM signal can be caught, blocked, or ignored. The command

```
kill -9 pid
```

"guarantees" that the process will die because signal 9, KILL, cannot be caught.

Use kill -9 only if a polite request fails. We put quotes around "guarantees" because processes can occasionally become so wedged that even KILL does not affect them (usually because of some degenerate I/O vapor lock such as waiting for a disk that has stopped spinning). Rebooting is usually the only way to get rid of these processes.

The killall command performs wildly different functions on UNIX and Linux. Under Linux, killall kills processes by name. For example, the following command kills all Apache web server processes:

```
ubuntu$ sudo killall httpd
```

The standard UNIX killall command that ships with Solaris, HP-UX, and AIX takes no arguments and simply kills all the current user's processes. Running it as root kills init and shuts down the machine.

The pgrep and pkill commands for Solaris, HP-UX, and Linux (but not AIX) search for processes by name (or other attributes, such as EUID) and display or signal them, respectively. For example, the following command sends a TERM signal to all processes running as the user ben:

```
$ sudo pkill -u ben
```

Q.8. Define Process States. Give all the states of the process. **[Important]**

Ans. Process States : A process is not automatically eligible to receive CPU time just because it exists. You need to be aware of the four execution states.

State	Meaning
Runnable	The process can be executed.
Sleeping	The process is waiting for some resource.
Zombie	The process is trying to die.
Stopped	The process is suspended (not allowed to execute).

A runnable process is ready to execute whenever CPU time is available. It has acquired all the resources it needs and is just waiting for CPU time to process its data. As soon as the process makes a system call that cannot be immediately completed (such as a request to read part of a file), the kernel puts it to sleep.

Sleeping processes are waiting for a specific event to occur. Interactive shells and system daemons spend most of their time sleeping, waiting for terminal input or network connections. Since a sleeping process is effectively blocked until its request has been satisfied, it will get no CPU time unless it receives a signal or a response to one of its I/O requests.

Some operations cause processes to enter an uninterruptible sleep state. This state is usually transient and not observed in ps output. However, a few degenerate situations can cause it to persist. The most common cause involves server problems on an NFS filesystem mounted with the "hard" option. Since processes in the uninterruptible sleep state cannot be roused even to service a signal, they cannot be killed. To get rid of them, you must fix the underlying problem or reboot.

Zombies are processes that have finished execution but have not yet had their status collected. If you see zombies hanging around, check their PPIDs with ps to find out where they're coming from. Stopped processes are administratively forbidden to run. Processes are stopped on receipt of a STOP or TSTP signal and are restarted with CONT. Being stopped is similar to sleeping, but there's no way for a process to get out of the stopped state other than having some other process wake it up (or kill it).

Q.9. How can boot process and shutdown process is executed in Linux system?

Ans. One of the most powerful aspects of Linux concerns its open method of starting and stopping the operating system, where it loads specified programs using their particular configurations, permits you to change those configurations to control the boot process, and shuts down in a graceful and organized way.

Beyond the question of controlling the boot or shutdown process, the open nature of Linux makes it much easier to determine the exact source of most problems associated with starting up or shutting down your system. A basic understanding of this process is quite beneficial to everybody who uses a Linux system.

A lot of Linux systems use lilo, the LInux LOader for booting operating systems. However, GRUB is easier to use and more flexible.

Boot process : When an x86 computer is booted, the processor looks at the end of the system memory for the BIOS (Basic Input/Output System) and runs it. The BIOS program is written into permanent read-only memory and is always available for use. The BIOS provides the lowest level interface to peripheral devices and controls the first step of the boot process.

The BIOS tests the system, looks for and checks peripherals, and then looks for a drive to use to boot the system. Usually it checks the floppy drive (or CD-ROM drive on many newer systems) for bootable media, if present, and then it looks to the hard drive. The order of the drives used for booting is usually controlled by a particular BIOS setting on the system. Once Linux is installed on the hard drive of a system, the BIOS looks for a Master Boot Record (MBR) starting at the first sector on the first hard drive, loads its contents into memory, then passes control to it.

This MBR contains instructions on how to load the GRUB (or LILO) boot-loader, using a pre-selected operating system. The MBR then loads the boot-loader, which takes over the process (if the boot-loader is installed in the MBR). In the default Red Hat Linux configuration, GRUB uses the settings in the MBR to display boot options in a menu. Once GRUB has received the correct instructions for the operating system to start, either from its command line or configuration file, it finds the necessary boot file and hands off control of the machine to that operating system.

Shutdown : UNIX was not made to be shut down, but if you really must, use the shutdown command. After completing the shutdown procedure, the -h option will halt the system, while -r will reboot it.

The reboot and halt commands are now able to invoke shutdown if run when the system is in run levels 1-5, and thus ensure proper shutdown of the system, but it is a bad habit to get into, as not all UNIX/Linux versions have this feature.

If your computer does not power itself down, you should not turn off the computer until you see a message indicating that the system is halted or finished shutting down, in order to give the system the time to unmount all partitions. Being impatient may cause data loss.

Q.10. Give some of the main features of GRUB? **[Important]**

Ans. This boot method is called direct loading because instructions are used to directly load the operating system, with no intermediary code between the boot-loaders and the operating system's main files (such as the kernel). The boot process used by other operating systems may differ slightly from the above, however. For example, Microsoft's DOS and Windows operating systems completely overwrite anything on the MBR when they are installed without incorporating any of the current MBR's configurations. This destroys any other information stored in the MBR by other operating systems, such as Linux. The Microsoft operating systems, as well as various other proprietary operating systems, are loaded using a chain loading boot method. With this method, the MBR points to the first sector of the partition holding the operating system, where it finds the special files necessary to actually boot that operating system.

GRUB supports both boot methods, allowing you to use it with almost any operating system, most popular file systems, and almost any hard disk your BIOS can recognize.

GRUB contains a number of other features; the most important include :

- GRUB provides a true command-based, pre-OS environment on x86 machines to allow maximum flexibility in loading operating systems with certain options or gathering information about the system.
- GRUB supports Logical Block Addressing (LBA) mode, needed to access many IDE and all SCSI hard disks. Before LBA, hard drives could encounter a 1024-cylinder limit, where the BIOS could not find a file after that point.
- GRUB's configuration file is read from the disk every time the system boots, preventing you from having to write over the MBR every time you change the boot options.

Q.11. What is a Kernel? Also define its services.

Ans. The UNIX kernel is the software that manages the user program's access to the systems hardware and software resources. These resources range from being granted CPU time, accessing memory, reading and writing to the disk drives, connecting to the network, and interacting with the terminal or GUI interface. The kernel makes this all possible by controlling and providing access to memory, processor, input/output devices, disk files, and special services to user programs.

Kernel Services : The basic UNIX kernel can be broken into four main subsystems :

- Process Management
- Memory Management
- I/O Management
- File Management

These subsystems should be viewed as separate entities that work in concert to provide services to a program that enable it to do meaningful work. These management subsystems make it possible for a user to access a database via a Web interface, print a report, or do something as complex as managing a 911 emergency system. At any moment in the system, numerous programs may request services from these subsystems. It is the kernel's responsibility to schedule work and, if the process is authorized, grant access to utilize these subsystems. In short, programs interact with the subsystems via software libraries and the systems call interface. Refer to your UNIX reference manuals for descriptions of the systems calls and libraries supported by your system.

Q.12. How can we manage Input and Output Peripherals using kernel?

[Important]

Ans. The simplest definition of input/output is the control of data between hardware devices and software. A system administrator is concerned with I/O at two separate levels. The first level is concerned with I/O between user address space and kernel address space; the second level is concerned with I/O between kernel address space and physical hardware devices. When data is written to disk, the first level of the I/O subsystem copies the data from user space to kernel space. Data is then passed from the kernel address space to the second level of the I/O subsystem. This is when the physical hardware device activates its own I/O subsystems, which determine the best location for the data on the available disks.

The OEM (Original Equipment Manufacture) UNIX configuration is satisfactory for many work environments, but does not take into consideration the network traffic or the behaviour of specific applications on your system. System administrators find that they need to reconfigure

the systems I/O to meet the expectations of the users and the demands of their applications. You should use the default configuration as a starting point and, as experience is gained with the demands on the system resources, tune the system to achieve peak I/O performance.

UNIX comes with a wide variety of tools that monitor system performance. Learning to use these tools will help you determine whether a performance problem is hardware or software related. Using these tools will help you determine whether a problem is poor user training, application tuning, system maintenance, or system configuration. sar, iostat, and monitor are some of your best basic I/O performance monitoring tools.

1. **sar :** The sar command writes to standard output the contents of selected cumulative activity counters in the operating system. The following list is a breakdown of those activity counters that sar accumulates.
   - File access
   - Buffer usage
   - System call activity
   - Disk and tape input/output activity
   - Free memory and swap space
   - Kernel Memory Allocation (KMA)
   - Interprocess communication
   - Paging
   - Queue Activity
   - Central Processing Unit (CPU)
   - Kernel tables
   - Switching
   - Terminal device activity
2. **Iostat :** Reports CPU statistics and input/output statistics for TTY devices, disks, and CD-ROMs.
3. **Monitor :** Like the sar command, but with a visual representation of the computer state.

Q.13. Define the working of Process Scheduler.

Ans. Most modern versions of UNIX (for instance, SVR4 and Solaris 2.x) are classified as preemptive operating systems. They are capable of interrupting an executing a process and "freezing" it so that the CPU can service a different process. This obviously has the advantage of fairly allocating the system's resources to all the processes in the system. This is one goal of the many systems architects and programmers who design and write schedulers. The disadvantages are that not all processes are equal and that complex algorithms must be designed and implemented as kernel code in order to maintain the illusion that each user process is running as if it was the only job in the system. The kernel maintains this balance by placing processes in the various priority queues or run queues and apportioning its CPU time-slice based on its priority class (Real-Time versus Timeshare).

Universities and UNIX system vendors have conducted extensive studies on how best to design and build an optimal scheduler. Each vendor's flavor of UNIX--4.4BSD, SVR4, HP-UX, Solaris, and AIX attempts to implement this research to provide a scheduler that best balances its customers' needs. The system administrator must realize that there are limits to the scheduler's ability to service batch, real-time, and interactive users in the same environment. Once the system becomes overloaded, it will become necessary for some jobs to suffer at the

expense of others. This is an extremely important issue to both users and system administrators alike.

Q.14. Discuss the process of Kernel Configuration. How it helps system administrator to alter the behaviour of the computer?

Ans. Kernel configuration is a detailed process in which the system administrator is altering the behaviour of the computer. The system administrator must remember that a change of a single parameter may affect other kernel subsystems, thus exposing the administrator to the "law of unintended consequences."

When Do You Rebuild the Kernel : Kernel components are generally broken into four major groups, and if changes are made to any of these groups, a kernel reconfiguration is required.

Subsystems : These are components that are required for special functionality (ISO9660)

Dump Devices : System memory dumps are placed here when a panic condition exist, core dumps are usually placed at the end of the swap area.

Configurable Parameters : These are tuning parameters and data structures. These are a significant number, and they may have inter-dependencies, so it is important that you are aware of the impact of each change.

Device Drivers : These handle interfaces to peripherals like modems, printers, disks, tape drives, kernel memory, and other physical devices.

The point-to-point protocol (PPP) allows you to dial-in over a telephone line and run Transmission Control Protocol/Internet Protocol (TCP/IP).This allows you to run your GUI applications that use IP from a system that is not directly connected to a network. Let's look at how to configure PPP into the Linux kernel.

Step 1. Linux source code is usually found in the /usr/rc/linux directory. Let's start by changing to this directory by typing the following command.

```
# cd /usr/src/linux
```

Step 2. Type the following :

```
# make config
```

You will be presented with a series of questions asking if you would like to include or enable specific modules, drivers, and other kernel options in your kernel. For our build, we are concerned that we have a modem and the required networking device driver information configured into our kernel. Make sure you have answered [y] to :

Networking Support (CONFIG_NET)

Network device support (CONFIG_NETDEVICES)

TCP/IP networking (CONFIG_INET)

PPP (point-to-point) support (CONFIG_PPP)

For the most part, you can accept the defaults of most of the questions. It's probably a good idea to go through this step once without changing anything to get a feel for the questions you will need to answer. That way you can set your configuration once and move on to the next step.

After you respond to all the questions, you will see a message telling you that the kernel is configured (it still needs to be built and loaded).

Step 3. The next two commands will set all of the source dependencies so that you can compile the kernel and clean up files that the old version leaves behind.

```
# make dep
```

```
# make clean
```

Step 4. To compile the new kernel issue the make command.

```
# make
```

Don't be surprised if this takes several minutes.

Step 5. To see the new kernel, do a long listing of the directory.

```
# ls -al
```

You should see vmlinux in the current directory.

Step 6. You now need to make a bootable kernel.

```
# make boot
```

To see the compressed bootable kernel image, do a long listing on arch/i386/boot. You will see a file named zImage.

Step 7. The last step is to install the new kernel to the boot drive.

```
# make zlilo
```

This command will make the previous kernel (/vmlinuz) to become /vmlinuz.old. Your new kernel image zImage is now/vmlinuz. You can now reboot to check your new kernel configuration. During the boot process, you should see messages about the newly configured PPP device driver scroll across as the system loads.

Once everything checks out and you are satisfied with your new Linux kernel, you can continue on with setting up the PPP software.

Q.15. Define the working of init and inittab. **[Important]**

Ans. init : The kernel, once it is loaded, finds init in sbin and executes it.

When init starts, it becomes the parent or grandparent of all of the processes that start up automatically on your Linux system. The first thing init does, is reading its initialization file, /etc/inittab. This instructs init to read an initial configuration script for the environment, which sets the path, starts swapping, checks the file systems, and so on. Basically, this step takes care of everything that your system needs to have done at system initialization: setting the clock, initializing serial ports and so forth.

Then init continues to read the /etc/inittab file, which describes how the system should be set up in each run level and sets the default run level. A run level is a configuration of processes. All UNIX-like systems can be run in different process configurations, such as the single user mode, which is referred to as run level 1 or run level S (or s). In this mode, only the system administrator can connect to the system. It is used to perform maintenance tasks without risks of damaging the system or user data. Naturally, in this configuration we don't need to offer user services, so they will all be disabled. Another run level is the reboot run level, or run level 6, which shuts down all running services according to the appropriate procedures and then restarts the system.

Use the who to check what your current run level is :

```
willy@ubuntu:~$ who -r
   run-level 2 2006-10-17 23:22  last=S
```

After having determined the default run level for your system, init starts all of the background processes necessary for the system to run by looking in the appropriate rc directory for that run level. init runs each of the kill scripts (their file names start with a K) with a stop parameter. It then runs all of the start scripts (their file names start with an S) in the appropriate run level directory so that all services and applications are started correctly. In fact, you can execute these same scripts manually after the system is finished booting with a command like

/etc/init.d/httpd stop or service httpd stop logged in as root, in this case stopping the web server.

Init run levels : The idea behind operating different services at different run levels essentially revolves around the fact that different systems can be used in different ways. Some services cannot be used until the system is in a particular state, or mode, such as being ready for more than one user or having networking available.

There are times in which you may want to operate the system in a lower mode. Examples are fixing disk corruption problems in run level 1 so no other users can possibly be on the system, or leaving a server in run level 3 without an X session running. In these cases, running services that depend upon a higher system mode to function does not make sense because they will not work correctly anyway. By already having each service assigned to start when its particular run level is reached, you ensure an orderly start up process, and you can quickly change the mode of the machine without worrying about which services to manually start or stop.

Available run levels are generally described in /etc/inittab, which is partially shown below :

```
#
# inittab This file describes how the INIT process should set up the
system in a certain run-level.
# Default run level. The run levels are :
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS
# (The same as 3, if you do not have networking)
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
#
id:5:initdefault:
<--cut-->
```

Feel free to configure unused run levels (commonly run level 4) as you see fit. Many users configure those run levels in a way that makes the most sense for them while leaving the standard run levels as they are by default. This allows them to quickly move in and out of their custom configuration without disturbing the normal set of features at the standard run levels.

If your machine gets into a state where it will not boot due to a bad /etc/inittab or will not let you log in because you have a corrupted /etc/passwd file (or if you have simply forgotten your password), boot into single-user mode..

Q.16. What are Linux Runlevels? Also give all runlevels.

Ans. Linux systems today generally use eight runlevels. Runlevels define what services or processes should be running on the system. The init process can run the system in one of eight runlevels. The system runs only one of the eight runlevels at a time.

The main runlevels are from 0 - 6. Here's what each runlevel is for :

Runlevel 0 : Halt System - To shutdown the system.

Runlevel 1 : Single user mode - Mode for administrative tasks

Runlevel 2 : Basic multi user mode without NFS - Does not configure network interfaces and does not export networks services

Runlevel 3 : Full multi user mode (text based) - Starts the system normally

Runlevel 4 : Unused - For special purposes

Runlevel 5 : Multi user mode with Graphical User Interface - As runlevel 3+ display manager

Runlevel 6 : Reboot System - Reboots the system

Runlevels 1 and 2 are generally used for debugging purposed only, and are not used during normal operations. Most desktop linux distributions boot into runlevel 5, which starts up the Graphical Login Prompt. This allows the user to use the system with X-Windows server enabled. Most servers boot into runlevel 3, which starts the text based login prompt.

Linux runlevels can be changed on the fly using the init tool. If you want to switch from text based operations to the Graphical Interface, you just have to type in 'telinit 5' in the root prompt. This will bring up the Graphical Interface in your system. Each runlevel can be configured by the system administrator. The "/etc/inittab" file has information on which runlevel to start the system at and lists the processes to be run at each runlevel.

Each runlevel has its own directory structure where you can define the order in which the services start. These directories are located in the /etc/rc.d/ directory, under which you have rc1.d, rc2.d, rc3.d.... rc6.d directories where the number from 0 through 6 that corresponds to the runlevel. Inside each directory are symbolic links that point to master initscripts found in /etc/init.d or /etc/rc.d/init.d.

You can also change the runlevel at boot time. If your system uses LILO as the boot manager, you can append the runlevel to the boot command :

LILO : linux 3 or

LILO : linux 5

If your system uses GRUB, you can change the boot runlevel by pressing the `e' key to edit the boot configuration. Append the runlevel to the end of the boot command as shown :

```
kernel/vmlinuz ro root=/dev/hda1 5
```

Q.17. Demonstrate Linux Runlevel scripts.

Ans. Linux Run level scripts : The runlevel scripts are used to bring up many system and networking functions. Since some functions are interdependent on other functions there is some required order in which these scripts must be run in order to bring the system up and to bring it gracefully down. Each runlevel has its own set of start(S) and kill(K) scripts but all these scripts are supported in the directory /etc/rc.d/init.d. This is because the start and kill scripts are soft links to the files in the /etc/rc.d/init.d directory.

The rc script Program : The script file /etc/rc.d/rc is run for the appropriate runlevel (typically 3 or 5). This file does the following :

1. It gets the previous and current system runlevels.
2. If the word confirm is in the file "/proc/cmdline" if sets up to run the scripts below in user confirmation mode.
3. All kill files (files whose first letter is 'K') in the subdirectory "/etc/rc.d/rc3.d" (assuming the previous runlevel was 3) are run. The parameter stop is usually passed on the command line to the kill script.

All startup files (files whose first letter is 'S") in the subdirectory "/etc/rc.d/rc5.d" (assuming the current runlevel is 5) are run. The parameter start is usually passed on the command line to the kill script.

These runlevel scripts are used to bring up (or down) various system services such as cron and gpm along with networking services from the network cards through Samba, and servers like DNS, DHCP, and NFS. A directory listing of the files in the /etc/rc.d/init.d will reveal the many possible services that the system can support.

```
drwxr-xr-x  2 root   root      4096 May 5 10:00.
drwxr-xr-x  10 root  root      4096 Apr 28 04:08 ..
-rwxr-xr-x  1 root   root      766 Sep 13 1999 amd
-rwxr-xr-x  1 root   root      1231 Sep 20 1999 apmd
-rwxr-xr-x  1 root   root      827 Sep 9 1999 arpwatch
-rwxr-xr-x  1 root   root      989 Aug 16 1999 atd
-rwxr-xr-x  1 root   root      4816 Sep 20 1999 autofs
-rwxr-xr-x  1 root   root      1011 Dec 20 08:21 bootparamd
-rw-------  1 root   root      1101824 Mar 2 11:25 core
-rwxr-xr-x  1 root   root      1031 Sep 10 1999 crond
-rwxr-xr-x  1 root   root      975 Apr 28 04:19 dhcpd
-rwxr-xr-x  1 root   root      7386 Sep 20 1999 functions
-rwxr-xr-x  1 root   root      1417 Aug 16 1999 gated
-rwxr-xr-x  1 root   root      1261 Sep 24 1999 gpm
-rwxr-xr-x  1 root   root      3129 Sep 20 1999 halt
-rwxr-xr-x  1 root   root      865 Sep 21 1999 httpd
-rwxr-xr-x  1 root   root      1151 Sep 13 1999 identd
-rwxr-xr-x  1 root   root      1463 Sep 10 1999 inet
-rwxr-xr-x  1 root   root      1924 Aug 30 1999 innd
-rwxr-xr-x  1 root   root      6029 Sep 24 1999 isdn
-rwxr-xr-x  1 root   root      1203 Sep 5 1999 keytable
-rwxr-xr-x  1 root   root      449 Sep 11 1999 killall
-rwxr-xr-x  1 root   root      1172 Sep 24 1999 kudzu
-rwxr-xr-x  1 root   root      1890 Sep 13 1999 ldap
lrwxrwxrwx  1 root   root      43 Dec 17 05:25 linuxconf -> /usr/lib/
                               inuxconf/redhat/scripts/linuxconf
-rwxr-xr-x  1 root   root      1176 Sep 10 1999 lpd
-rwxr-xr-x  1 root   root      1104 Sep 10 1999 mars-nwe
-rwxr-xr-x  1 root   root      1171 Sep 24 1999 mcserv
-rwxr-xr-x  1 root   root      1331 Sep 24 1999 named
-rwxr-xr-x  1 root   root      3217 Sep 20 1999 netfs
-rwxr-xr-x  1 root   root      6573 Sep 21 1999 network
-rwxr-xr-x  1 root   root      2257 Sep 24 1999 nfs
-rwxr-xr-x  1 root   root      1722 Sep 24 1999 nfslock
-rwxr-xr-x  1 root   root      1603 Sep 20 1999 nscd
-r-xr-xr-x  1 root   root      3439 Sep 27 1999 pcmcia
```

These services can be functionally categorized as a system service or a network service. Normally any of these services may be stopped, started, restarted, or status be checked by typing the name of one of these services (with the correct path) followed by the word stop, start, restart, or status respectively. For example the line :

```
/etc/rc.d/init.d/nfs restart
```

will restart network file sharing assuming it was running. To see the status type :

```
/etc/rc.d/init.d/nfs status
```

The rc.local Script Program : The file "/etc/rc.d/rc3.d/S99.local" is a link file to the file "/etc/rc.d/rc.local". This file doesn't do much except for setting up the "/etc/issue" and "/etc/issue.net" files to reflect the system version when a user begins a terminal or telnet session. This is where most administrators will put any system customization they want to make.

Q.18. How do we manage user groups access?

Ans. Linux groups are a mechanism to manage a collection of computer system users. All Linux users have a user ID and a group ID and a unique numerical identification number called a userid (UID) and a groupid (GID) respectively. Groups can be assigned to logically tie users together for a common security, privilege and access purpose. It is the foundation of Linux security and access. Files and devices may be granted access based on a users ID or group ID. This tutorial attempts to show how this is used.

File, directory and device (special file) permissions are granted based on "user", "group" or "other" (world) identification status. Permission is granted (or denied) for read, write and execute access.

Q.19. What is User Administration? How we can add new users to a new machine? **[Important]**

Ans. While performing user administration you will call upon all of your skills in virtually every area of system management. Whether it is keeping disks as empty as possible, finding system bottlenecks, answering questions, or adding new users, your job revolves almost totally around and for the machine's users. Those users are much like the fans of a baseball game to your rendition of the umpire: You have done a good job if you are unnoticed. Though easy to forget in a harried environment, learning and interacting with system esoterica only can happen because the users are at least indirectly paying the freight.

Adding New Users : Logically enough, user administration begins when adding users to a new machine. Functionally, there are a variety of ways to accomplish this task. Each of the methods involves adding information to the password file, /etc/passwd. The/etc/group file is another file that requires attention as are miscellaneous files such as system shell startup files and the system mail alias file.

Q.20. How can we protect our system using password file?

Ans. The format of /etc/passwd is consistent among most flavors of Unix. This file contains the following colon-separated entries:

```
username:pswd:uid:gid:uid comments:directory:shell
```

On some BSD type systems, the file contains slightly different colon-separated entries :

```
username:pswd:uid:gid:user class:pswd change:acct expiration:
    uid comments:directory:shell
```

The username is what the user types in at the UNIX login: prompt. Usually, the field's makeup is eight or less alphanumeric characters with the alphabetic characters in lower case.

It should be unique. This field should not contain colons because the colon is used as the field delimiter. For best compatibility, this field should not contain dots and should not begin with a hyphen or a plus sign.

The pswd field is a password entry and can have many different forms. The entry can be blank, indicating that there is no password required for login. The position can contain up to 13 characters that are the encrypted version of the password for the user. The location can contain a character not in the following set { . / 0-9 A-Z a-z } connoting that the username is valid but cannot be logged into. For example an "*" is not in this set. If it is used then the username is a valid account but cannot be logged into.

The characters (in order) are: . / 0-9 A-Z a-z. The "." character is equated to the number zero and "z" is equated to 63. References made to characters using their numeric value are common. The characters indicate the number of weeks the password is valid and the number of weeks that must expire before a change is allowed to the password, respectively. If the former is zero (dot), the user must change the password at the next login attempt. Though generally frowned upon by security conscience individuals, if the latter is greater than the former, only the superuser can change the password.

The uid or user id is simply a unique numerical user value for the username. Normally this value is a positive number up to 65535, although some systems can handle non recommended double precision user id numbers. If this is non-unique, all usernames with the same user id look like a single (usually the first) username with this user id. Some user ids are reserved or special. They include:

0:	**The superuser**
1-10:	Daemons and pseudo users
11-99:	System, reserved and "famous" users
100+:	Normal users
60001:	"nobody" (occasionally 32000 or 65534)
60002:	"noaccess" (occasionally 32001)

The gid or group id is a numerical default group id for the username. This number corresponds to an entry in the/etc/group file.

The uid comments field is historically known as GECOS, or GCOS, information for the operating system it originated from. For a generic finger command to accurately display this information, it should contain the user's real name, company or office number, office phone number, and home telephone number separated by commas. Not all entries need to be specified although placeholders must be kept if trying to keep the general GECOS syntax. For example,Homer User,,,800-IAM-HOME would show entries for the user's real name and the user's home telephone number. The real user name is also displayed by the mail system as part of the outgoing mail headers.

The directory field is commonly known as a username's home directory or initial working directory. Basically, it is the directory the user is placed in after being logged in by the system but before the user's personal startup files are executed.

The shell field is the command interpreter or program that the username is placed in after logging in. Among the many shells are: sh (Bourne), ksh (Korn), csh , tcsh (TENEX/TOPS-20 type C), BASH (Bourne Again Shell). If not specified, the default shell is the Bourne shell. Note that this entry does not have to be a shell; it can be a program that locks the username

into a captive application. For this field to be valid, some systems require this entry to be present in a shell validation file.

The class field is unused but is for specifying a class of user attributes for the username.

The pswd change field indicates how soon a password must be changed. It is the number of seconds since the epoch (Jan 1 1970 @ 00:00). If omitted, no forced change of the password occurs.

The acct expiration field is the number of seconds since the epoch until the account will expire. If left blank, account expiration is not enforced for the username.

Additionally, if Network Information Service (NIS) / Yellow Pages (YP) is installed and running, the password file can include other types of entries. The additional username field entries include :

+	**all YP entries should be included**
+username	include the explicit username from YP
-username	exclude the explicit username from YP
+@netgroup	include all usernames from YP from the desired group
-@netgroup	exclude all usernames from YP from the desired group

Generally, within such entries, if the uid, gid, uid comments, directory, or shell fields are specified, they supplant the value that YP sends for that field. Also, be aware that these entries are taken in order, so the first occurrence, not the last occurrence, dictates what is going to happen. For example :

```
root:x:0:0:Superuser:/:
daemon:*:1:5::/:/sbin/sh
bin:*:2:2::/usr/bin:/sbin/sh
sys:*:3:3::/:
adm:*:4:4::/var/adm:/sbin/sh
uucp:*:5:3::/var/spool/uucppublic:/usr/lbin/uucp/uucico
lp:*:9:7::/var/spool/lp:/sbin/sh
nuucp:*:11:11::/var/spool/uucppublic:/usr/lbin/uucp/uucico
hpdb:*:27:1:ALLBASE:/:/sbin/sh
nobody:*:-2:60001::/:
dave:x:100:10:Dave G,13,x3911,unlisted:/usr1/dave:/bin/tcsh
charlene:x:101:10:Charlene G,14,x1800,unlisted:/usr1/charlene:/bin/tcsh
john:x:102:60:John S,2,555-1234,x1400:/usr2/john:/bin/ksh
georgia:x:103:60:Georgia S,11,x143,x143:/usr2/georgia:/bin/csh
-steve:::::
+@friends:::20:::
+wayne::102:::/usr3/wayne:/bin/sh
```

Username steve is always excluded, even if it occurs within the netgroup friends. All friends are included with the noted exception. Every friends included is placed into the 20 group by default. YP wayne is included. All of his fields, except the group id and user id comment fields, are overridden with the specified information. Notice in our sample that the letter x in the

pswd field was substituted for the actual encrypted password for this publication. The character * in the pswd field is shown as found for these nonlogin psuedo users.

Q.21. What do you mean by Shadow Password File?

Ans. Since /etc/passwd is usually globally readable, security conscientious sites normally use a shadow password scheme that is available under most Unix operating systems. Ultimately, this redirects the encrypted passwords to another restricted read file that may or may not contain other information. This scheme is employed because average machines today can crack user passwords readily if the user's choice is bad. Bad choices include any dictionary word, the login name, no password, or any information included in the user id comment field. The schemes used for shadow password files vary considerably among the various UNIX variants.

For example, IRIX and Solaris systems have a file named /etc/shadow that is generated by running the commandpwconv that includes the following :

```
username:pswd:lastchg:min:max:warn:inactive:expire:flag
```

- The username is a copy of the username from the /etc/passwd file.
- The pswd field contains either the 13-character encrypted password; null, indicating that no password is needed for login; or a string containing a character not from the following set--{ . / 0-9 A-Z a-z}. If the password contains a character not from the encryption set, the username cannot be logged into. Normally, system administrators would use* or *LK* for the entry.
- The lastchg field is the number of days from the epoch that the password was last changed.
- The min field is the minimum number of days to elapse between a successful password change and another change.
- The max field is the maximum number of days that the password will be valid.
- The warn field contains the number of days before password expiration that the user will begin to get warnings that the password is about to expire.
- The inactive field is the number of days that the username can remain inactive before not being allowed to log in.
- The expire field is an absolute number of days specification. When used, this specifies when a username will no longer be considered valid to log into.
- The flag field is currently unused.

HP-UX has adopted another scheme for shadowing the password file on a trusted system. Each username has a file named /tcb/files/auth/first letter/username where first letter is the beginning letter of the username, and username is the login name for the user. For example, /tcb/files/auth/b/buster would exist for the username buster. This file contains information that is termcap in appearance and at last count, there were 32 possible options contained within this file that deal with user security. In general, the file holds :

- The username and user id mirrored from the password file.
- The encrypted password for the username, if any.
- **The names of :** The owner of the account. The last account to change the password on this account if it was not the account itself. The last successful and unsuccessful login attempt terminal name or host name.
- **The times indicating :** The number of seconds allowed between successful password changes. When the password expires (next login will require the user change the password). When the lifetime of the password expires (only the sysadmin can reallow

login). The last time a successful and unsuccessful password change or attempt was made. When the account expires (an absolute-lifetime offsets from a password change). The maximum time allowed between logins. How long before password expiration that a user should be notified. The time of day logins for the account are valid. The last time a successful and unsuccessful login entry or attempt was made.

- **Flags showing :** If the username is allowed to boot the system. If audits occur for the username. If the user can select the account's password or must use a system generated one. If the user can have the system generate a password for the account. Whether a chosen password undergoes a check for being too easily guessed. If the account can have no (null) password. If the user can generate "random" characters and letters for a password. If the account is administratively locked.
- **Numbers specifying :** An audit id for the account. The maximum length a password can be. An additional random number an account must specify to a password if the system administrator reset the password. The count of unsuccessful logins until the next successful one. The maximum consecutive unsuccessful login attempts allowed before the account is locked.

Berkeley-type systems have yet another type of shadowing system that uses the files /etc/master.passwd or/etc/spwd.db.

Q.22. What is the use of Group File? Illustrate it.

Ans. Another file referred to previously that comes into play is /etc/group. It is part of the general protection scheme employed by UNIX: user, group, and other permissions on files. The colon-separated template for the file appears as :

```
group_name:password:group_id:list
```

- The group_name field contains the textual name of the group.
- The password field is a placeholder for an encrypted password for the group. If null, no password is required.
- The group_id field contains a unique numerical value for the group.
- The list field contains a comma-separated list of users who belong to this group. Users need not be listed in groups that are specified for their username in /etc/passwd.

A sample /etc/group file follows :

```
root::0:root
other::1:root,hpdb
bin::2:root,bin
sys::3:root,uucp
adm::4:root,adm
daemon::5:root,daemon
mail::6:root
lp::7:root,lp
tty::10:
nuucp::11:nuucp
users::20:root,dave,charlene,john,georgia,operator,steve,judy,wayne,jamie
nogroup:*:-2:
systech::110:dave,disdb,diskf,disjs,dispm,diskj
dba::201:oracle,john,kathy,pete
```

```
psdev::202:ps001,ps002,ps101
hrdev::203:hrprw,hrpps,hrpsl,hrpla,consult1,consult3,
fsdev::209:glpmk,glpsf,consult2
fsftp::222:glpmk,glpsf,glpjh
```

If Network Information Service/Yellow Pages is enabled, this file, like /etc/passwd, c contain entries beginning with a minus or plus sign to exclude or include (respectively) gro information from NIS/YP.

Q.23. What is a shell? Also define shell prompt? **[Importan**

Ans. A UNIX shell is what gives UNIX its powerful capabilities. The shell is the part of th system with which the user interacts, and as such it is arguably the most important part of UNIX system. A UNIX shell interprets commands such as "pwd", "cd" or "traceroute" an sends the proper instructions to the actual operating system itself. Other functions of a she include scripting capability, path memory, multitasking, and file handling.

The shell provides you with an interface to the UNIX system. It gathers input from yo and executes programs based on that input. When a program finishes executing, it displa that program's output. A shell is an environment in which we can run our commands, program and shell scripts. There are different flavors of shells, just as there are different flavors o operating systems. Each flavor of shell has its own set of recognized commands and function While the prompt is displayed, you can type a command.

Shell Prompt : The prompt, $, which is called command prompt, is issued by the shel

The shell reads your input after you press Enter. It determines the command you wan executed by looking at the first word of your input. A word is an unbroken set of character Spaces and tabs separate words.

Following is a simple example of date command which displays current date and time:

```
$dateThu Jun 25 08:30:19 MST 2009
```

Q.24. Define some available Shells.

Ans. There are several shells available on the OIT Unix systems: bash (Bourne Again), cs (C), ksh (Korn), rksh (restricted Korn), sh, tcsh (Turbo C) and zsh (Z). By default, all enable accounts are set up to use csh (C shell). For those who intend to use Unix more extensively tcsh (Turbo C Shell) is a popular shell. Bash (Bourne Again Shell) is another popular shel often used by beginning Computer Science students.

1. The C Shells : There are two C shells : Csh and Tcsh. Csh is the most common she on the Princeton campus because it is the Princeton-specified default. Many commercial Uni systems set ksh as the default; Linux specifies bash.

There are several nifty things about the C shells :

- Aliasing
- Prompt Customization
- History Recording
- Filename Completion (tcsh only)

The setenv command controls shell variables. For example, if you wanted to use the M Server to use Xterms and other sundry Unix windows in your dorm room, you could SSH arizona as usual, and upon entering the shell type "setenv DISPLAY yourNetID.student.princeton.edu:0.0". This sets the environment variable to display all windo commands to your PC at home.

Another useful function of setenv variables is use in scripting. In a shell script, referring to

```
$HOME
```

would be the same as replacing

```
$HOME
```

with

```
/u/yourNetID
```

These variables work the same way in almost all shells.

The Tcsh is an extension of csh, and allows command line editing, file name completion, plus previous command recall via the up-arrow (which are all very useful). Tcsh is very versatile, and has all the features that csh has.

For example, filename completion is an essential feature of tcsh. When typing a command, such as netscape, you can type

```
netsc
```

and press [TAB] at this point. Since there are no other commands on the arizona systems that begin with "mythes", the shell finishes the command for you to

```
mythesis_for_school
```

If you spend much time manipulating files in Unix, you will find this feature indispensable. It works with directory names, as well. In almost any situation, you can fill in filenames and commands using the [TAB] key.

2. The Bourne Again Shell : The Bash shell is a redundant acronym short for "Bourne Again Shell". One advantage is that it is made by the Free Software Foundation. A feature of bash is its way of manipulating environment variables. For example, bash uses the "export" command to keep track of the $DISPLAY value (important for using X-terminals) instead of the csh/sh/tcsh-standard "setenv DISPLAY=" command.

So, if you were logged in via an ssh window and wanted to get windows on your x server from that session, you would type

```
export DISPLAY=$REMOTEHOST:0.0
```

That would set the DISPLAY variable to your logged in host.

When it really comes down to it, these shells are pretty much the same, except that bash uses somewhat different terminology.

One cool thing about bash is that system crackers do NOT like it. Bash keeps a record of what commands you execute, and stores it in a file called .bash_history in your home directory. Since this leaves traces of what commands are executed by a user of the account, a cracker would either delete this file or could be detected by looking at backed up copies of the file. Either way, crackers dislike bash. If you use bash, you can keep tabs on what goes on in your account.

3. The Korn Shell : Unlike Bash, the Korn shell (ksh) and the Korn Restricted Shell (rksh) were written by David Korn of Bell Labs. What advantages are there to using Korn? Korn has a command history feature, a command alias feature, support for shell scripts, arithmetic expressions, filename completion (disabled by default), and command line editing (also disabled by default). The sheer number of things disabled by default makes it a chore to start using any Korn shell. To make matters worse, the K shells all use the "export" feature, instead of "setenv."

- ksh keeps a 128-command history file by default

- When editing a command at the prompt, you have to use ridiculously annoying keys, starting with ESC, then "h" for right and "l" for left. Arrow keys are not supported.
- You can't change the look of your command prompt.
- Korn shells don't support filename completion.

Q.25. What do you mean by Restricted Shells?

Ans. Restricted shells can be invoked in any of the following ways :

```
rksh  Korn shell
ksh -r
set -r

rsh   Bourne Shell
set -r
```

Restricted shells can also be set up by supplying rksh and rsh in the shell field of /etc/passwd or by using them as the value for the SHELL variable.

Restricted shells act the same as their non-restricted counterparts, except that the following are prohibited :

- Changing directory (i.e., using cd).
- Setting the PATH variable. rksh also prohibits setting ENV and SHELL.
- Specifying a / for command names or pathnames.
- Redirecting output (i.e., using > and >>).

Shell scripts can still be run, since in that case the restricted shell will call ksh or sh to run the script.

Q.26. Give all the commands used in UNIX.

Ans. Contents : Each entry is labeled with the command name on the outer edge of the page. The syntax line is followed by a brief description and a list of all available options. (Obsolete options have been marked as "SVR3 only.") Many commands come with examples at the end of the entry. If you need only a quick reminder or suggestion about a command, you can skip directly to the examples.

Alphabetical Summary of Commands :

- admin
- apropos
- ar
- as
- at
- atq
- atrm
- awk
- banner
- basename
- batch
- bc
- bdiff
- bfs
- cal
- calendar
- cancel
- cat
- cb
- cc
- cd
- cdc
- cflow
- chgrp
- chkey
- chmod
- chown
- clear
- cmp
- cof2elf
- sh
- shl
- shutdown
- size
- sleep
- soelim
- sort
- spell
- split
- srchtxt
- strings
- strip

- stty
- tabs
- tar
- telnet
- timex
- tr
- truss
- tty
- uncompress
- units
- users
- uuencode
- uuname
- uuto
- val
- vi
- wait
- what
- who
- write
- zcat
- su
- tail
- tbl
- test
- touch
- troff
- tset
- umask
- unget
- unpack
- uucp
- uuglist
- uupick
- uux
- vc
- view
- wall
- whatis
- whoami
- xargs
- sum
- talk
- tee
- time
- tput
- true
- tsort
- uname
- uniq
- uptime
- uudecode
- uulog
- uustat
- vacation
- vedit
- w
- wc
- which
- whois
- yacc

Q.27. How many types of accounts are there on a Unix System? **[Important]**

Ans. There are three types of accounts on a Unix system :

1. **Root account :** This is also called superuser and would have complete and unfettered control of the system. A superuser can run any command without any restriction. This user should be assumed as a system administrator.
2. **System accounts :** System accounts are those needed for the operation of system-specific components for example mail accounts and the sshd accounts. These accounts are usually needed for some specific function on your system, and any modifications to them could adversely affect the system.
3. **User accounts :** User accounts provide interactive access to the system for users and groups of users. General users are typically assigned to these accounts and usually have limited access to critical system files and directories.

Unix supports a concept of Group Account which logically groups a number of accounts. Every account would be a part of any group account. Unix groups play important role in handling file permissions and process management.

Q.28. How can we manage Users and Groups in UNIX? **[Important]**

Ans. There are three main user administration files :

1. **/etc/passwd :** Keeps user account and password information. This file holds the majority of information about accounts on the Unix system.
2. **/etc/shadow :** Holds the encrypted password of the corresponding account. Not all the systems support this file.
3. **/etc/group :** This file contains the group information for each account.

4. /etc/gshadow : This file contains secure group account information.

Check all the above files using cat command.

Following are commands available on the majority of Unix systems to create and manage accounts and groups :

Command	Description
Useradd	Adds accounts to the system.
Usermod	Modifies account attributes.
Userdel	Deletes accounts from the system.
groupadd	Adds groups to the system.
groupmod	Modifies group attributes.
Groupdel	Removes groups from the system.

1. Create a Group : You would need to create groups before creating any account otherwise you would have to use existing groups at your system. You would have all the groups listed in /etc/groups file.

All the default groups would be system account specific groups and it is not recommended to use them for ordinary accounts. So following is the syntax to create a new group account :

```
groupadd [-g gid [-o]] [-r] [-f] groupname
```

Here is the detail of the parameters:

Option	Description
-g GID	The numerical value of the group's ID.
-o	This option permits to add group with non-unique GID
-r	This flag instructs groupadd to add a system account
-f	This option causes to just exit with success status if the specified group already exists. With -g, if specified GID already exists, other (unique) GID is chosen
groupname	Actual group name to be created.

If you do not specify any parameter then system would use default values.

Following example would create developers group with default values, which is very much acceptable for most of the administrators.

```
[amrood]$ groupadd developers
```

2. Modify a Group : To modify a group, use the groupmod syntax :

```
[amrood]$ groupmod -n new_modified_group_name old_group_name
```

To change the developers_2 group name to developer, type :

```
[amrood]$ groupmod -n developer developer_2
```

Here is how you would change the financial GID to 545 :

```
[amrood]$ groupmod -g 545 developer
```

3. Delete a Group : To delete an existing group, all you need is the groupdel command and the group name. To delete the financial group, the command is :

```
[amrood]$ groupdel developer
```

This removes only the group, not any files associated with that group. The files are still accessible by their owners.

1. Create an Account : Let us see how to create a new account on your Unix system. Following is the syntax to create a user's account :

```
useradd -d homedir -g groupname -m -s shell -u userid accountname
```

Here is the detail of the parameters :

Option	Description
-d homedir	Specifies home directory for the account.
-g groupname	Specifies a group account for this account.
-m	Creates the home directory if it doesn't exist.
-s shell	Specifies the default shell for this account.
-u userid	You can specify a user id for this account.
accountname	Actual account name to be created

If you do not specify any parameter then system would use default values. The useradd command modifies the /etc/passwd, /etc/shadow, and /etc/group files and creates a home directory.

Following is the example which would create an account mcmohd setting its home directory to/home/mcmohd and group as developers. This user would have Korn Shell assigned to it.

```
[amrood]$ useradd -d /home/mcmohd -g developers -s /bin/ksh mcmohd
```

Before issuing above command, make sure you already have developers group created using groupadd command.

Once an account is created you can set its password using the passwd command as follows :

```
[amrood]$ passwd mcmohd20
Changing password for user mcmohd20.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

When you type passwd accountname, it gives you option to change the password provided you are super user otherwise you would be able to change just your password using the same command but without specifying your account name.

2. Modify an Account : The usermod command enables you to make changes to an existing account from the command line. It uses the same arguments as the useradd command, plus the -l argument, which allows you to change the account name.

For example, to change the account name mcmohd to mcmohd20 and to change home directory accordingly, you would need to issue following command :

```
[amrood]$ usermod -d /home/mcmohd20 -m -l mcmohd mcmohd20
```

3. Delete an Account : The userdel command can be used to delete an existing user. This is a very dangerous command if not used with caution.

There is only one argument or option available for the command: .r, for removing the account's home directory and mail file.

For example, to remove account mcmohd20, you would need to issue following command :

```
[amrood]$ userdel -r mcmohd20
```

If you want to keep her home directory for backup purposes, omit the -r option. You can remove the home directory as needed at a later time.

Q.29. What is User Management in Linux System? How is it accomplished?

Ans. User management is a critical part of maintaining a secure system. Ineffective user and privilege management often lead many systems into being compromised. Therefore, it is important that you understand how you can protect your server through simple and effective user account management techniques.

Where is root?

Users are encouraged to make use of a tool by the name of sudo to carry out system administrative duties. Sudo allows an authorized user to temporarily elevate their privileges using their own password instead of having to know the password belonging to the root account. This simple yet effective methodology provides accountability for all user actions, and gives the administrator granular control over which actions a user can perform with said privileges.

If for some reason you wish to enable the root account, simply give it a password :

```
sudo passwd
```

Sudo will prompt you for your password, and then ask you to supply a new password for root as shown below :

```
[sudo] password for username: (enter your own password)
Enter new UNIX password: (enter a new password for root)
Retype new UNIX password: (repeat new password for root)
passwd: password updated successfully
```

To disable the root account, use the following passwd syntax :

```
sudo passwd -l root
```

You should read more on Sudo by checking out it's man page :

```
man sudo
```

By default, the initial user created by the Ubuntu installer is a member of the group "admin" which is added to the file/etc/sudoers as an authorized sudo user. If you wish to give any other account full root access through sudo, simply add them to the admin group.

Adding and Deleting Users : The process for managing local users and groups is straight forward and differs very little from most other GNU/Linux operating systems. Ubuntu and other Debian based distributions, encourage the use of the "adduser" package for account management.

To add a user account, use the following syntax, and follow the prompts to give the account a password and identifiable characteristics such as a full name, phone number, etc.

```
sudo adduser username
```

To delete a user account and its primary group, use the following syntax :

```
sudo deluser username
```

Deleting an account does not remove their respective home folder. It is up to you whether or not you wish to delete the folder manually or keep it according to your desired retention policies.

Remember, any user added later on with the same UID/GID as the previous owner will now have access to this folder if you have not taken the necessary precautions.

You may want to change these UID/GID values to something more appropriate, such as the root account, and perhaps even relocate the folder to avoid future conflicts :

```
sudo chown -R root:root /home/username/
sudo mkdir /home/archived_users/
sudo mv /home/username /home/archived_users/
```

To temporarily lock or unlock a user account, use the following syntax, respectively :

```
sudo passwd -l username
sudo passwd -u username
```

To add or delete a personalized group, use the following syntax, respectively :

```
sudo addgroup groupname
sudo delgroup groupname
```

To add a user to a group, use the following syntax :

```
sudo adduser username groupname
```

Q.30. How can we implement User Profile Security in Unix? Also give the policies of security.

Ans. When a new user is created, the adduser utility creates a brand new home directory named /home/username, respectively. The default profile is modeled after the contents found in the directory of /etc/skel, which includes all profile basics.

If your server will be home to multiple users, you should pay close attention to the user home directory permissions to ensure confidentiality. By default, user home directories in Ubuntu are created with world read/execute permissions. This means that all users can browse and access the contents of other users home directories. This may not be suitable for your environment.

To verify your current users home directory permissions, use the following syntax :

```
ls -ld /home/username
```

The following output shows that the directory /home/username has world readable permissions :

```
drwxr-xr-x 2 username username 4096 2007-10-02 20:03 username
```

You can remove the world readable permissions using the following syntax :

```
sudo chmod 0750 /home/username
```

A much more efficient approach to the matter would be to modify the adduser global default permissions when creating user home folders. Simply edit the file /etc/adduser.conf and modify the DIR_MODE variable to something appropriate, so that all new home directories will receive the correct permissions.

```
DIR_MODE=0750
```

After correcting the directory permissions using any of the previously mentioned techniques, verify the results using the following syntax :

```
ls -ld /home/username
```

The results below show that world readable permissions have been removed :

```
drwxr-x--- 2 username username 4096 2007-10-02 20:03 username
```

Password Policy : A strong password policy is one of the most important aspects of your security posture. Many successful security breaches involve simple brute force and dictionary attacks against weak passwords. If you intend to offer any form of remote access involving your local password system, make sure you adequately address minimum password complexity

requirements, maximum password lifetimes, and frequent audits of your authentication systems.

Minimum Password Length : By default, Ubuntu requires a minimum password length of 6 characters, as well as some basic entropy checks. These values are controlled in the file / etc/pam.d/common-password, which is outlined below.

```
password [success=2 default=ignore] pam_unix.so obscure sha512
```

If you would like to adjust the minimum length to 8 characters, change the appropriate variable to min=8. The modification is outlined below.

```
password [success=2 default=ignore] pam_unix.so obscure sha512 min=8
```

Password Expiration : When creating user accounts, you should make it a policy to have a minimum and maximum password age forcing users to change their passwords when they expire.

To easily view the current status of a user account, use the following syntax :

```
sudo chage -l username
```

The output below shows interesting facts about the user account, namely that there are no policies applied :

```
Last password change : Jan 20, 2008
Password expires : never
Password inactive : never
Account expires : never
Minimum number of days between password change : 0
Maximum number of days between password change : 99999
Number of days of warning before password expires : 7
```

To set any of these values, simply use the following syntax, and follow the interactive prompts :

```
sudo chage username
```

The following is also an example of how you can manually change the explicit expiration date (-E) to 01/31/2008, minimum password age (-m) of 5 days, maximum password age (-M) of 90 days, inactivity period (-I) of 5 days after password expiration, and a warning time period (-W) of 14 days before password expiration.

```
sudo chage -E 01/31/2011 -m 5 -M 90 -I 30 -W 14 username
```

To verify changes, use the same syntax as mentioned previously :

```
sudo chage -l username
```

The output below shows the new policies that have been established for the account :

```
Last password change : Jan 20, 2008
Password expires : Apr 19, 2008
Password inactive : May 19, 2008
Account expires : Jan 31, 2008
Minimum number of days between password change : 5
Maximum number of days between password change : 90
Number of days of warning before password expires : 14
```

Q.31. How Group ownership of Files and Directories are accomplished? What are the Group permissions of Files and Directories? **[Important]**

Ans. Every file and directory has a username and a groupname associated with it. We say the username is the owner and the groupname owns the file or directory. A directory is a collection of files and possibly other sub-directories. There are commands for managing group ownership for both directories and files. In the example commands given in this document we use filename to indicate the name of a file, but in most cases you can use the same command with the name of a directory.

The long format of the listing command gives the permission modes, the owner and the group for both files and directories. Use the ls -dl filename command to get a one-line listing of a single file or directory. The command ll (or ls -l) will list all the files and directories in your current directory. The ones beginning with a "d" are directories.

When a file or directory is first created it takes as its group the current group of your shell. This is the default group for all login shells, but you can start another shell with any group with the command newgrp project. If you are going to create files for a secondary group then it is easier to create all these files from a shell started with thenewgrp command.

If you want to change the group associated with a file or directory which already exists use the command chgrp project filename. You must be the owner of the file filename and you must be a member of the group project to make the change. If the long listing shows a file which is not owned by the proper group you must contact the owner of the file and get them to change the group.

In many cases the group ownership does not matter, but if you want to share a file with a group, then it is important that you get the ownership correct. Otherwise you may be inviting all users to put their large files in your directory.

Group permissions of Files and Directories : Just setting up a file to be owned by a group does not give your group any access to the file. Granting and limiting access is done by setting permission modes. You can see the permission modes as a set of 10 letters or dashes in the long listing of a file or directory using the ls -dl command. The -dl option on the ls command will list the information for the directory or file in long format. Without the "d" all the files in the directory would be listed instead of just the directory you asked for. For example to get a long listing for a directory with the name kneeland

```
<2>% ls -dl kneeland
drwxr-x--- 3 dnairn 0217 512 Aug 14 15:14 kneeland
```

The first string of characters are the mode, the following number is a count, the user name is the owner and the 4 digit account code is the group.

`mode: drwxr-x---`

Begins with a "d" so it is a directory, The owner, dnairn, has permission modes rwx which is full access. Any other user in group 0217 has permission modesr-x which is browsing access can read and search without permission to add, rename or delete files in the directory.) Every other user, that is not dnairn and not in group 0217 has permission modes --- which is no access.

`count: 3`

There are three files in this directory. The count is always one if you are listing a file.

`username: dnairn`

The user with login name dnairn is the owner of the file. The owner will have permission modes according the the first three codes after the "d". The owner always can change permission modes with the chmod command.

`groupname: 0217`

The directory is said to be owned by this group. Any user in group 0217 , except dnairn, will have permissions granted according to the middle three codes in the permission modes.

Some UNIX Commands for Working with Groups

Command	Description	Example
chdgrp	List groups with title and remaining balance	chdgrp
groups	See groups to which you belong with primary group first	groups
id	See current group as part of your id	id
newgrp	Start a shell in a different group	newgrp 1234
chmod	Change permissions for directories and files	chmod g+rwx myfile
chgrp	Change group ownership of directories and files	chgrp 1234 myfile
ls	List file permissions	ls -l

UNIT—2

MANAGING UNIX FILE SYSTEM AND CONFIGURING THE TCP/IP NETWORKING

Q.1. What do you understand by Filesystems in UNIX? Discuss the role of system administrator in managing UNIX Filesystems.

Ans. To Unix systems, a filesystem is some device (such as a hard drive, floppy, or CD-ROM) that is formatted to store files. Filesystems can be found on hard drives, floppies, CD-ROMs, and other storage media that permit random access. (A tape allows only sequential access, and therefore can't contain a filesystem per se.)

The exact format and means by which files are stored is not important; the system provides a common interface for all *filesystem types* it recognizes. Under Linux, filesystem types include the Second Extended filesystem, or *ext2fs*, which you probably use to store Linux files (also *ext3* is slowly taking over); the VFAT filesystem, which allows files on Windows 95/98/ME partitions and floppies to be accessed under Linux; and several others, including the ISO 9660 filesystem used by CD-ROM.

Each filesystem type has a very different underlying format for storing data. However, when you access any filesystem under Linux, the system presents the data as files arranged into a hierarchy of directories, along with owner and group IDs, permission bits, and the other characteristics with which you're familiar.

In fact, information on file ownership, permissions, and so forth is provided only by filesystem types that are meant to be used for storing Linux files. For filesystem types that don't store this information, the kernel drivers used to access these filesystems "fake" the information. For example, the MS-DOS filesystem has no concept of file ownership; therefore, all files are presented as if they were owned by root. This way, above a certain level, all filesystem types look alike, and each file has certain attributes associated with it. Whether this data is actually used in the underlying filesystem is another matter altogether.

As the system administrator, you need to know how to create filesystems should you want to store Linux files on a floppy or add additional filesystems to your hard drives. You also need to know how to use the various tools to check and maintain filesystems should data corruption occur. Also, you must know the commands and files used to access filesystems — for example, those on floppy or CD-ROM.

Q.2. How can you access any filesystem under Linux? Illustrate the format of *mount* command with the help of examples.

Ans. Mounting Filesystem : In order to access any filesystem under Linux, you must mount it on a certain directory. This makes the files on the filesystem appear as though they reside in the given directory, allowing you to access them.

The mount *command is used to do this and usually must be executed as root. (Ordinary users can use* mount *if the device is listed in the /etc/fstab file.) The format of this command is :*

```
mount -t type device mount-point
```

where type is the type name of the filesystem, device is the physical device where the filesyste resides (the device file in /dev), and mount-point is the directory on which to mount th filesystem. You have to create the directory before issuing mount.

For example, if you have a Second Extended filesystem on the partition /dev/hda2 an wish to mount it on the directory /mnt, use the command :

```
mount -t ext2 /dev/hda2 /mnt
```

Likewise, to mount a floppy that was created on a Windows system and therefore i in DOS format, you use the command :

```
mount -t msdos /dev/fd0 /mnt
```

This makes the files available on an MS-DOS-format floppy under /mnt. Note tha using msdos means that you use the old DOS format that is limited to filenames of 8 plus characters. If you use vfat instead, you get the newer format that was introduced with Windo 95.

There are many options to the mount command, which can be specified with the -o switch For example, the MS-DOS and ISO 9660 filesystems support "autoconversion" of text file from MS-DOS format (which contain CR-LF at the end of each line), to Unix format (whic contains merely a newline at the end of each line). Using a command, such as:

```
mount -o conv=auto -t msdos /dev/fd0 /mnt
```

turns on this conversion for files that don't have a filename extension that could be associate with a binary file (such as .exe, .bin, and so forth).

One common option to mount is -o ro (or, equivalently, -r), which mounts the filesyste as read-only. All write access to such a filesystem is met with a "permission denied" erro Mounting a filesystem as read-only is necessary for media like CD-ROMs that are nonwritabl You can successfully mount a CD-ROM without the -r option, but you'll get the annoyin warning message :

```
mount: block device /dev/cdrom is write-protected, mounting read-onl
```

Use a command, such as :

```
mount -t iso9660 -r /dev/cdrom /mnt
```

instead. This is also necessary if you are trying to mount a floppy that has the write-prote tab in place.

The mount manual page lists all available mounting options. Not all are of immediat interest, but you might have a need for some of them, someday. A useful variant o usingmount is mount -a, which mounts all filesystems listed in /etc/fstab except those marke with the noauto option.

Q.3. What do you mean by UnMounting Filesystem?

Ans. The inverse of mounting a filesystem is, naturally, unmounting it. Unmounting filesystem has two effects: it synchronizes the system's buffers with the actual contents of th filesystem on disk, and it makes the filesystem no longer available from its mount point. Yo are then free to mount another filesystem on that mount point.

Unmounting is done with the umount command (note that the first "n" is missing from th word "unmount"), as in:

```
umount /dev/fd0
```

to unmount the filesystem on /dev/fd0. Similarly, to unmount whatever filesystem is currentl mounted on a particular directory, use a command, such as :

```
umount /mnt
```

Reads and writes to filesystems on floppies are buffered in memory as they are for hard drives. This means that when you read or write data to a floppy, there may not be any immediate drive activity. The system handles I/O on the floppy asynchronously and reads or writes data only when absolutely necessary. So if you copy a small file to a floppy, but the drive light doesn't come on, don't panic; the data will be written eventually. You can use the sync command to force the system to write all filesystem buffers to disk, causing a physical write of any buffered data. Unmounting a filesystem makes this happen as well.

If you wish to allow mortal users to mount and unmount certain devices, you have two options. The first option is to include the user option for the device in /etc/fstab. This allows any user to use the mount and umount command for a given device. Another option is to use one of the mount frontends available for Linux. These programs run setuid root and allow ordinary users to mount certain devices. In general, you wouldn't want normal users mounting and unmounting a hard-drive partition, but you could be more lenient about the use of CD-ROM and floppy drives on your system.

Q.4. How can we create Partitions of disk in Linux?

Ans. Linux partitions can be created with the fdisk command. In general, you need to create at least one partition for the Linux software itself and another partition for swap space.

Many distributions nowadays provide a more user-friendly interface to fdisk. While those are usually not as flexible as plain fdisk, they can help you make the right choices more easily. The tools all do more or less the same things in the end; some simply have more sugar-coating than others. You can also make use of the information presented here for fixing or checking something that you suspect didn't go right with the graphical tool.

After booting the installation medium, run fdisk by typing :

```
fdisk drive
```

where drive is the Linux device name of the drive to which you plan to add partitions. For instance, if you want to run fdisk on the first SCSI disk in your system, use the command:

```
# fdisk /dev/sda
```

/dev/hda (the first IDE drive) is the default if you don't specify one.

If you are creating Linux partitions on more than one drive, run fdisk once for each drive:

```
# fdisk /dev/hda
```

```
Command (m for help):
```

Here fdisk is waiting for a command; you can type m to get a list of options :

```
Command (m for help): m
Command action
   a toggle a bootable flag
   d delete a partition
   l list known partition types
   m print this menu
   n add a new partition
   p print the partition table
   q quit without saving changes
   t change a partition's system id
```

```
u change display/entry units
v verify the partition table
w write table to disk and exit
x extra functionality (experts only)
```

Command (m for help) : The n command is used to create a new partition. Most other options you won't need to worry about. To quit fdisk without saving any changes, use the q command. To quitfdisk and write the changes to the partition table to disk, use the w command. This is worth repeating: so long as you quit with q without writing, you can mess around as much as you want with fdisk without risking harm to your data. Only when you type w can you cause potential disaster to your data if you do something wrong.

The first thing you should do is display your current partition table and write the information down for later reference. Use the p command to see the information. It is a good idea to copy the information to your notebook after each change you have made to the partition table. If, for some reason, your partition table is damaged, you will not access any data on your hard disk any longer, even though the data itself is still there. But by using your notes, you might be able to restore the partition table and get your data back in many cases by running fdisk again and deleting and re-creating the partitions with the parameters you previously wrote down. Don't forget to save the restored partition table when you are done.

Here is an example of a printed partition table, where blocks, sectors, and cylinders are units into which a hard disk is organized :

```
Command (m for help): p

Disk /dev/hda: 16 heads, 38 sectors, 683 cylinders
Units = cylinders of 608 * 512 bytes
 Device Boot Begin Start End Blocks Id System
/dev/hda1 * 1 1 203 61693 6 DOS 16-bit >=32M

Command (m for help):
```

In this example, we have a single Windows partition on /dev/hda1, which is 61693 blocks (about 60 MB). This partition starts at cylinder number 1 and ends on cylinder 203. We have a total of 683 cylinders in this disk; so there are 480 cylinders left on which to create Linux partitions.

To create a new partition, use the n command. In this example, we'll create two primary partitions (/dev/hda2 and /dev/hda3) for Linux :

```
Command (m for help): n
Command action
 e extended
 p primary partition (1-4)
 p
```

Here, fdisk is asking which type of the partition to create: extended or primary. In our example, we're creating only primary partitions, so we choose p :

```
Partition number (1-4):
```

fdisk will then ask for the number of the partition to create; because partition 1 is already used, our first Linux partition will be number 2 :

```
Partition number (1-4): 2
```

```
First cylinder (204-683):
```

Now, we enter the starting cylinder number of the partition. Because cylinders 204 through 683 are unused, we'll use the first available one (numbered 204). There's no reason to leave empty space between partitions :

```
First cylinder (204-683): 204
Last cylinder or +size or +sizeM or +sizeK (204-683):
```

fdisk is asking for the size of the partition we want to create. We can either specify an ending cylinder number, or a size in bytes, kilobytes, or megabytes. Because we want our partition to be 80 MB in size, we specify +80M. When specifying a partition size in this way, fdisk will round the actual partition size to the nearest number of cylinders :

```
Last cylinder or +size or +sizeM or +sizeK (204-683): +80M
```

If you see a warning message such as this, it can be ignored. fdisk prints the warning because it's an older program and dates back before the time that Linux partitions were allowed to be larger than 64 MB.

Now we're ready to create our second Linux partition. For sake of demonstration, we'll create it with a size of 10 MB :

```
Command (m for help): n
Command action
   e extended
   p primary partition (1-4)
p
Partition number (1-4): 3
First cylinder (474-683): 474
Last cylinder or +size or +sizeM or +sizeK (474-683): +10M
```

At last, we'll display the partition table. Again, write down all this information — especially the block sizes of your new partitions. You'll need to know the sizes of the partitions when creating filesystems. Also, verify that none of your partitions overlaps :

```
Command (m for help): p

Disk /dev/hda: 16 heads, 38 sectors, 683 cylinders
Units = cylinders of 608 * 512 bytes
   Device Boot Begin Start End Blocks Id System
/dev/hda1 * 1 1 203 61693 6 DOS 16-bit >=32M
/dev/hda2 204 204 473 82080 83 Linux native
/dev/hda3 474 474 507 10336 83 Linux native
```

As you can see, /dev/hda2 is now a partition of size 82080 blocks (which corresponds to about 80 MB), and /dev/hda3 is 10336 blocks (about 10 MB).

Note that most distributions require you to use the t command in fdisk to change the type of the swap partition to "Linux swap," which is numbered 82. You can use the L command to print a list of known partition type codes, and then use the t command to set the type of the swap partition to that which corresponds to "Linux swap."

Finally, we use the w command to write the changes to disk and exit fdisk :

```
Command (m for help): w
```

Keep in mind that none of the changes you make while running fdisk takes effect until you give the w command, so you can toy with different configurations and save them when you're done. Also, if you want to quit fdisk at any time without saving the changes, use the q command. Remember that you shouldn't modify partitions for operating systems other than Linux with the Linux fdisk program.

You may not be able to boot Linux from a partition using cylinders numbered over 1023. Therefore, you should try to create your Linux root partition within the sub-1024 cylinder range, which is almost always possible (e.g., by creating a small root partition in the sub-1024 cylinder range). If, for some reason, you cannot or do not want to do this, you can simply boot Linux from floppy.

Some Linux distributions require you to reboot the system after running fdisk to allow the changes to the partition table to take effect before installing the software. Newer versions of fdisk automatically update the partition information in the kernel, so rebooting isn't necessary. To be on the safe side, after running fdisk you should reboot from the installation medium before proceeding.

Q.5. How can we manage Swap Space in Linux? **[Important]**

Ans. Swap space is a generic term for disk storage used to increase the amount of apparent memory available on the system. Under Linux, swap space is used to implement paging, a process whereby memory pages are written out to disk when physical memory is low and read back into physical memory when needed (a page is 4096 bytes on Intel x86 systems; this value can differ on other architectures). The process by which paging works is rather involved, but it is optimized for certain cases. The virtual memory subsystem under Linux allows memory pages to be shared between running programs.

For example, if you have multiple copies of Emacs running simultaneously, only one copy of the Emacs code is actually in memory. Also, text pages (those pages containing program code, not data) are usually read-only, and therefore not written to disk when swapped out. Those pages are instead freed directly from main memory and read from the original executable file when they are accessed again.

Of course, swap space cannot completely make up for a lack of physical RAM. Disk access is much slower than RAM access, by several orders of magnitude. Therefore, swap is useful primarily as a means to run a number of programs simultaneously that would not otherwise fit into physical RAM; if you are switching between these programs rapidly you'll notice a lag as pages are swapped to and from disk.

At any rate, Linux supports swap space in two forms: as a separate disk partition or a file somewhere on your existing Linux filesystems. You can have up to eight swap areas, with each swap area being a disk file or partition up to 2 GB in size.

Note that using a swap partition can yield better performance because the disk blocks are guaranteed to be contiguous. In the case of a swap file, however, the disk blocks may be scattered around the filesystem, which can be a serious performance hit in some cases. Many people use a swap file when they must add additional swap space temporarily — for example, if the system is thrashing because of lack of physical RAM and swap. Swap files are a good way to add swap on demand.

Nearly all Linux systems utilize swap space of some kind — usually a single swap partition.

The free command reports information on system-memory usage :

```
rutabaga% free
```

```
total used free shared buffers cached
Mem: 127888 126744 1144 27640 1884 51988
-/+ buffers/cache: 72872 55016
Swap: 130748 23916 106832
```

All the numbers here are reported in 1024-byte blocks. Here, we see a system with 127,888 blocks (about 127 MB) of physical RAM, with 126,744 (about 126 MB) currently in use. Note that your system actually has more physical RAM than that given in the "total" column; this number does not include the memory used by the kernel for its own sundry needs.

The "shared" column lists the amount of physical memory shared between multiple processes. Here, we see that about 27 MB of pages are being shared, which means that memory is being utilized well. The "buffers" column shows the amount of memory being used by the kernel buffer cache. The buffer cache is used to speed up disk operations by allowing disk reads and writes to be serviced directly from memory. The buffer cache size will increase or decrease as memory usage on the system changes; this memory is reclaimed if applications need it. Therefore, although we see that 126 MB of system memory is in use, not all (but most) of it is being used by application programs. The "cache" column indicates how many memory pages the kernel has cached for faster access later.

Because the memory used for buffers and cache can easily be reclaimed for use by applications, the second line (-/+ buffers/cache) provides an indication of the memory actually used by applications (the "used" column) or available to applications (the "free" column). The sum of the memory used by buffers and cache reported in the first line is subtracted from the total used memory and added to the total free memory to give the two figures on the second line.

In the third line, we see the total amount of swap, 130,748 blocks (about 128 MB). In this case, only very little of the swap is being used; there is plenty of physical RAM available. If additional applications were started, larger parts of the buffer cache memory would be used to host them. Swap space is generally used as a last resort when the system can't reclaim physical memory in other ways.

Note that the amount of swap reported by free is somewhat less than the total size of your swap partitions and files. This is because several blocks of each swap area must be used to store a map of how each page in the swap area is being utilized. This overhead should be rather small; only a few kilobytes per swap area.

If you're considering creating a swap file, the df command gives you information on the amount of space remaining on your various filesystems. This command prints a list of filesystems, showing each one's size and what percentage is currently occupied.

Q.6. What do you mean by device files?

Ans. Device files allow user programs to access hardware devices on the system through the kernel. They are not "files" per se, but look like files from the program's point of view: you can read from them, write to them, mmap() onto them, and so forth. When you access such a device "file," the kernel recognizes the I/O request and passes it a device driver, which performs some operation, such as reading data from a serial port or sending data to a sound card.

Device files provide a convenient way to access system resources without requiring the applications programmer to know how the underlying device works. Under Linux, as with most Unix systems, device drivers themselves are part of the kernel.

Device files are located in the directory /dev on nearly all Unix-like systems. Each device on the system should have a corresponding entry in /dev. For example, /dev/ttyS0 corresponds

to the first serial port, known as COM1 under MS-DOS; /dev/hda2 corresponds to the second partition on the first IDE drive. In fact, there should be entries in /dev for devices you do not have. The device files are generally created during system installation and include every possible device driver. They don't necessarily correspond to the actual hardware on your system.

Q.7. How can we create Filesystem in Unix? **[Important]**

Ans. You can create a filesystem using the mkfs command. Creating a filesystem is analogous to "formatting" a partition or floppy, allowing it to store files.

Each filesystem type has its own mkfs command associated with it — for example; MS-DOS filesystem may be created using mkfs.msdos, Second Extended filesystem using mkfs.ext2, and so on. The program mkfs itself is a frontend that creates a filesystem of any type by executing the appropriate version of mkfs for that type.

Under Linux the mkfs command historically created a Minix filesystem. On newer Linux systems, mkfs is a frontend for any filesystem type, and Minix filesystems are created using mkfs.minix.

When you installed Linux, you may have created filesystems by hand using a command such as mke2fs. In fact, mke2fs is equivalent to mkfs.ext2. The programs are the same, but the mkfs.fs-type filename makes it easier for mkfs to execute the appropriate filesystem-type-specific program. If you don't have the mkfs frontend, you can use mke2fs or mkfs.ext2 directly.

Assuming that you're using the mkfs frontend, you can create a filesystem using this command :

```
mkfs -t type device
```

where type is the type of filesystem to create and device is the device on which to create the filesystem (such as /dev/fd0 for a floppy).

For example, to create an ext2 filesystem on a floppy, you use this command :

```
mkfs -t ext2 /dev/fd0
```

You could create an MS-DOS floppy using -t msdos instead.

We can now mount the floppy copy files to it and so forth. Remember to unmount the floppy before removing it from the drive.

Creating a filesystem deletes all data on the corresponding physical device (floppy, hard-drive partition, whatever). mkfs usually does not prompt you before creating a filesystem, so be absolutely sure you know what you're doing.

Creating a filesystem on a hard-drive partition is done exactly as shown earlier, except that you would use the partition name, such as /dev/hda2, as the device. Don't try to create a filesystem on a device, such as /dev/hda. This refers to the entire drive, not just a single partition on the drive.

You should be especially careful when creating filesystems on hard-drive partitions. Be absolutely sure that the device and size arguments are correct. If you enter the wrong device, you could end up destroying the data on your current filesystems, and if you specify the wrong size, you could overwrite data on other partitions. Be sure that size corresponds to the partition size as reported by Linux fdisk.

When creating filesystems on floppies, it's usually best to do a low-level format first. This lays down the sector and track information on the floppy so that its size can be automatically detected using the devices /dev/fd0 or /dev/fd1. One way to do a low-level format is with the MS-DOS FORMAT command; another way is with the Linux program fdformat. For example, to format the floppy in the first floppy drive, use the command:

```
rutabaga# fdformat /dev/fd0
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
```

Using the -n option with fdformat will skip the verification step.

Each filesystem-specific version of mkfs supports several options you might find useful. Most types support the -c option, which causes the physical media to be checked for bad blocks while creating the filesystem. If bad blocks are found, they are marked and avoided when writing data to the filesystem. In order to use these type-specific options, include them after the -t type option to mkfs, as follows :

```
mkfs -t type -c device blocks
```

You may not have all available type-specific versions of mkfs installed. If this is the case, mkfs will fail when you try to create a filesystem of a type for which you have nomkfs.type. Many filesystem types supported by Linux have a corresponding mkfs.type available, somewhere.

If you run into trouble using mkfs, it's possible that Linux is having problems accessing the physical device. In the case of a floppy, this might just mean a bad floppy. In the case of a hard drive, it could be more serious; for example, the disk device driver in the kernel might be having problems reading your drive. This could be a hardware problem or a simple matter of your drive geometry being specified incorrectly.

Q.8. Define Raw and Block Files in Unix. **[Important]**

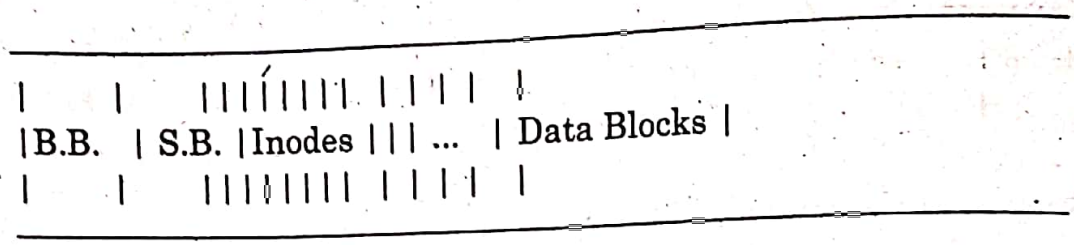
Ans. In the original Unix file system, Unix divided physical disks into logical disks called *partitions*. Each partition is a standalone file system. We will use the term "file system" when referring to a single partition.

Each disk device is given its own *major device number*, and each partition has an associated *minor device number* which the device driver uses to access the raw file system.

The major/minor device number combination serves as a handle into the device switch table. That is, the major number acts as an index, and the minor number is passed as an argument to the driver routines so that they can recognize the specific instance of a device.

Each filesystem contains :

1. a *boot block* located in the first few sectors of a file system. The boot block contains the initial bootstrap program used to load the operating system. Typically, the first sector contains a bootstrap program that reads in a larger bootstrap program from the next few sectors, and so forth.
2. a *super block* describes the state of the file system: the total size of the partition, the block size, pointers to a list of free blocks, the inode number of the root directory, magic number, etc.
3. a linear array of *inodes* (short for "index nodes"). There is a one to one mapping of files to inodes and vice versa. An inode is identified by its "inode number", which contains the information needed to find the inode itself on the disk. Thus, while users think of files in terms of file names, Unix thinks of files in terms of inodes.
4. *data blocks* blocks containing the actual contents of files.



Q.9. What do you mean by 'inode' in file system of Unix?

Ans. An inode is the "handle" to a file and contains the following information :

- file ownership indication
- file type (e.g., regular, directory, special device, pipes, etc.)
- file access permissions. May have setuid (sticky) bit set.
- time of last access, and modification
- number of links (aliases) to the file
- pointers to the data blocks for the file
- size of the file in bytes (for regular files), major and minor device numbers for special devices.

An integral number of inodes fits in a single data block.

Information the inode does not contain :

- path (short or full) name of file

Example :

```
Look at /cs/bin/
< wpi /cs/bin 1 >ls -l
total 192
drwx—— 2 mvoorhis csadmin 4096 Jan 16 2001 archives/
-rws-x— 1 root 771 32768 Jan 18 1999 csquotamgr*
-rwx—— 1 csadmin csadmin 162 Jan 12 1998 genQuota*
-rwx—— 1 csadmin csadmin 46 Feb 16 1998 generic*
drwxrwx— 2 mvoorhis csadmin 4096 Oct 29 10:23 gredStuff/
-rwx—— 1 mvoorhis 1067 672 Jan 20 2000 list1*
-rwx—— 1 mvoorhis 1067 859 Jan 20 2000 list2*
-rwx—— 1 csadmin 646 140 Jan 10 2000 reclaim*
-rwxrwx— 1 csadmin csadmin 1635 Sep 26.1995 stp_create_system.pl*
-rwxrwxr-x 1 csadmin csadmin 725 Sep 26 1995 stp_default_system.pl*
-rw-rw-r— 1 csadmin csadmin 114 Feb 10 1995 stp_setup
drwx—— 14 mvoorhis csadmin 4096 Oct 30 14:57 tDir/
-rwsr-xr-x 1 mvoorhis 1067 114688 Nov 8 10:10 turnin*
drwxr-xr-x 2 root 771 4096 May 26 1999 utility/
```

Internally, Unix stores directories in files. The file type (of the inode) is marked "directory", and the file contains pairs of name/inode numbers.

For example, when a user issues *open("/etc/passwd", ...)* the kernel performs the following operations :

1. because the file name is a full path name, find the inode of the root directory (found in superblock) and search the corresponding file for the entry "etc"

2. when the entry "etc" is found, fetch its corresponding inode and check that it is of type directory.
3. scan the file associated with "/etc" looking for "passwd"
4. finally, fetch the inode associated with *passwd*'s directory entry, verify that it is a regular file, and start accessing the file.

Eventually, the system would find the inode corresponding to the device, and note that its file type was "special". Thus, it would extract the major/minor device number pair from the length field of the inode, and use the device number as an index into the device switch table.

Q.10. What do you mean by formatting? How formatting is done in Unix?

Ans. Formatting is the process of writing marks on the magnetic media that are used to mark tracks and sectors. Before a disk is formatted, its magnetic surface is a complete mess of magnetic signals. When it is formatted, some order is brought into the chaos by essentially drawing lines where the tracks go, and where they are divided into sectors. The actual details are not quite exactly like this, but that is irrelevant. What is important is that a disk cannot be used unless it has been formatted.

The terminology is a bit confusing here: in MS-DOS and MS Windows, the word formatting is used to cover also the process of creating a filesystem (which will be discussed below). There, the two processes are often combined, especially for floppies. When the distinction needs to be made, the real formatting is called low-level formatting, while making the filesystem is called high-level formatting . In UNIX circles, the two are called formatting and making a filesystem, so that's what is used in this book as well.

For IDE and some SCSI disks the formatting is actually done at the factory and doesn't need to be repeated; hence most people rarely need to worry about it. In fact, formatting a hard disk can cause it to work less well, for example because a disk might need to be formatted in some very special way to allow automatic bad sector replacement to work.

Disks that need to be or can be formatted often require a special program anyway, because the interface to the formatting logic inside the drive is different from drive to drive. The formatting program is often either on the controller BIOS, or is supplied as an MS-DOS program; neither of these can easily be used from within Linux.

During formatting one might encounter bad spots on the disk, called bad blocks or bad sectors. These are sometimes handled by the drive itself, but even then, if more of them develop, something needs to be done to avoid using those parts of the disk. The logic to do this is built into the filesystem; how to add the information into the filesystem is described below. Alternatively, one might create a small partition that covers just the bad part of the disk; this approach might be a good idea if the bad spot is very large, since filesystems can sometimes have trouble with very large bad areas.

Floppies are formatted with fdformat . The floppy device file to use is given as the parameter. For example, the following command would format a high density, 3.5 inch floppy in the first floppy drive:

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity
1440 kB.
Formatting ... done
Verifying ... done
$
```

Note that if you want to use an autodetecting device (e.g., /dev/fd0), you must set the parameters of the device with setfdprm first. To achieve the same effect as above, one would have to do the following :

```
$ setfdprm /dev/fd0 1440/1440
$ fdformat /dev/fd0
Double-sided, 80 tracks, 18 sec/track. Total capacity
1440 KB.
Formatting ... done
Verifying ... done
$
```

It is usually more convenient to choose the correct device file that matches the type of the floppy. Note that it is unwise to format floppies to contain more information than what they are designed for.

fdformat also validates the floppy, i.e., check it for bad blocks. It will try a bad block several times (you can usually hear this, the drive noise changes dramatically). If the floppy is only marginally bad (due to dirt on the read/write head, some errors are false signals), fdformat won't complain, but a real error will abort the validation process. The kernel will print log messages for each I/O error it finds; these will go to the console or, ifsyslog is being used, to the file /var/log/messages, fdformat itself won't tell where the error is (one usually doesn't care, floppies are cheap enough that a bad one is automatically thrown away).

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity
1440 KB.
Formatting ... done
Verifying ... read: Unknown error
$
```

The badblocks command can be used to search any disk or partition for bad blocks (including a floppy). It does not format the disk, so it can be used to check even existing filesystems. The example below checks a 3.5 inch floppy with two bad blocks.

```
$ badblocks /dev/fd0H1440 1440
718
719
$
```

badblocks outputs the block numbers of the bad blocks it finds. Most filesystems can avoid such bad blocks. They maintain a list of known bad blocks, which is initialized when the filesystem is made, and can be modified later. The initial search for bad blocks can be done by the mkfs command (which initializes the filesystem), but later checks should be done with badblocks and the new blocks should be added with fsck.

Many modern disks automatically notice bad blocks, and attempt to fix them by using a special, reserved good block instead. This is invisible to the operating system. This feature should be documented in the disk's manual, if you're curious if it is happening. Even such disks can fail, if the number of bad blocks grows too large, although chances are that by then the disk will be so rotten as to be unusable.

Q.11. What do you mean by logical volume? What are different types of Logical volume in Linux? [Important]

Ans. A logical volume lives in a volume group that is made up of one or more physical volumes. All volume groups are part of the Logical Volume Manager. Here is a table that lists the three types of Logical Volume's.

Logical Volume Types		
Logical Volume	**Physical Device**	**File System**
volume group	one or more disks	vgdisplay
physical volume group	physical extents on a drive	pvdisplay
logical volume group	multiple physical volume groups, one or more disks	lvdisplay

The following table provides an overview of some of the commands used in LVM and the functions they service.

LVM Commands	
Command	**LVM Function**
vgcreate	Create a Volume Group. *(Create a subset of the overall LVM)*
pvcreate	Create a Physical Volume, assign to Volume Group. *(Specify a disk for inclusion in the overall LVM)*
vgextend	Add a new physical disk to a volume group.
lvcreate	Create a Logical Volume. *(Storage area for related files that is part of a Volume Group. A Volume Group consists of many logical volumes)*
lvextend	Increase the size of a Logical Volume.
lvreduce	Decrease the size of a Logical Volume.
lvremove	Removes a Logical Volume. *(Frees the storage area set aside for a logical volume)*
vgreduce	Reduce a Volume Group. *(Reduces the number of disks in a Volume Group)*
vgremove	Remove a Volume Group. *(Removes the designation of a group of disks as a Volume Group)*
vgdisplay	Volume Group Display. *(Displays information about one or more Volume Groups)*
pvdisplay	Physical Volume Display. *(Displays information about one or more Volume Groups)*
lvdisplay	Logical Volume Display. *(Displays information about one or more Logical Volumes)*

Q.12. How we can create logical volumes in Linux system?

Ans. The following subsections describe the commands used for creating logical volumes.

Partition Types: Before using a hard disk as a physical volume, decide if the physical volume will use the entire disk (/dev/sdc) or a disk partition (/dev/sdc1).

To create the physical volume using a partition, set the partition type to 0x8e (Linux LVM) using fdisk or some other similar program.

Each logical volume will use the entire hard disk. This requires that no partition table exists on the disk. When using the whole disk, the partition table must be erased, which will effectively destroy all data on that disk. An existing partition table can be removed by zeroing the first sector on the disk using the dd command.

```
[root@testnode1 ~]# dd if=/dev/zero of=/dev/sdc bs=512 count=1
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.00379734 seconds, 135 kB/s

[root@testnode1 ~]# dd if=/dev/zero of=/dev/sdd bs=512 count=1
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.00200099 seconds, 256 kB/s
```

Verify that the partition table has been removed from both hard disks.

```
[root@testnode1 ~]# fdisk -l | grep '^Disk \/dev\/sd[cd][^:]'
Disk /dev/sdc doesn't contain a valid partition table
Disk /dev/sdd doesn't contain a valid partition table
Disk /dev/dm-0 doesn't contain a valid partition table
Disk /dev/dm-1 doesn't contain a valid partition table
```

Initialize Physical Volumes : Use the pvcreate command to initialize a block device t be used as a physical volume. The following commands will initialize the whole disk for eac hard disk.

```
[root@testnode1 ~]# pvcreate /dev/sdc
  Physical volume "/dev/sdc" successfully created

[root@testnode1 ~]# pvcreate /dev/sdd
  Physical volume "/dev/sdd" successfully created
```

When using a partition, run pvcreate on the partition.

```
[root@testnode1 ~]# pvcreate /dev/sdc1
  Physical volume "/dev/sdc1" successfully created

[root@testnode1 ~]# pvcreate /dev/sdd1
  Physical volume "/dev/sdd1" successfully created
```

This creates a volume group descriptor at the start of the /de sdc1 and /dev/sdd1 partition.

Use the lvmdiskscan command to scan for block devices and verify that the two hard di can be used as physical volumes.

```
[root@testnode1 ~]# lvmdiskscan | grep '\/dev\/sd[cd]'
  /dev/sdc     [36.00 GB] LVM physical volume
  /dev/sdd     [36.00 GB] LVM physical volume
```

Create Volume Group : Use the vgcreate command to create a volume group from on more physical volumes. The vgcreate command creates a new volume group by na (vg_oradata and vg_orafra for example) and adds at least one physical volume to it. Creat new volume group on each hard disk.

```
[root@testnode1 ~]# vgcreate vg_oradata /dev/sdc
```

```
  Volume group "vg_oradata" successfully created

[root@testnode1 ~]# vgcreate vg_orafra /dev/sdd
  Volume group "vg_orafra" successfully created
```

This creates a volume group descriptor at the start of each disk. When using partitions, run the vgcreate command on the partition (for example, vgcreate vg_oradata /dev/sdc1) which will create a volume group descriptor at the start of the partition.

Create Logical Volumes : With the new volume groups in place, use the lvcreate command to create the appropriate logical volumes (lv_oradata and lv_orafra). Logical volumes can be created as linear volumes, striped volumes, and mirrored volumes. For the purpose of this example, create a single linear volume within each of the volume groups that uses all of the unallocated space within the volume group.

```
[root@testnode1 ~]# lvcreate -l 100%FREE -n lv_oradata vg_oradata
  Logical volume "lv_oradata" created

[root@testnode1 ~]# lvcreate -l 100%FREE -n lv_orafra vg_orafra
  Logical volume "lv_orafra" created
```

Q.13. What do you mean by Network File System (NFS)? **[Important]**

Ans. Network File System (NFS) is a distributed file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a network in a manner similar to how local storage is accessed. NFS, like many other protocols, builds on the Open Network Computing Remote Procedure Call (ONC RPC) system. The Network File System is an open standard defined in RFCs, allowing anyone to implement the protocol.

Using NFS, clients can mount partitions of a server as if they were physically connected to the client. In addition to simply allowing remote access to files over the network,NFS allows many (relatively) low-cost computer systems to share the same high-capacity disk drive at the same time. NFS server programs have been written for many different operating systems, which let users on UNIX workstations have remote access to files stored on a variety of different platforms. NFS clients have been written for microcomputers such as the IBM/PC and Apple Macintosh, giving PC users much of the same flexibility enjoyed by their UNIX coworkers, as well as a relatively easy method of data interchange.

NFS is nearly transparent. In practice, a workstation user simply logs into the workstation and begins working, accessing it as if the files were locally stored. In many environments, workstations are set up to mount the disks on the server automatically at boot time or when files on the disk are first referenced. NFS also has a network mounting program that can be configured to mount the NFS disk automatically when an attempt is made to access files stored on remote disks.

There are several basic security problems with NFS :

- NFS is built on top of Sun's RPC (Remote Procedure Call), and in most cases uses RPC for user authentication. Unless a secure form of RPC is used, NFS can be easily spoofed.
- Even when Secure RPC is used, information sent by NFS over the network is not encrypted, and is thus subject to monitoring and eavesdropping. As we mention elsewhere, the data can be intercepted and replaced (thereby corrupting or Trojaning files being imported via NFS).

- NFS uses the standard UNIX filesystem for access control, opening the networked filesystem to many of the same problems as a local filesystem.

Q.14. What are the different methods of creating a Boot Disk? **[Important]**

Ans. 1. Creating a Boot Disk in DOS : You need formatted 3.5" HD floppy disk and bootable 3.5" floppy disk drive. The boot directory on CD 1 contains a number of disk image With a suitable utility, these images can be copied to floppy disks. A floppy disk prepared i this way is referred to as a boot disk.

The disk images also include the loader SYSLINUX and the program linuxrc. SYSLINU enables the selection of a kernel during the boot procedure and the specification of an parameters needed for the hardware used. The program linuxrc supports the loading of kerne modules for your hardware and subsequently starts the installation.

2. Creating a Boot Disk with rawwritewin : In Windows, boot disks can be create with the graphical utility rawwritewin. In Windows, find this utility in the directory dosutils rawwritewin on CD 1.

On start-up, specify the image file. The image files are located in the boot directory o CD 1. At the least, you will need the images "bootdisk" and "modules1". To list these images i the file browser, set the file type to "all files". Then insert a floppy disk in your floppy dis drive and click "write". To create several floppy disks, repeat the same procedure.

3. Creating a Boot Disk with rawrite : The DOS utility rawrite.exe (CD 1 directory dosutils/rawrite can be used for creating SUSE boot and module disks. To use thi utility, you need a computer with DOS (such as FreeDOS) or Windows.

In Windows XP, proceed as follows :

1. Insert SUSE LINUX CD 1.
2. Open a DOS window (in the start menu, select 'Accessories' ->'Command Prompt'),
3. Run rawrite.exe with the correct path specification for the CD drive. The exampl assumes that you are in the directory Windows on the hard disk C: and your CD driv is D:.

   ```
   d:\dosutils\rawrite\rawrite
   ```

4. On start-up, the utility asks for the source and destination of the file to copy. Th image of the boot disk is located in the directory boot on CD 1. The file name is bootdisk Remember to specify the path for your CD drive.
5. d:\dosutils\rawrite\rawrite
6. RaWrite 1.2 - Write disk file to raw floppy diskette
7. Enter source file name: d:\boot\bootdisk
   Enter destination drive: a:

After you enter the destination drive a:, rawrite prompts you to insert a formatted floppy disk and press Enter. Subsequently, the progress of the copy action is displayed. The proces can be terminated withCtrl + C.

The other disk images (modules1, modules2, modules3, and modules4) can be created in the same way. These floppy disks are required if you have USB or SCSI devices or a network or PCMCIA card that you want to address during the installation. A module disk may also be needed if using a special file system during the installation.

4. Creating a Boot Disk in a UNIX-Type System : On a UNIX or Linux system, you need a CD-ROM drive and a formatted floppy disk. Proceed as follows to create boot disks:

1. If you need to format the disks first, use :
   `fdformat /dev/fd0u1440`
2. Mount CD 1 (for example, to /media/cdrom):
   `mount -tiso9660 /dev/cdrom /media/cdrom`
3. Change to the boot directory on the CD:
   `cd /media/cdrom/boot`
4. Create the boot disk with the following command:

`dd if=/media/cdrom/boot/bootdisk of=/dev/fd0 bs=8k`

The README file in the boot directory provides details about the floppy disk images. You can read these files with more or less.

The other disk images, modules1, modules2, modules3, and modules4, can be created in the same way. These floppy disks are required if you have USB or SCSI devices or a network or PCMCIA card that you want to address during the installation. A module disk may also be needed to use a special file system during the installation.

To use a custom kernel during the installation, the procedure is a bit more complex. In this case, write the default image bootdisk to the floppy disk then overwrite the kernel linux with your own kernel :

```
dd if=/media/cdrom/boot/bootdisk of=/dev/fd0 bs=8k
mount -t msdos /dev/fd0 /mnt
cp /usr/src/linux/arch/i386/boot/vmlinuz /mnt/linux
umount /mnt
```

5. Booting from a Floppy Disk (SYSLINUX) : The boot disk can be used for handling special installation requirements (for example, if the CD-ROM drive is not available).

The boot procedure is initiated by the boot loader SYSLINUX (syslinux). When the system is booted, SYSLINUX runs a minimum hardware detection that mainly consists of the following steps :

1. The program checks if the BIOS provides VESA 2.0–compliant framebuffer supports and boots the kernel accordingly.
2. The monitor data (DDC info) is read.
3. The first block of the first hard disk (MBR) is read to map BIOS IDs to Linux device names during the boot loader configuration. The program attempts to read the block by means of the the lba32 functions of the BIOS to determine if the BIOS supports these functions.

Q.15. How can we mount the /proc Filesystem in Linux system?

Ans. Some of the configuration tools of the Linux NET-2 and NET-3 release rely on the */proc* filesystem for communicating with the kernel. This interface permits access to kernel runtime information through a filesystem-like mechanism. When mounted, you can list its files like any other filesystem, or display their contents. Typical items include the *loadavg* file, which contains the system load average, and *meminfo*, which shows current core memory and swap usage.

To this, the networking code adds the *net* directory. It contains a number of files that show things like the kernel ARP tables, the state of TCP connections, and the routing tables. Most network administration tools get their information from these files.

The *proc* filesystem (or *procfs*, as it is also known) is usually mounted on */proc* at system boot time. The best method is to add the following line to */etc/fstab* :

```
# procfs mount point:
none   /proc    proc   defaults
```

Then execute mount /proc from your */etc/rc* script.

The *procfs* is now configured into most kernels by default. If the *procfs* is not kernel, you will get a message such as: mount: fs type procfs not supported by kernel. then have to recompile the kernel and answer "yes" when asked for *procfs* support.

Q.16. How can we install the Binaries in Linux System?

Ans. If you are using one of the prepackaged Linux distributions, it will contain the networking applications and utilities along with a coherent set of sample files. The on in which you might have to obtain and install new utilities is when you install a new release. As they occasionally involve changes in the kernel networking layer, you will update the basic configuration tools. This update at least involves recompiling, but som you may also be required to obtain the latest set of binaries. These binaries are call *tools-XXX.tar.gz*, where *XXX* is the version number. The release matching Linux 2.0 *tools-1.45*.

If you want to compile and install the standard TCP/IP network applications yourse can obtain the sources from most Linux FTP servers. All modern Linux distributions i a fairly comprehensive range of TCP/IP network applications, such as World Wid browsers, telnet and ftp programs, and other network applications, such as talk. If you something that you do need to compile yourself, the chances are good that it will c under Linux from source quite simply if you follow the instructions included in the package.

Q.17. Write the methods for setting the Hostname in Unix. [Impor

Ans. Most, if not all, network applications rely on you to set the local host's name to reasonable value. This setting is usually made during the boot procedure by executin hostname command. To set the hostname to *name*, enter :

```
# hostname name
```

It is common practice to use the unqualified hostname without specifying the do name. For instance, hosts at the Virtual Brewery might be called *vale.vbrew.co vlager.vbrew.com*. These are their official *fully qualified domain names* (FQDNs). Their hostnames would be the first component of the name, such as *vale*. However, as the hostname is frequently used to look up the host's IP address, you have to make sure tha resolver library is able to look up the host's IP address. This usually means that you ha enter the name in */etc/hosts*.

Some people suggest using the domainname command to set the kernel's idea of a do name to the remaining part of the FQDN. This way you could combine the output from hostn and domainname to get the FQDN again. However, this is at best only half correct. domainn is generally used to set the host's NIS domain, which may be entirely different from the I domain to which your host belongs. Instead, to ensure that the short form of your hostnam resolvable with all recent versions of the hostname command, either add it as an entry in y local Domain Name Server or place the fully qualified domain name in the */etc/hosts* file. may then use the —*fqdn* argument to the hostname command, and it will print the fully quali domain name.

Q.18. What do you mean by assigning IP Addresses in Unix system?

Ans. If you configure the networking software on your host for standalone operation (for instance, to be able to run the INN Netnews software), you can safely skip this section, because the only IP address you will need is for the loopback interface, which is always *127.0.0.1*.

Things are a little more complicated with real networks like Ethernets. If you want to connect your host to an existing network, you have to ask its administrators to give you an IP address on this network. When setting up a network all by yourself, you have to assign IP addresses yourself.

Hosts within a local network should usually share addresses from the same logical IP network. Hence, you have to assign an IP network address. If you have several physical networks, you have to either assign them different network numbers, or use subnetting to split your IP address range into several subnetworks.

When picking an IP network number, much depends on whether you intend to get on the Internet in the near future. If so, you should obtain an official IP address *now*. Ask your network service provider to help you. If you want to obtain a network number, just in case you might get on the Internet someday, request a Network Address Application Form from hostmaster@internic.net, or your country's own Network Information Center, if there is one.

If your network is not connected to the Internet and won't be in the near future, you are free to choose any legal network address. Just make sure no packets from your internal network escape to the real Internet. To make sure no harm can be done even if packets *did* escape, you should use one of the network numbers reserved for private use. The Internet Assigned Numbers Authority (IANA) has set aside several network numbers from classes A, B, and C that you can use without registering. These addresses are valid only within your private network and are not routed between real Internet sites. Note that the second and third blocks contain 16 and 256 networks, respectively.

Picking your addresses from one of these network numbers is not only useful for networks completely unconnected to the Internet; you can still implement a slightly more restricted access using a single host as a gateway. To your local network, the gateway is accessible by its internal IP address, while the outside world knows it by an officially registered address (assigned to you by your provider). We will assume that the brewery's network manager uses a class B network number, say *172.16.0.0*. Of course, a class C network number would definitely suffice to accommodate both the Brewery's and the Winery's networks. We'll use a class B network here for the sake of simplicity.

Q.19. Write the methods of creating Subnets. **[Important]**

Ans. To operate several Ethernets (or other networks, once a driver is available), you have to split your network into subnets. Note that subnetting is required only if you have more than one *broadcast network* — point-to-point links don't count. For instance, if you have one Ethernet, and one or more SLIP links to the outside world, you don't need to subnet your network.

To accommodate the two Ethernets, the Brewery's network manager decides to use 8 bits of the host part as additional subnet bits. This leaves another 8 bits for the host part, allowing for 254 hosts on each of the subnets. She then assigns subnet number 1 to the brewery, and gives the winery number 2. Their respective network addresses are thus *172.16.1.0* and *172.16.2.0*. The subnet mask is *255.255.255.0*.

vlager, which is the gateway between the two networks, is assigned a host number of 1 on both of them, which gives it the IP addresses *172.16.1.1* and *172.16.2.1*, respectively.

Note that in this example we are using a class B network to keep things simple, but a class C network would be more realistic. With the new networking code, subnetting is not limited to byte boundaries, so even a class C network may be split into several subnets. For instance, you could use two bits of the host part for the netmask, giving you 4 possible subnets with 64 hosts on each.

The first number on each subnet is the subnetwork address, and the last number on each subnet is reserved as the broadcast address, so it's actually 62 hosts per subnet.

Q.20. What do you mean by Writing hosts and networks Files?

Ans. After you have subnetted your network, you should prepare for some simple sort of hostname resolution using the */etc/hosts* file. If you are not going to use DNS or NIS for address resolution, you have to put all hosts in the *hosts* file.

Even if you want to run DNS or NIS during normal operation, you should have some subset of all hostnames in */etc/hosts*. You should have some sort of name resolution, even when no network interfaces are running, for example, during boot time. This is not only a matter of convenience, but it allows you to use symbolic hostnames in your network *rc* scripts. Thus, when changing IP addresses, you only have to copy an updated *hosts* file to all machines and reboot, rather than edit a large number of *rc* files separately. Usually you put all local hostnames and addresses in *hosts*, adding those of any gateways and NIS servers used. You need the address of an NIS server only if you use Peter Eriksson's NYS. Other NIS implementations locate their servers only at runtime by using *ypbind*.

You should make sure your resolver only uses information from the *hosts* file during initial testing. Sample files that come with your DNS or NIS software may produce strange results. To make all applications use */etc/hosts* exclusively when looking up the IP address of a host, you have to edit the */etc/host.conf* file. Comment out any lines that begin with the keyword *order* by preceding them with a hash sign, and insert the line :

```
order hosts
```

The *hosts* file contains one entry per line, consisting of an IP address, a hostname, and an optional list of aliases for the hostname. The fields are separated by spaces or tabs, and the address field must begin in the first column. Anything following a hash sign (#) is regarded as a comment and is ignored.

Hostnames can be either fully qualified or relative to the local domain. For *vale*, you would usually enter the fully qualified name, *vale.vbrew.com*, and *vale* by itself in the *hosts* file, so that it is known by both its official name and the shorter local name.

This is an example how a *hosts* file at the Virtual Brewery might look. Two special names are included, *vlager-if1* and *vlager-if2*, which give the addresses for both interfaces used on *vlager*:

```
#
# Hosts file for Virtual Brewery/Virtual Winery
#
# IP            FQDN                        aliases
127.0.0.1       localhost
#
172.16.1.1      vlager.vbrew.com
172.16.1.2      vstout.vbrew.com            vlager vlager-if1
                                            vstout
```

```
172.16.1.3      vale.vbrew.com          vale
#
172.16.2.1      vlager-if2
172.16.2.2      vbeaujolais.vbrew.com   vbeaujolais
172.16.2.3      vbardolino.vbrew.com    vbardolino
172.16.2.4      vchianti.vbrew.com      vchianti
```

Just as with a host's IP address, you should sometimes use a symbolic name for network numbers, too. Therefore, the *hosts* file has a companion called */etc/networks* that maps network names to network numbers, and vice versa. At the Virtual Brewery, we might install a *networks* file like this :

```
# /etc/networks for the Virtual Brewery
brew-net    172.16.1.0
wine-net    172.16.2.0
```

Note that names in *networks* must not collide with hostnames from the *hosts* file, or else some programs may produce strange results.

Q.21. What is Interface Configuration for IP? Explain various types of Interfaces in detail.

Ans. After setting up your hardware, you have to make these devices known to the kernel networking software. A couple of commands are used to configure the network interfaces and initialize the routing table. These tasks are usually performed from the network initialization script each time you boot the system. The basic tools for this process are called ifconfig (where "if" stands for interface) and route.

ifconfig is used to make an interface accessible to the kernel networking layer. This involves the assignment of an IP address and other parameters, and activation of the interface, also known as "bringing up" the interface. Being active here means that the kernel will send and receive IP datagrams through the interface. The simplest way to invoke it is with :

```
ifconfig interface ip-address
```

This command assigns *ip-address* to *interface* and activates it. All other parameters are set to default values. For instance, the default network mask is derived from the network class of the IP address, such as *255.255.0.0* for a class B address.

route allows you to add or remove routes from the kernel routing table. It can be invoked as :

```
route [add|del] [-net|-host] target [if]
```

The *add* and *del* arguments determine whether to add or delete the route to *target*. The *-net* and *-host* arguments tell the route command whether the target is a network or a host (a host is assumed if you don't specify). The *if* argument is again optional, and allows you to specify to which network interface the route should be directed — the Linux kernel makes a sensible guess if you don't supply this information.

1. **The Loopback Interface :** The very first interface to be activated is the loopback interface :

```
# ifconfig lo 127.0.0.1
```

Occasionally, you will see the dummy hostname *localhost* being used instead of the IP address. ifconfig will look up the name in the *hosts* file, where an entry should declare it as the hostname for *127.0.0.1* :

```
# Sample /etc/hosts entry for localhost
localhost    127.0.0.1
```

To view the configuration of an interface, you invoke ifconfig, giving it only the interface name as argument :

```
$ ifconfig lo
lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        UP LOOPBACK RUNNING MTU:3924 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        Collisions:0
```

As you can see, the loopback interface has been assigned a netmask of *255.0.0.0*, since *127.0.0.1* is a class A address.

Now you can almost start playing with your mini-network. What is still missing is an entry in the routing table that tells IP that it may use this interface as a route to destination *127.0.0.1*. This is accomplished by using :

```
# route add 127.0.0.1
```

Again, you can use *localhost* instead of the IP address, provided you've entered it into your */etc/hosts*.

Next, you should check that everything works fine, for example by using ping. Ping is the networking equivalent of a sonar device. The command is used to verify that a given address is actually reachable, and to measure the delay that occurs when sending a datagram to it and back again. The time required for this process is often referred to as the "round-trip time" :

```
# ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=0.4 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=0.4 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=255 time=0.4 ms
^C
— localhost ping statistics —
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.4/0.4 ms
#
```

2. Ethernet Interfaces : Configuring an Ethernet interface is pretty much the same as the loopback interface; it just requires a few more parameters when you are using subnetting.

At the Virtual Brewery, we have subnetted the IP network, which was originally a class B network, into class C subnetworks. To make the interface recognize this, the ifconfig incantation would look like this :

```
# ifconfig eth0 vstout netmask 255.255.255.0
```

This command assigns the *eth0* interface the IP address of *vstout* (*172.16.1.2*). If we omitted the netmask, ifconfig would deduce the netmask from the IP network class, which would result in an incorrect netmask of *255.255.0.0*. Now a quick check shows:

```
# ifconfig eth0
eth0  Link encap 10Mps Ethernet HWaddr 00:00:C0:90:B3:42
```

```
inet addr 172.16.1.2 Bcast 172.16.1.255 Mask 255.255.255.0
UP BROADCAST RUNNING MTU 1500 Metric 1
RX packets 0 errors 0 dropped 0 overrun 0
TX packets 0 errors 0 dropped 0 overrun 0
```

You can see that ifconfig automatically sets the broadcast address (the *Bcast* field) to the usual value, which is the host's network number with all the host bits set. Also, the maximum transmission unit (the maximum size of IP datagrams the kernel will generate for this interface) has been set to the maximum size of Ethernet packets: 1,500 bytes. The defaults are usually what you will use, but all these values can be overidden if required.

3. Routing Through a Gateway : One encounters networks connected to one another by gateways. These gateways may simply link two or more Ethernets, but may also provide a link to the outside world, such as the Internet. In order to use a gateway, you have to provide additional routing information to the networking layer.

The Ethernets of the Virtual Brewery and the Virtual Winery are linked through such a gateway, namely the host *vlager*. Assuming that *vlager* has already been configured, we just have to add another entry to *vstout*'s routing table that tells the kernel it can reach all hosts on the Winery's network through *vlager*. The appropriate incantation of route is shown below; the *gw* keyword tells it that the next argument denotes a gateway :

```
# route add wine-net gw vlager
```

Of course, any host on the Winery network you wish to talk to must have a routing entry for the Brewery's network. Otherwise you would only be able to send data to the Winery network from the Brewery network, but the hosts on the Winery would be unable to reply.

This example describes only a gateway that switches packets between two isolated Ethernets. Now assume that *vlager* also has a connection to the Internet (say, through an additional SLIP link). Then we would want datagrams to *any* destination network other than the Brewery to be handed to *vlager*. This action can be accomplished by making it the default gateway for *vstout* :

```
# route add default gw vlager
```

The network name *default* is a shorthand for *0.0.0.0*, which denotes the default route. The default route matches every destination and will be used if there is no more specific route that matches. You do not have to add this name to */etc/networks* because it is built into route.

If you see high packet loss rates when pinging a host behind one or more gateways, this may hint at a very congested network. Packet loss is not so much due to technical deficiencies as to temporary excess loads on forwarding hosts, which makes them delay or even drop incoming datagrams.

Configuring a Gateway : Configuring a machine to switch packets between two Ethernets is pretty straightforward. Assume we're back at *vlager*, which is equipped with two Ethernet cards, each connected to one of the two networks. All you have to do is configure both interfaces separately, giving them their respective IP addresses and matching routes, and that's it.

It is quite useful to add information on the two interfaces to the *hosts* file as shown in the following example, so we have handy names for them, too :

```
172.16.1.1      vlager.vbrew.com        vlager vlager-if1
172.16.2.1      vlager-if2
```

The sequence of commands to set up the two interfaces is then :

```
# ifconfig eth0 vlager-if1
```

```
# route add brew-net
# ifconfig eth1 vlager-if2
# route add wine-net
```

If this sequence doesn't work, make sure your kernel has been compiled with support for IP forwarding enabled. One good way to do this is to ensure that the first number on the second line of */proc/net/snmp* is set to 1.

4. The PLIP Interface : A PLIP link used to connect two machines is a little different from an Ethernet. PLIP links are an example of what are called *point-to-point* links, meaning that there is a single host at each end of the link. Networks like Ethernet are called *broadcast* networks. Configuration of point-to-point links is different because unlike broadcast networks, point-to-point links don't support a network of their own.

PLIP provides very cheap and portable links between computers. As an example, we'll consider the laptop computer of an employee at the Virtual Brewery that is connected to *vlager* via PLIP. The laptop itself is called *vlite* and has only one parallel port. At boot time, this port will be registered as *plip1*. To activate the link, you have to configure the *plip1* interface using the following commands :

```
# ifconfig plip1 vlite pointopoint vlager
# route add default gw vlager
```

Note that **pointopoint** is not a typo. It's really spelled like this.

The first command configures the interface, telling the kernel that this is a point-to-point link, with the remote side having the address of *vlager*. The second installs the default route, using *vlager* as gateway. On *vlager*, a similar ifconfig command is necessary to activate the link (a route invocation is not needed) :

```
# ifconfig plip1 vlager pointopoint vlite
```

Note that the *plip1* interface on *vlager* does not need a separate IP address, but may also be given the address *172.16.1.1*. Point-to-point networks don't support a network directly, so the interfaces don't require an address on any supported network. The kernel uses the interface information in the routing table to avoid any possible confusion.

Q.22. What is IP Alias? What is its main use?

Ans. New kernels support a feature that can completely replace the dummy interface and serve other useful functions. *IP Alias* allows you to configure multiple IP addresses onto a physical device. In the simplest case, you could replicate the function of the dummy interface by configuring the host address as an alias onto the loopback interface and completely avoid using the dummy interface. In more complex uses, you could configure your host to look like many different hosts, each with its own IP address. This configuration is sometimes called "Virtual Hosting," although technically it is also used for a variety of other techniques.

More correctly, using IP aliasing is known as network layer virtual hosting. It is more common in the WWW and STMP worlds to use application layer virtual hosting, in which the same IP address is used for each virtual host, but a different hostname is passed with each application layer request. Services like FTP are not capable of operating in this way, and they demand network layer virtual hosting.

Q.23. How can we configure IP alias? What is *ifconfig*? **[Important]**

Ans. To configure an alias for an interface, you must first ensure that your kernel has been compiled with support for IP Alias (check that you have a */proc/net/ip_alias* file; if not, you will have to recompile your kernel). Configuration of an IP alias is virtually identical to

configuring a real network device; you use a special name to indicate it's an alias that you want. For example :

```
# ifconfig lo:0 172.16.1.1
```

This command would produce an alias for the loopback interface with the address 172.16.1.1. IP aliases are referred to by appending: n to the actual network device, in which "n" is an integer. In our example, the network device we are creating the alias on is lo, and we are creating an alias numbered zero for it. This way, a single physical device may support a number of aliases.

Each alias may be treated as though it is a separate device, and as far as the kernel IP software is concerned, it will be; however, it will be sharing its hardware with another interface.

All About ifconfig : There are many more parameters to ifconfig. Its normal invocation is this :

```
ifconfig interface [address [parameters]]
```

interface is the interface name, and *address* is the IP address to be assigned to the interface. This may be either an IP address in dotted quad notation or a name that ifconfig will look up in */etc/hosts*.

If ifconfig is invoked with only the interface name, it displays that interface's configuration. When invoked without any parameters, it displays all interfaces you have configured so far; a *-a* option forces it to show the inactive ones as well. A sample invocation for the Ethernet interface *eth0* may look like this :

```
# ifconfig eth0
eth0  Link encap 10Mbps Ethernet HWaddr 00:00:C0:90:B3:42
      inet addr 172.16.1.2 Bcast 172.16.1.255 Mask 255.255.255.0
      UP BROADCAST RUNNING MTU 1500 Metric 0
      RX packets 3136 errors 217 dropped 7 overrun 26
      TX packets 1752 errors 25 dropped 0 overrun 0
```

The MTU and Metric fields show the current MTU and metric value for that interface. The metric value is traditionally used by some operating systems to compute the cost of a route. Linux doesn't use this value yet, but defines it for compatibility, nevertheless.

The RX and TX lines show how many packets have been received or transmitted error free, how many errors occurred, how many packets were dropped (probably because of low memory), and how many were lost because of an overrun. Receiver overruns usually occur when packets come in faster than the kernel can service the last interrupt. The flag values printed by ifconfig roughly correspond to the names of its command-line options.

The following is a list of parameters recognized by ifconfig with the corresponding flag names. Options that simply turn on a feature also allow it to be turned off again by preceding the option name by a dash (-).

up : This option makes an interface accessible to the IP layer. This option is implied when an *address* is given on the command line. It may also be used to reenable an interface that has been taken down temporarily using the down option.

This option corresponds to the flags *UP* and *RUNNING*.

down : This option marks an interface inaccessible to the IP layer. This effectively disables any IP traffic through the interface. Note that this option will also automatically delete all routing entries that use this interface.

netmask *mask* : This option assigns a subnet mask to be used by the interface. It may be given as either a 32-bit hexadecimal number preceded by 0x, or as a dotted quad of decimal numbers. While the dotted quad format is more common, the hexadecimal representation is often easier to work with. Netmasks are essentially binary, and it is easier to do binary-to-hexadecimal than binary-to-decimal conversion.

pointopoint *address* : This option is used for point-to-point IP links that involve only two hosts. This option is needed to configure SLIP or PLIP interfaces, for example. If a point-to-point address has been set, ifconfig displays the *POINTOPOINT* flag.

broadcast *address* : The broadcast address is usually made up from the network number by setting all bits of the host part. Some IP implementations (systems derived from BSD 4.2, for instance) use a different scheme in which all host part bits are cleared instead. The *broadcast* option adapts to these strange environments. If a broadcast address has been set, ifconfig displays the *BROADCAST* flag.

***irq* :** This option allows you to set the IRQ line used by certain devices. This is especially useful for PLIP, but may also be useful for certain Ethernet cards.

metric *number* : This option may be used to assign a metric value to the routing table entry created for the interface. This metric is used by the Routing Information Protocol (RIP) to build routing tables for the network. The default metric used by ifconfig is zero. If you don't run a RIP daemon, you don't need this option at all; if you do, you will rarely need to change the metric value.

RIP chooses the optimal route to a given host based on the "length" of the path. It is computed by summing up the individual metric values of each host-to-host link. By default, a hop has length 1, but this may be any positive integer less than 16. (A route length of 16 is equal to infinity. Such routes are considered unusable.) The *metric* parameter sets this hop cost, which is then broadcast by the routing daemon.

mtu *bytes* : This sets the Maximum Transmission Unit, which is the maximum number of octets the interface is able to handle in one transaction. For Ethernets, the MTU defaults to 1,500 (the largest allowable size of an Ethernet packet); for SLIP interfaces, it is 296. (There is no constraint on the MTU of SLIP links; this value is a good compromise.)

***-arp* :** This option disables the use of ARP on this interface.

***-promisc* :** This option turns promiscuous mode off.

***-allmulti*:** This option turns multicast addresses off.

Q.24. Define the netstat Command in detail with examples. **[Important]**

Ans. Netstat is a useful tool for checking your network configuration and activity. It is in fact a collection of several tools lumped together.

Displaying the Routing Table : When you invoke netstat with the *-r* flag, it displays the kernel routing table in the way we've been doing with route. On *vstout*, it produces:

```
# netstat -nr
Kernel IP routing table
Destination  Gateway      Genmask          Flags  MSS Window  irtt  Iface
127.0.0.1    *            255.255.255.255  UH       0      0     0  lo
172.16.1.0   *            255.255.255.0    U        0      0     0  eth0
172.16.2.0   172.16.1.1   255.255.255.0    UG       0      0     0  eth0
```

The *-n* option makes netstat print addresses as dotted quad IP numbers rather than the symbolic host and network names. This option is especially useful when you want to avoid address lookups over the network (e.g., to a DNS or NIS server).

The second column of netstat's output shows the gateway to which the routing entry points. If no gateway is used, an asterisk is printed instead. The third column shows the "generality" of the route, i.e., the network mask for this route. When given an IP address to find a suitable route for, the kernel steps through each of the routing table entries, taking the bitwise AND of the address and the genmask before comparing it to the target of the route.

The fourth column displays the following flags that describe the route :

G : The route uses a gateway.

U : The interface to be used is up.

H : Only a single host can be reached through the route. For example, this is the case for the loopback entry *127.0.0.1*.

D : This route is dynamically created. It is set if the table entry has been generated by a routing daemon like gated or by an ICMP redirect message.

M : This route is set if the table entry was modified by an ICMP redirect message.

! : The route is a reject route and datagrams will be dropped.

Q.25. What are the commands for checking the ARP Tables in Unix?

Ans. On some occasions, it is useful to view or alter the contents of the kernel's ARP tables, for example when you suspect a duplicate Internet address is the cause for some intermittent network problem. The arp tool was made for situations like this. Its command-line options are:

```
arp [-v] [-t hwtype] -a [hostname]
arp [-v] [-t hwtype] -s hostname hwaddr
arp [-v] -d hostname [hostname...]
```

All *hostname* arguments may be either symbolic hostnames or IP addresses in dotted quad notation.

The first invocation displays the ARP entry for the IP address or host specified, or all hosts known if no *hostname* is given. For example, invoking arp on *vlager* may yield :

```
# arp -a
IP address   HW type          HW address
172.16.1.3   10Mbps Ethernet  00:00:C0:5A:42:C1
172.16.1.2   10Mbps Ethernet  00:00:C0:90:B3:42
172.16.2.4   10Mbps Ethernet  00:00:C0:04:69:AA
```

which shows the Ethernet addresses of *vlager*, *vstout* and *vale*.

You can limit the display to the hardware type specified using the *-t* option. This may be *ether*, *ax25*, or *pronet*, standing for 10 Mbps Ethernet; AMPR AX.25, and IEEE 802.5 token ring equipment, respectively.

The *-s* option is used to permanently add *hostname*'s Ethernet address to the ARP tables. The *hwaddr* argument specifies the hardware address, which is by default expected to be an Ethernet address specified as six hexadecimal bytes separated by colons. You may also set the hardware address for other types of hardware, using the *-t* option.

For some reason, ARP queries for the remote host sometimes fail, for instance when its ARP driver is buggy or there is another host in the network that erroneously identifies itself with that host's IP address; this problem requires you to manually add an IP address to the ARP table. A hard-wiring IP address in the ARP table is also a (very drastic) measure to protect yourself from hosts on your Ethernet that pose as someone else.

Invoking arp using the *-d* switch deletes all ARP entries relating to the given host. This switch may be used to force the interface to re-attempt obtaining the Ethernet address for the IP address in question. This is useful when a misconfigured system has broadcasted wrong ARP information (of course, you have to reconfigure the broken host first).

The -s option may also be used to implement *proxy* ARP. This is a special technique through which a host, say *gate*, acts as a gateway to another host named *fnord* by pretending that both addresses refer to the same host, namely *gate*. It does so by publishing an ARP entry for *fnord* that points to its own Ethernet interface. Now when a host sends out an ARP query for *fnord*, *gate* will return a reply containing its own Ethernet address. The querying host will then send all datagrams to *gate*, which dutifully forwards them to *fnord*.

These contortions may be necessary when you want to access *fnord* from a DOS machine with a broken TCP implementation that doesn't understand routing too well. When you use proxy ARP, it will appear to the DOS machine as if *fnord* was on the local subnet, so it doesn't have to know about how to route through a gateway.

Another useful application of proxy ARP is when one of your hosts acts as a gateway to some other host only temporarily, for instance, through a dial-up link. In a previous example, we encountered the laptop *vlite*, which was connected to *vlager* through a PLIP link from time to time. Of course, this application will work only if the address of the host you want to provide proxy ARP for is on the same IP subnet as your gateway. *vstout* could proxy ARP for any host on the Brewery subnet (*172.16.1.0*), but never for a host on the Winery subnet (*172.16.2.0*).

The proper invocation to provide proxy ARP for *fnord* is given below; of course, the given Ethernet address must be that of *gate* :

```
# arp -s fnord 00:00:c0:a1:42:e0 pub
```

The proxy ARP entry may be removed again by invoking :

```
# arp -d fnord
```

Q.26. What are the different issues in TCP/IP networking?

Ans. *Issues of TCP/IP Networking*, TCP/IP networking may rely on different schemes to convert names into addresses. The simplest way is a host table stored in */etc/hosts*. This is useful only for small LANs that are run by one single administrator and otherwise have no IP traffic with the outside world.

Alternatively, you can use the Berkeley Internet Name Domain service (BIND) for resolving hostnames to IP addresses. Configuring BIND can be a real chore, but once you've done it, you can easily make changes in the network topology. On Linux, as on many other Unixish systems, name service is provided through a program called named. At startup, it loads a set of master files into its internal cache and waits for queries from remote or local user processes. There are different ways to set up BIND, and not all require you to run a name server on every host. It should be sufficient if you have a small LAN and an Internet uplink.

Q.27. What do you mean by Resolver Library? Define its various files.

Ans. The term resolver refers not to a special application, but to the resolver library. This is a collection of functions that can be found in the standard C library. The central routines are *gethostbyname(2)* and *gethostbyaddr(2)*, which look up all IP addresses associated with a host name, and vice versa. They may be configured to simply look up the information in *hosts*, to query a number of DNS name servers, or to use the hosts database of Network Information Service (NIS).

The resolver functions read configuration files when they are invoked. From these configuration files, they determine what databases to query, in which order, and other details relevant to how you've configured your environment. The older Linux standard library, libc, used */etc/host.conf* as its master configuration file, but Version 2 of the GNU standard library, glibc, uses */etc/nsswitch.conf*.

1. The host.conf File : The */etc/host.conf* tells the older Linux standard library resolver functions which services to use, and in what order.

Options in *host.conf* must appear on separate lines. Fields may be separated by white space (spaces or tabs). A hash sign (#) introduces a comment that extends to the next newline. The following options are available :

order : This option determines the order in which the resolving services are tried. Valid options are *bind* for querying the name server, *hosts* for lookups in */etc/hosts*, and *nis* for NIS lookups. Any or all of them may be specified. The order in which they appear on the line determines the order in which the respective services are tried.

multi : multi takes *on* or *off* as options. This determines if a host in */etc/hosts* is allowed to have several IP addresses, which is usually referred to as being "multi-homed." The default is off. This flag has no effect on DNS or NIS queries.

nospoof : DNS allows you to find the hostname belonging to an IP address by using the *in-addr.arpa* domain. Attempts by name servers to supply a false hostname are called spoofing. To guard against this, the resolver can be configured to check whether the original IP address is in fact associated with the obtained hostname. If not, the name is rejected and an error is returned. This behaviour is turned on by setting *nospoof* on.

alert : This option takes *on* or *off* as arguments. If it is turned on, any spoof attempts will cause the resolver to log a message to the syslog facility.

trim : This option takes an argument specifying a domain name that will be removed from hostnames before lookup. This is useful for *hosts* entries, for which you might only want to specify hostnames without a local domain. If you specify your local domain name here, it will be removed from a lookup of a host with the local domain name appended, thus allowing the lookup in */etc/hosts* to succeed. The domain name you add must end with the (.) character (e.g., :linux.org.au.) if trim is to work correctly.

trim options accumulate; you can consider your host as being local to several domains.

A sample file for *vlager* is shown in Example.

Example : Sample host.conf File.

```
# /etc/host.conf
# We have named running, but no NIS (yet)
order bind,hosts
# Allow multiple addrs
multi on
# Guard against spoof attempts
nospoof on
# Trim local domain (not really necessary).
trim vbrew.com.
```

2. The nsswitch.conf File : Version 2 of the GNU standard library includes a more powerful and flexible replacement for the older *host.conf* mechanism. The concept of the name service has been extended to include a variety of different types of information. Configuration

options for all of the different functions that query these databases have been brought back into a single configuration file; the *nsswitch.conf* file.

The *nsswitch.conf* file allows the system administrator to configure a wide variety of different databases. We'll limit our discussion to options that relate to host and network address resolution.

Options in *nsswitch.conf* must appear on separate lines. Fields may be separated by whitespace (spaces or tabs). A hash sign (#) introduces a comment that extends to the next newline. Each line describes a particular service; hostname resolution is one of these. The first field in each line is the name of the database, ending with a colon. The database name associated with host address resolution is *hosts*. A related database is *networks*, which is used for resolution of network names into network addresses. The remainder of each line stores options that determine the way lookups for that database are performed.

The following options are available :

`dns`

Use the Domain Name System (DNS) service to resolve the address. This makes sense only for host address resolution, not network address resolution. This mechanism uses the */etc/resolv.conf* file.

`files`

Search a local file for the host or network name and its corresponding address. This option uses the traditional */etc/hosts* and */etc/network* files.

`nis` or `nisplus`

The order in which the services to be queried are listed determines the order in which they are queried when attempting to resolve a name. The query-order list is in the service description in the */etc/nsswitch.conf* file. The services are queried from left to right and by default searching stops when a resolution is successful.

3. Configuring Name Server Lookups Using resolv.conf : When configuring the resolver library to use the BIND name service for host lookups, you also have to tell it which name servers to use. There is a separate file for this called *resolv.conf*. If this file does not exist or is empty, the resolver assumes the name server is on your local host.

To run a name server on your local host, you have to set it up separately. If you are on a local network and have the opportunity to use an existing name server, this should always be preferred. If you use a dialup IP connection to the Internet, you would normally specify the name server of your service provider in the *resolv.conf* file.

Q.28. How DNS Works? Explain its working in detail. **[Important]**

Ans. DNS organizes hostnames in a domain hierarchy. A domain is a collection of sites that are related in some sense — because they form a proper network (e.g., all machines on campus, or all hosts on BITNET), because they all belong to a certain organization (e.g., the U.S. government), or because they're simply geographically close. For instance, universities are commonly grouped in the *edu* domain, with each university or college using a separate subdomain, below which their hosts are subsumed. Groucho Marx University have the *groucho.edu* domain, while the LAN of the Mathematics department is assigned *maths.groucho.edu*. Hosts on the departmental network would have this domain name tacked onto their hostname, so *erdos* would be known as *erdos.maths.groucho.edu*. This is called the fully qualified domain name (FQDN), which uniquely identifies this host worldwide.

Figure 1 shows a section of the namespace. The entry at the root of this tree, which is denoted by a single dot, is quite appropriately called the root domain and encompasses all other domains. To indicate that a hostname is a fully qualified domain name, rather than a name relative to some (implicit) local domain, it is sometimes written with a trailing dot. This dot signifies that the name's last component is the root domain.

Depending on its location in the name hierarchy, a domain may be called top-level, second-level, or third-level. More levels of subdivision occur, but they are rare. This list details several top-level domains you may see frequently :

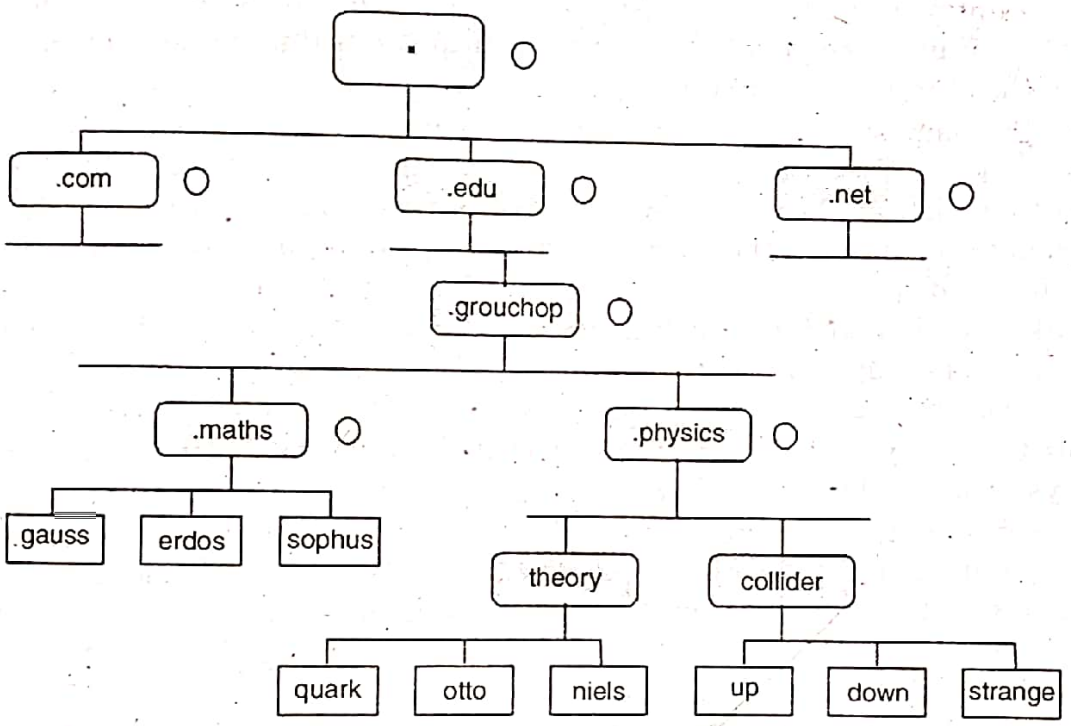


Fig. 1. A part of the domain namespace

Domain	Description
edu	(Mostly U.S.) educational institutions like universities.
com	Commercial organizations and companies.
org	Non-commercial organizations. Private UUCP networks are often in this domain.
net	Gateways and other administrative hosts on a network.
mil	U.S. military institutions.
gov	U.S. government institutions.
uucp	Officially, all site names formerly used as UUCP names without domains have been moved to this domain.

Historically, the first four of these were assigned to the U.S., but recent changes in policy have meant that these domains, named global Top Level Domains (gTLD), are now considered global in nature. Negotiations are currently underway to broaden the range of gTLDs, which may result in increased choice in the future.

Outside the U.S., each country generally uses a top-level domain of its own named after the two-letter country code defined in ISO-3166. Finland, for instance, uses the *fi* domain; *fr* is

used by France, *de* by Germany, and *au* by Australia and 'in' is used for India. Below this t
level domain, each country's NIC is free to organize hostnames in whatever way they wa
Australia has second-level domains similar to the international top-level domains, nam
com.au and *edu.au*. Other countries, like Germany, don't use this extra level, but have sligh
long names that refer directly to the organizations running a particular domain. It's
uncommon to see hostnames like *ftp.informatik.uni-erlangen.de*. Chalk that up to Germ
efficiency.

Of course, these national domains do not imply that a host below that domain is actu
located in that country; it means only that the host has been registered with that count
NIC. A Swedish manufacturer might have a branch in Australia and still have all its h
registered with the *se* top-level domain.

Organizing the namespace in a hierarchy of domain names nicely solves the proble
name uniqueness; with DNS, a hostname has to be unique only within its domain to give
name different from all other hosts worldwide. Furthermore, fully qualified names are eas
remember. Taken by themselves, these are already very good reasons to split up a large dom
into several subdomains.

Name Lookups with DNS : At first glance, all this domain and zone fuss seems to m
name resolution an awfully complicated business. After all, if no central authority con
what names are assigned to which hosts, how is a humble application supposed to know

Now comes the really ingenious part about DNS. If you want to find the IP addre
erdos, DNS says, "Go ask the people who manage it, and they will tell you."

In fact, DNS is a giant distributed database. It is implemented by so-called name se
that supply information on a given domain or set of domains. For each zone there are at
two, or at most a few, name servers that hold all authoritative information on hosts in
zone. To obtain the IP address of *erdos*, all you have to do is contact the name server fo
groucho.edu zone, which will then return the desired data.

Easier said than done, you might think. So how do I know how to reach the name serv
Groucho Marx University? In case your computer isn't equipped with an address-reso
oracle, DNS provides for this, too. When your application wants to look up informatio
erdos, it contacts a local name server, which conducts a so-called iterative query for it. It s
off by sending a query to a name server for the root domain, asking for the addre
erdos.maths.groucho.edu. The root name server recognizes that this name does not belo
its zone of authority, but rather to one below the *edu* domain. Thus, it tells you to conta
edu zone name server for more information and encloses a list of all *edu* name servers
with their addresses. Your local name server will then go on and query one of those, for ins
a.isi.edu. In a manner similar to the root name server, *a.isi.edu* knows that the *grouch*
people run a zone of their own, and points you to their servers. The local name server wil
present its query for *erdos* to one of these, which will finally recognize the name as belo
to its zone, and return the corresponding IP address.

This looks like a lot of traffic being generated for looking up a measly IP address, b
really only miniscule compared to the amount of data that would have to be transferre
were still stuck with *HOSTS.TXT*. There's still room for improvement with this sc
however.

To improve response time during future queries, the name server stores the inform
obtained in its local cache. So the next time anyone on your local network wants to look
address of a host in the *groucho.edu* domain, your name server will go directly to the *grouc*
name server.

Of course, the name server will not keep this information forever; it will discard it after some time. The expiration interval is called the time-to-live, or TTL. Each datum in the DNS database is assigned such a TTL by administrators of the responsible zone.

Types of Name Servers : Name servers that hold all information on hosts within a zone are called authoritative for this zone, and sometimes are referred to as master name servers. Any query for a host within this zone will end up at one of these master name servers.

Master servers must be fairly well synchronized. Thus, the zone's network administrator must make one the primary server, which loads its zone information from data files, and makes the others secondary servers, which transfer the zone data from the primary server at regular intervals.

Having several name servers distributes workload; it also provides backup. When one name server machine fails in a benign way, like crashing or losing its network connection, all queries will fall back to the other servers. Of course, this scheme doesn't protect you from server malfunctions that produce wrong replies to all DNS requests, such as from software bugs in the server program itself.

You can also run a name server that is not authoritative for any domain. This is useful, as the name server will still be able to conduct DNS queries for the applications running on the local network and cache the information. Hence it is called a caching-only server.

The DNS Database : We have seen that DNS not only deals with IP addresses of hosts, but also exchanges information on name servers. DNS databases may have, in fact, many different types of entries.

A single piece of information from the DNS database is called a resource record (RR). Each record has a type associated with it describing the sort of data it represents, and a class specifying the type of network it applies to. The latter accommodates the needs of different addressing schemes, like IP addresses (the IN class), Hesiod addresses (used by MIT's Kerberos system), and a few more. The prototypical resource record type is the A record, which associates a fully qualified domain name with an IP address.

A host may be known by more than one name. For example you might have a server that provides both FTP and World Wide Web servers, which you give two names: *ftp.machine.org* and *www.machine.org*. However, one of these names must be identified as the official or canonical hostname, while the others are simply aliases referring to the official hostname. The difference is that the canonical hostname is the one with an associated A record, while the others only have a record of type CNAME that points to the canonical hostname.

We will not go through all record types here, but we will give you a brief example. Example shows a part of the domain database that is loaded into the name servers for the *physics.groucho.edu* zone.

2.29. What is the structure of DNS Database Files? Also give all the related commands in unix.

ns. Master files included with named, like named.hosts, always have a domain associated ith them, which is called the origin. This is the domain name specified with the *cache* and *rimary* options. Within a master file, you are allowed to specify domain and host names elative to this domain. A name given in a configuration file is considered absolute if it ends in single dot, otherwise it is considered relative to the origin. The origin by itself may be referred using (@).

The data contained in a master file is split up in resource records(RRs). RRs are the smallest nits of information available through DNS. Each resource record has a type. A records, for

instance, map a hostname to an IP address, and a CNAME record associates an alias for a h with its official hostname. To see an example, look at Example, which shows the named.ho master file for the Virtual Brewery.

Resource record representations in master files share a common format.:

[domain] [ttl] [class] type rdata

Fields are separated by spaces or tabs. An entry may be continued across several lines an opening brace occurs before the first newline and the last field is followed by a closi brace. Anything between a semicolon and a newline is ignored. A description of the form terms follows:

domain

This term is the domain name to which the entry applies. If no domain name is given, t RR is assumed to apply to the domain of the previous RR.

ttl

In order to force resolvers to discard information after a certain time, each RR is associat a time to live (ttl). The ttl field specifies the time in seconds that the information is valid af it has been retrieved from the server. It is a decimal number with at most eight digits.

If no ttl value is given, the field value defaults to that of the *minimum* field of the preced SOA record.

class

This is an address class, like IN for IP addresses or HS for objects in the Hesiod class. I TCP/IP networking, you have to specify IN.

If no class field is given, the class of the preceding RR is assumed.

type

This describes the type of the RR. The most common types are A, SOA, PTR, and NS.

rdata

This holds the data associated with the RR. The format of this field depends on the type RR.

A

This record associates an IP address with a hostname. The resource data field contai the address in dotted quad notation.

For each hostname, there must be only one A record. The hostname used in this A recor is considered the official or canonical hostname. All other hostnames are aliases and must mapped onto the canonical hostname using a CNAME record. If the canonical name of a host were vlager, we'd have an A record that associated that hostname with its IP addres Since we may also want another name associated with that address, say news, we'd create CNAME record that associates this alternate name with the canonical name.

NS

NS records are used to specify a zone's primary server and all its secondary servers. A NS record points to a master name server of the given zone, with the resource data fie containing the hostname of the name server.

You will meet NS records in two situations: The first situation is when you delega authority to a subordinate zone; the second is within the master zone database of the subordinat zone itself. The sets of servers specified in both the parent and delegated zones should matc

The NS record specifies the name of the primary and secondary name servers for a zon These names must be resolved to an address so they can be used. Sometimes the serve

belong to the domain they are serving, which causes a "chicken and egg" problem; we can't resolve the address untii the name server is reachable, but we can't reach the name server until we resolve its address. To solve this dilemma, we can configure special A records directly into the name server of the parent zone. The A records allow the name servers of the parent domain to resolve the IP address of the delegated zone name servers. These records are commonly called glue records because they provide the "glue" that binds a delegated zone to its parent.

CNAME

This record associates an alias with a host's canonical hostname. It provides an alternate name by which users can refer to the host whose canonical name is supplied as a parameter. The canonical hostname is the one the master file provides an A record for; aliases are simply linked to that name by a CNAME record, but don't have any other records of their own.

PTR

This type of record is used to associate names in the *in-addr.arpa* domain with hostnames. It is used for reverse mapping of IP addresses to hostnames. The hostname given must be the canonical hostname.

MX

This RR announces a mail exchanger for a domain. The syntax of an MX record is :

`[domain] [ttl] [class] MX preference host`

host names the mail exchanger for *domain*. Every mail exchanger has an integer *preference* associated with it. A mail transport agent that wants to deliver mail to *domain* tries all hosts who have an MX record for this domain until it succeeds. The one with the lowest preference value is tried first, then the others, in order of increasing preference value.

HINFO

This record provides information on the system's hardware and software. Its syntax is :

`[domain] [ttl] [class] HINFO hardware software`

The *hardware* field identifies the hardware used by this host. Special conventions are used to specify this. A list of valid "machine names" is given in the Assigned Numbers RFC (RFC-1700). If the field contains any blanks, it must be enclosed in double quotes. The *software* field names the operating system software used by the system. Again, a valid name from the Assigned Numbers RFC should be chosen.

❐❐❐

UNIT—3

TCP/IP FIREWALL AND NETWORK ADDRESS TRANSLATION

Q.1. Why security is important for individuals?

Ans. Security is increasingly important for companies and individuals alike. The Internet h provided them with a powerful tool to distribute information about them and obtain informati from others, but it has also exposed them to dangers that they have previously been exem from. Computer crime, information theft, and malicious damage are all potential dangers.

An unauthorized and unscrupulous person who gains access to a computer system m guess system passwords or exploit the bugs and idiosyncratic behaviour of certain progra to obtain a working account on that machine. Once they are able to log in to the machine, th may have access to information that may be damaging, such as commercially sensiti information like marketing plans, new project details, or customer information databas Damaging or modifying this type of data can cause severe setbacks to the company.

The safest way to avoid such widespread damage is to prevent unauthorized people fr gaining network access to the machine. This is where firewalls come in.

WARNING : Constructing secure firewalls is an art. It involves a good understanding technology, but equally important, it requires an understanding of the philosophy behi firewall designs.

Q.2. Discuss various methods of attack and methods to prevent them. [Importa

Ans. Methods of Attack : As a network administrator, it is important that you understa the nature of potential attacks on computer security. Here are some of the more import methods of attack and ways of protecting yourself against them :

1. **Unauthorized access :** This simply means that people who shouldn't use yo computer services are able to connect and use them. For example, people outside yo company might try to connect to your company accounting machine or to your N server.

   Prevention : There are various ways to avoid this attack by carefully specifying w can gain access through these services. You can prevent network access to all exc the intended users.

2. **Exploitation of known weaknesses in programs :** Some programs and netw services were not originally designed with strong security in mind and are inheren vulnerable to attack. The BSD remote services (rlogin, rexec, etc.) are an example

   Prevention : The best way to protect yourself against this type of attack is to disa any vulnerable services or find alternatives. With Open Source, it is sometimes possi to repair the weaknesses in the software.

3. **Denial of service:** Denial of service attacks cause the service or program to ce functioning or prevent others from making use of the service or program. These

be performed at the network layer by sending carefully crafted and malicious datagrams that cause network connections to fail. They may also be performed at the application layer, where carefully crafted application commands are given to a program that cause it to become extremely busy or stop functioning.

Prevention : Preventing suspicious network traffic from reaching your hosts and preventing suspicious program commands and requests are the best ways of minimizing the risk of a denial of service attack. It's useful to know the details of the attack method, so you should educate yourself about each new attack as it gets publicized.

4. **Spoofing :** This type of attack causes a host or application to mimic the actions of another. Typically the attacker pretends to be an innocent host by following IP addresses in network packets. For example, a well-documented exploit of the BSD rlogin service can use this method to mimic a TCP connection from another host by guessing TCP sequence numbers.

   Prevention : To protect against this type of attack, verify the authenticity of datagrams and commands. Prevent datagram routing with invalid source addresses. Introduce unpredictablility into connection control mechanisms, such as TCP sequence numbers and the allocation of dynamic port addresses.

5. **Eavesdropping :** This is the simplest type of attack. A host is configured to "listen" to and capture data not belonging to it. Carefully written eavesdropping programs can take usernames and passwords from user login network connections. Broadcast networks like Ethernet are especially vulnerable to this type of attack.

   Prevention : To protect against this type of threat, avoid use of broadcast network technologies and enforce the use of data encryption.

IP firewalling is very useful in preventing or reducing unauthorized access, network layer denial of service, and IP spoofing attacks. It is not very useful in avoiding exploitation of weaknesses in network services or programs and eavesdropping.

Q.3. What is a Firewall? Discuss the importance of firewall in networks. **[Important]**

Ans. A firewall is a secure and trusted machine that sits between a private network and a public network. The firewall machine is configured with a set of rules that determines which network traffic will be allowed to pass and which will be blocked or refused. In some large organizations, you may even find a firewall located inside their corporate network to segregate sensitive areas of the organization from other employees. Many cases of computer crime occur from within an organization, not just from outside.

The term *firewall* comes from a device used to protect people from fire. The firewall is a shield of material resistant to fire that is placed between a potential fire and the people it is protecting.

Firewalls can be constructed in quite a variety of ways. The most sophisticated arrangement involves a number of separate machines and is known as a *perimeter network*. Two machines act as "filters" called chokes to allow only certain types of network traffic to pass, and between these chokes reside network servers such as a mail gateway or a World Wide Web proxy server. This configuration can be very safe and easily allows quite a great range of control over who can connect both from the inside to the outside, and from the outside to the inside. This sort of configuration might be used by large organizations.

Typically though, firewalls are single machines that serve all of these functions. These are a little less secure, because if there is some weakness in the firewall machine itself that allows

people to gain access to it, the whole network security has been breached. Nevertheless, these types of firewalls are cheaper and easier to manage than the more sophisticated arrangement just described. Figure 1 illustrates the two most common firewall configurations.

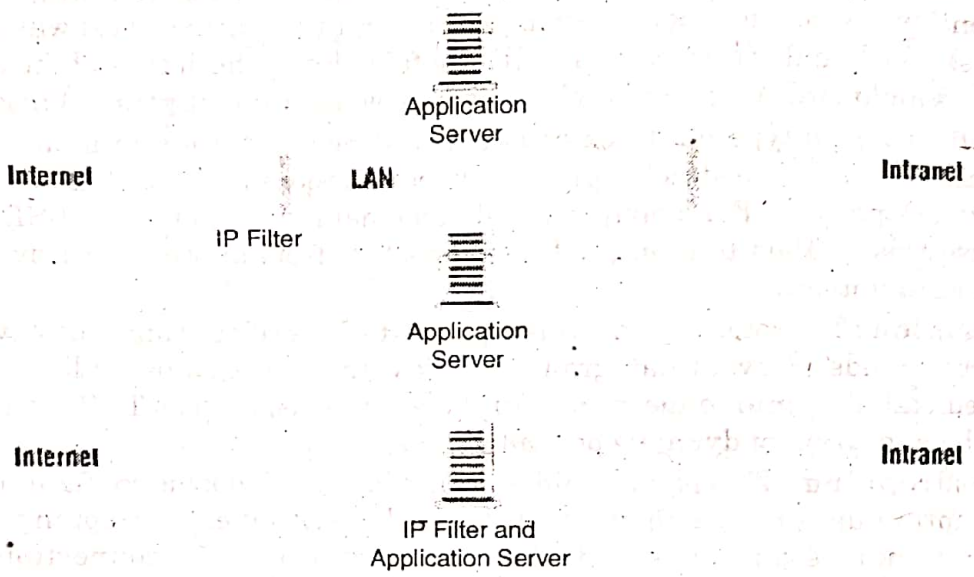


Fig. 1. The two major classes of firewall design

The Linux kernel provides a range of built-in features that allow it to function quite nicely as an IP firewall. The network implementation includes code to do IP filtering in a number of different ways, and provides a mechanism to quite accurately configure what sort of rules you'd like to put in place. The Linux firewall is flexible enough to make it very useful in either of the configurations illustrated in Fig. 1.

Q.4. What Is IP Filtering? **[Important]**

Ans. IP filtering is simply a mechanism that decides which types of IP datagrams will be processed normally and which will be discarded. By *discarded* we mean that the datagram is deleted and completely ignored, as if it had never been received. You can apply many different sorts of criteria to determine which datagrams you wish to filter; some examples of these are :

- **Protocol type :** TCP, UDP, ICMP, etc.
- **Socket number (for TCP/UPD)**
- **Datagram type :** SYN/ACK, data, ICMP Echo Request, etc.
- **Datagram source address :** where it came from
- **Datagram destination address :** where it is going to

It is important to understand at this point that IP filtering is a network layer facility. This means it doesn't understand anything about the application using the network connections, only about the connections themselves. For example, you may deny users access to your internal network on the default telnet port, but if you rely on IP filtering alone, you can't stop them from using the telnet program with a port that you do allow to pass through your firewall. You can prevent this sort of problem by using proxy servers for each service that you allow across your firewall. The proxy servers understand the application they were designed to proxy and can therefore prevent abuses, such as using the telnet program to get past a firewall by using the World Wide Web port. If your firewall supports a World Wide Web proxy, their telnet

connection will always be answered by the proxy and will allow only HTTP requests to pass. A large number of proxy-server programs exist. Some are free software and many others are commercial products.

The IP filtering rule set is made up of many combinations of the criteria listed previously For example, let's imagine that you wanted to allow World Wide Web users within the Virtual Brewery network to have no access to the Internet except to use other sites' web servers. You would configure your firewall to allow forwarding of :

- datagrams with a source address on Virtual Brewery network, a destination address of anywhere, and with a destination port of 80 (WWW)
- datagrams with a destination address of Virtual Brewery network and a source port of 80 (WWW) from a source address of anywhere

Note that we've used two rules here. We have to allow our data to go out, but also the corresponding reply data to come back in. Linux simplifies this and allows us to specify this in one command.

Q.5. How we can build Linux IP Firewalling?

Ans. To build a Linux IP firewall, it is necessary to have a kernel built with IP firewall support and the appropriate configuration utility. In all production kernels prior to the 2.2 series, you would use the ipfwadm utility. The 2.2.x kernels marked the release of the third generation of IP firewall for Linux called *IP Chains*. IP chains use a program similar to ipfwadm called ipchains. Linux kernels 2.3.15 and later support the fourth generation of Linux IP firewalls called *netfilter*. The *netfilter* code is the result of a large redesign of the packet handling flow in Linux. The *netfilter* is a multifaceted creature, providing direct backward-compatible support for both ipfwadm and ipchains as well as a new alternative command called iptables.

Q.6. How does kernel configured with IP Firewall?

Ans. Kernel Configured with IP Firewall : The Linux kernel must be configured to support IP firewalling. There isn't much more to it than selecting the appropriate options when performing a make menuconfig of your kernel. In 2.2 kernels you should select the following options :

Networking options :

- Network firewalls
- TCP/IP networking
- IP : firewalling
- IP : firewall packet logging

Firewall packet logging is a special feature that writes a line of information about each datagram that matches a particular firewall rule out to a special device so you can see them.

In kernels 2.4.0, you should select this option instead :

Networking options :

- Network packet filtering (replaces ipchains)

IP : Netfilter Configuration —>

- Userspace queueing via NETLINK (EXPERIMENTAL)
- IP tables support (required for filtering/masq/NAT)
- limit match support
- MAC address match support

- netfilter MARK match support
- Multiple port match support
- TOS match support
- Connection state match support
- Unclean match support (EXPERIMENTAL)
- Owner match support (EXPERIMENTAL)
- Packet filtering
- REJECT target support
- MIRROR target support (EXPERIMENTAL)
- Packet mangling
- TOS target support
- MARK target support
- LOG target support
- ipchains (2.2-style) support
- ipfwadm (2.0-style) support

1. **The ipfwadm Utility :** The ipfwadm (IP Firewall Administration) utility is the tool used to build the firewall rules for all kernels prior to 2.2.0. Its command syntax can be very confusing because it can do such a complicated range of things, but we'll provide some common examples that will illustrate the most important variations of these:
   The ipfwadm utility is included in most modern Linux distributions, but perhaps not by default. There may be a specific software package for it that you have to install. If your distribution does not include it, you can obtain the source package from *ftp.xos.nl* in the */pub/linux/ipfwadm/* directory, and compile it yourself.
2. **The ipchains Utility :** Just as for the ipfwadm utility, the ipchains utility can be somewhat baffling to use at first. It provides all of the flexibility of ipfwadm with a simplified command syntax, and additionally provides a "chaining" mechanism that allows you to manage multiple rulesets and link them together.
   The ipchains command appears in most Linux distributions based on the 2.2 kernels. Included in the source package is a wrapper script called ipfwadm-wrapper that mimics the ipfwadm command, but actually invokes the ipchains command.
3. **The iptables Utility :** The syntax of the iptables utility is quite similar to that of the ipchains syntax. The changes are improvements and a result of the tool being redesigned to be extensible through shared libraries. Just as foripchains, we'll present iptables equivalents of the examples so you can compare and contrast its syntax with the others.

Q.7. What are the three ways for doing Filtering? **[Important]**

Ans. Consider how a UNIX machine, or in fact any machine capable of IP routing, processes IP datagrams. The basic steps, shown in Fig. 2 are :

- The IP datagram is received. (1)
- The incoming IP datagram is examined to determine if it is destined for a process on this machine.
- If the datagram is for this machine, it is processed locally. (2)
- If it is not destined for this machine, a search is made of the routing table for an appropriate route and the datagram is forwarded to the appropriate interface or dropped if no route can be found. (3)

- Datagrams from local processes are sent to the routing software for forwarding to the appropriate interface. (4)
- The outgoing IP datagram is examined to determine if there is a valid route for it to take, if not, it is dropped.
- The IP datagram is transmitted. (5)

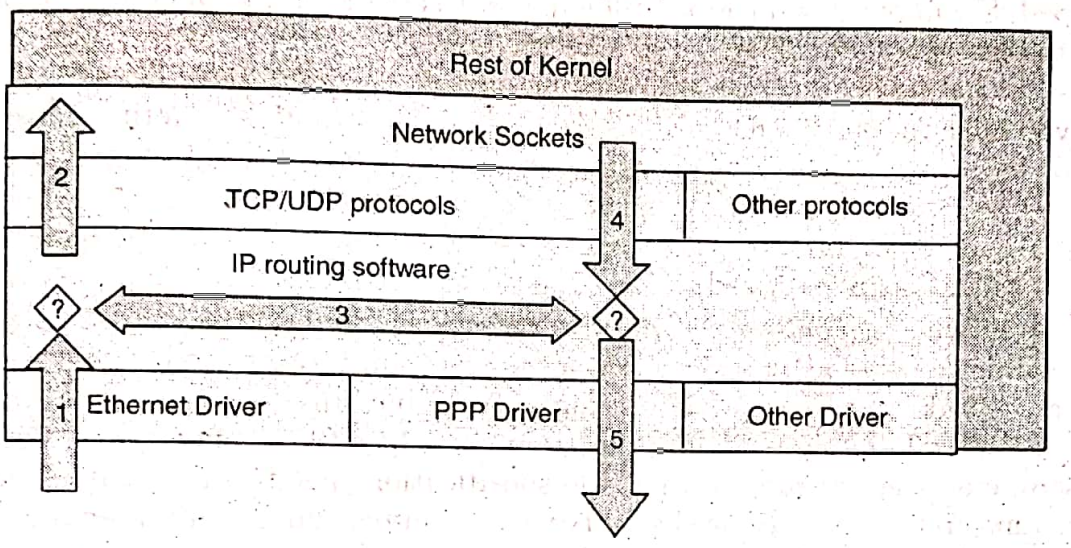


Fig. 2. The stages of IP datagram processing

In our diagram, the flow 1 → 3 → 5 represents our machine routing data between a host on our Ethernet network to a host reachable via our PPP link. The flows 1 → 2 and 4 → 5 represent the data input and output flows of a network program running on our local host. The flow 4 → 3 → 2 would represent data flow via a loopback connection. Naturally data flows both into and out of network devices. The question marks on the diagram represent the points where the IP layer makes routing decisions.

The Linux kernel IP firewall is capable of applying filtering at various stages in this process. That is, you can filter the IP datagrams that come in to your machine, filter those datagrams being forwarded across your machine, and filter those datagrams that are ready to be transmitted.

In ipfwadm and ipchains, an Input rule applies to flow 1 on the diagram, a Forwarding rule to flow 3, and an Output rule to flow 5. We'll see when we discuss *netfilter* later that the points of interception have changed so that an Input rule is applied at flow 2, and an Output rule is applied at flow 4. This has important implications for how you structure your rulesets, but the general principle holds true for all versions of Linux firewalling.

This may seem unnecessarily complicated at first, but it provides flexibility that allows some very sophisticated and powerful configurations to be built.

Q.8. How do we test a Firewall Configuration?

Ans. It's important to validate that it does do what you want it to do. One way to do this is to use a test host outside your network to attempt to pierce your firewall: this can be quite clumsy and slow, though, and is limited to testing only those addresses that you can actually use.

A faster and easier method is available with the Linux firewall implementation. It allows you to manually generate tests and run them through the firewall configuration just as if you were testing with actual datagrams. All varieties of the Linux kernel firewall software, ipfwadm, ipchains, and iptables, provide support for this style of testing. The implementation involves use of the relevant *check* command.

The general test procedure is as follows:

1. Design and configure your firewall using ipfwadm, ipchains, or iptables.
2. Design a series of tests that will determine whether your firewall is actually working as you intend. For these tests you may use any source or destination address, so choose some address combinations that should be accepted and some others that should be dropped. If you're allowing or disallowing only certain ranges of addresses, it is a good idea to test addresses on either side of the boundary of the range — one address just inside the boundary and one address just outside the boundary. This will help ensure that you have the correct boundaries configured, because it is sometimes easy to specify netmasks incorrectly in your configuration. If you're filtering by protocol and port number, your tests should also check all important combinations of these parameters. For example, if you intend to accept only TCP under certain circumstances, check that UDP datagrams are dropped.
3. Develop ipfwadm, ipchains, or iptables rules to implement each test. It is probably worthwhile to write all the rules into a script so you can test and re-test easily as you correct mistakes or change your design. Tests use almost the same syntax as rule specifications, but the arguments take on slightly differing meanings. For example, the source address argument in a rule specification specifies the source address that datagrams matching this rule should have. The source address argument in test syntax, in contrast, specifies the source address of the test datagram that will be generated. For ipfwadm, you must use the –c option to specify that this command is a test, while for ipchains and iptables, you must use the –C option. In all cases you must *always* specify the source address, destination address, protocol, and interface to be used for the test. Other arguments, such as port numbers or TOS bit settings, are optional.
4. Execute each test command and note the output. The output of each test will be a single word indicating the final target of the datagram after running it through the firewall configuration — that is, where the processing ended. For ipchains and iptables, user-specified chains will be tested in addition to the built-in ones.
5. Compare the output of each test against the desired result. If there are any discrepancies, you will need to analyze your rule set to determine where you've made the error. If you've written your test commands into a script file, you can easily rerun the test after correcting any errors in your firewall configuration. It's a good practice to flush your rule sets completely and rebuild them from scratch, rather than to make changes dynamically. This helps ensure that the active configuration you are testing actually reflects the set of commands in your configuration script.

Let's take a quick look at what a manual test transcript would look like for our naïve example with ipchains. You will remember that our local network in the example was 172.16.1.0 with a netmask of 255.255.255.0, and we were to allow TCP connections out to web servers on the net. Nothing else was to pass our forward chain. Start with a transmission that we know should work, a connection from a local host to a web server outside :

```
# ipchains -C forward -p tcp -s 172.16.1.0 1025 -d 44.136.8.2 80 -i eth0
accepted
```

Note the arguments had to be supplied and the way they've been used to describe a datagram. The output of the command indicates that the datagram was accepted for forwarding, which is what we hoped for.

Now try another test, this time with a source address that doesn't belong to our network. This one should be denied:

```
# ipchains -C forward -p tcp -s 172.16.2.0 1025 -d 44.136.8.2 80 -i eth0
denied
```

You'll go a long way toward achieving peace of mind if you design a series of exhaustive tests. While this can sometimes be as difficult as designing the firewall configuration, it's also the best way of knowing that your design is providing the security you expect of it.

Q.9. Illustrate a Sample Firewall Configuration.

Ans. The configuration in this example has been designed to be easily extended and customized. We've provided three versions. The first version is implemented using the ipfwadm command (or the ipfwadm-wrapper script), the second uses ipchains, and the third uses iptables. The example doesn't attempt to exploit user-defined chains, but it will show you the similarities and differences between the old and new firewall configuration tool syntaxes:

IPFWADM VERSION

```
#!/bin/bash
##############################################################################
# IPFWADM VERSION
# This sample configuration is for a single host firewall configuration
# with no services supported by the firewall machine itself.
##############################################################################

# USER CONFIGURABLE SECTION

# The name and location of the ipfwadm utility. Use ipfwadm-wrapper for
# 2.2.* kernels.
IPFWADM=ipfwadm

# The path to the ipfwadm executable.
PATH="/sbin"

# Our internal network address space and its supporting network device.
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"

# The outside address and the network device that supports it.
ANYADDR="0/0"
ANYDEV="eth1"

# The TCP services we wish to allow to pass - "" empty means all ports
# note: space separated
TCPIN="smtp www"
TCPOUT="smtp www ftp ftp-data irc"
```

```
# The UDP services we wish to allow to pass - "" empty means all ports
# note: space separated
UDPIN="domain"
UDPOUT="domain"

# The ICMP services we wish to allow to pass - "" empty means all types
# ref: /usr/include/netinet/ip_icmp.h for type numbers
# note: space separated
ICMPIN="0 3 11"
ICMPOUT="8 3 11"

# Logging; uncomment the following line to enable logging of datagrams
# that are blocked by the firewall.
# LOGGING=1

# END USER CONFIGURABLE SECTION
##############################################################################
# Flush the Incoming table rules
$IPFWADM -I -f

# We want to deny incoming access by default.
$IPFWADM -I -p deny

# SPOOFING
# We should not accept any datagrams with a source address matching ours
# from the outside, so we deny them.
$IPFWADM -I -a deny -S $OURNET -W $ANYDEV

# SMURF
# Disallow ICMP to our broadcast address to prevent "Smurf" style attack.
$IPFWADM -I -a deny -P icmp -W $ANYDEV -D $OURBCAST

# TCP
# We will accept all TCP datagrams belonging to an existing connection
# (i.e. having the ACK bit set) for the TCP ports we're allowing through.
# This should catch more than 95 % of all valid TCP packets.
$IPFWADM -I -a accept -P tcp -D $OURNET $TCPIN -k -b

# TCP - INCOMING CONNECTIONS
# We will accept connection requests from the outside only on the
# allowed TCP ports.
$IPFWADM -I -a accept -P tcp -W $ANYDEV -D $OURNET $TCPIN -y
```

```
# TCP - OUTGOING CONNECTIONS
# We accept all outgoing tcp connection requests on allowed TCP ports.
$IPFWADM -I -a accept -P tcp -W $OURDEV -D $ANYADDR $TCPOUT -y

# UDP - INCOMING
# We will allow UDP datagrams in on the allowed ports.
$IPFWADM -I -a accept -P udp -W $ANYDEV -D $OURNET $UDPIN

# UDP - OUTGOING
# We will allow UDP datagrams out on the allowed ports.
$IPFWADM -I -a accept -P udp -W $OURDEV -D $ANYADDR $UDPOUT

# ICMP - INCOMING
# We will allow ICMP datagrams in of the allowed types.
$IPFWADM -I -a accept -P icmp -W $ANYDEV -D $OURNET $UDPIN

# ICMP - OUTGOING
# We will allow ICMP datagrams out of the allowed types.
$IPFWADM -I -a accept -P icmp -W $OURDEV -D $ANYADDR $UDPOUT

# DEFAULT and LOGGING
# All remaining datagrams fall through to the default
# rule and are dropped. They will be logged if you've
# configured the LOGGING variable above.
#
if [ "$LOGGING" ]
then
    # Log barred TCP
    $IPFWADM -I -a reject -P tcp -o

    # Log barred UDP
    $IPFWADM -I -a reject -P udp -o

    # Log barred ICMP
    $IPFWADM -I -a reject -P icmp -o
fi
#
# end.
```

Q.10. What do you mean by IP Accounting? Why it is so important? **[Important]**

Ans. In today's world of commercial Internet service, it is becoming increasingly important to know how much data you are transmitting and receiving on your network connections. If you are an Internet Service Provider and you charge your customers by volume, this will be essential to your business. If you are a customer of an Internet Service Provider that charges by data

volume, you will find it useful to collect your own data to ensure the accuracy of your Inte charges.

There are other uses for network accounting that have nothing to do with dollars and If you manage a server that offers a number of different types of network services, it mig useful to you to know exactly how much data is being generated by each one. This in information could assist you in making decisions, such as what hardware to buy or how servers to run.

The Linux kernel provides a facility that allows you to collect all sorts of useful inform about the network traffic it sees. This facility is called *IP accounting*.

Q.11. How can we configure the Kernel for IP Accounting?

Ans. The Linux IP accounting feature is very closely related to the Linux firewall softw The places you want to collect accounting data are the same places that you would be inter in performing firewall filtering: into and out of a network host, and in the software that the routing of datagrams.

To activate the Linux IP accounting feature, you should first see if your Linux kern configured for it. Check to see if the */proc/net/ip_acct* file exists. If it does, your kernel alr supports IP accounting. If it doesn't, you must build a new kernel, ensuring that you an "Y" to the options in 2.0 and 2.2 series kernels:

Networking options :

- Network firewalls
- TCP/IP networking

....

- **IP :** accounting

or in 2.4 series kernels :

Networking options :

- Network packet filtering (replaces ipchains)

Q.12. What do you mean by configuring IP Accounting?

Ans. Because IP accounting is closely related to IP firewall, the same tool was designate configure it, so ipfwadm, ipchains or iptables are used to configure IP accounting. The comm syntax is very similar to that of the firewall rules.

The general syntax for IP accounting with ipfwadm is :

```
# ipfwadm -A [direction] [command] [parameters]
```

The direction argument is new. This is simply coded as in, out, or both. These direct are from the perspective of the Linux machine itself, so it means data coming into the mach from a network connection and out means data that is being transmitted by this host network connection. The both direction is the sum of both the incoming and outgoing direct

The general command syntax for ipchains and iptables is :

```
# ipchains -A chain rule-specification
# iptables -A chain rule-specification
```

The ipchains and iptables commands allow you to specify direction in a manner consistent with the firewall rules. IP Firewall Chains don't allow you to configure a rule aggregates both directions, but it does allow you to configure rules in the forward chain the older implementation did not.

The commands are much the same as firewall rules, except that the policy rules do apply here. We can add, insert, delete, and list accounting rules. In the

f ipchains and iptables, all valid rules are accounting rules, and any command that doesn't pecify the *-j* option performs accounting only.

Q.13. In how many ways accounting can be done? Explain each in detail.

Ans. The rule specification parameters for IP accounting are the same as those used for IP firewall. These are what we use to define precisely what network traffic we wish to count and total.

1. Accounting by Address : Imagine we have a Linux-based router that serves two epartments at the Virtual Brewery. The router has two Ethernet devices, *eth0* and *eth1*, each f which services a department; and a PPP device, *ppp0*, that connects us via a high-speed erial link to the main campus of the Groucho Marx University.

Let's also imagine that for billing purposes we want to know the total traffic generated by ach of the departments across the serial link, and for management purposes we want to know ne total traffic generated between the two departments.

The following table shows the interface addresses we will use in our example :

iface	address	netmask
eth0	172.16.3.0	255.255.255.0
eth1	172.16.4.0	255.255.255.0

To answer the question, "How much data does each department generate on the PPP nk?" We could use a rule that looks like this :

```
# ipfwadm -A both -a -W ppp0 -S 172.16.3.0/24 -b
# ipfwadm -A both -a -W ppp0 -S 172.16.4.0/24 -b
```

or :

```
# ipchains -A input -i ppp0 -d 172.16.3.0/24
# ipchains -A output -i ppp0 -s 172.16.3.0/24
# ipchains -A input -i ppp0 -d 172.16.4.0/24
# ipchains -A output -i ppp0 -s 172.16.4.0/24
```

and with iptables :

```
# iptables -A FORWARD -i ppp0 -d 172.16.3.0/24
# iptables -A FORWARD -o ppp0 -s 172.16.3.0/24
# iptables -A FORWARD -i ppp0 -d 172.16.4.0/24
# iptables -A FORWARD -o ppp0 -s 172.16.4.0/24
```

The first half of each of these set of rules say, "Count all data traveling in either direction ross the interface named ppp0 with a source or destination (remember the function of the flag in ipfwadm and iptables) address of 172.16.3.0/24." The second half of each ruleset is e same, but for the second Ethernet network at our site.

To answer the second question, "How much data travels between the two departments?", need a rule that looks like this :

```
# ipfwadm -A both -a -S 172.16.3.0/24 -D 172.16.4.0/24 -b
```

or :

```
# ipchains -A forward -s 172.16.3.0/24 -d 172.16.4.0/24 -b
```

or :

```
# iptables -A FORWARD -s 172.16.3.0/24 -d 172.16.4.0/24
# iptables -A FORWARD -s 172.16.4.0/24 -d 172.16.3.0/24
```

. These rules will count all datagrams with a source address belonging to one department networks and a destination address belonging to the other.

2. Accounting by Service Port : Let's suppose we also want a better idea of exactly sort of traffic is being carried across our PPP link. We might, for example, want to know much of the link the FTP, smtp, and World Wide Web services are consuming.

A script of rules to enable us to collect this information might look like :

```
#!/bin/sh
# Collect FTP, smtp and www volume statistics for data carried on
# PPP link using ipfwadm
#
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 ftp ftp-data
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 smtp
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 www
```

or :

```
#!/bin/sh
# Collect ftp, smtp and www volume statistics for data carried on
# PPP link using ipchains
#
ipchains -A input -i ppp0 -p tcp -s 0/0 ftp-data:ftp
ipchains -A output -i ppp0 -p tcp -d 0/0 ftp-data:ftp
ipchains -A input -i ppp0 -p tcp -s 0/0 smtp
ipchains -A output -i ppp0 -p tcp -d 0/0 smtp
ipchains -A input -i ppp0 -p tcp -s 0/0 www
ipchains -A output -i ppp0 -p tcp -d 0/0 www
```

There are a couple of interesting features to this configuration. Firstly, we've specified protocol. When we specify ports in our rules, we must also specify a protocol because TCP UDP provide separate sets of ports. Since all of these services are TCB-based, we've specified as the protocol. Secondly, we've specified the two services ftp and ftp-data in command. ipfwadm allows you to specify single ports, ranges of ports, or arbitrary lists of The ipchains command allows either single ports or ranges of ports, which is what we've here. The syntax "ftp-data:ftp" means "ports ftp-data (20) through ftp (21)," and is how we ranges of ports in both ipchains and iptables. When you have a list of ports in an accounting rule, it means that any data received for any of the ports in the list will cause the data added to that entry's totals. Remembering that the FTP service uses two ports, the command port and the data transfer port, we've added them together to total the FTP traffic. Lastly, we specified the source address as "0/0," which is special notation that matches all addresses required by both the ipfwadm and ipchains commands in order to specify ports.

3. Accounting of ICMP Datagrams : The ICMP protocol does not use service port numbers and is therefore a little bit more difficult to collect details on. ICMP uses a number of different types of datagrams. Many of these are harmless and normal, while others should only be seen under special circumstances. Sometimes people with too much time on their hands attempt to maliciously disrupt the network access of a user by generating large numbers of ICMP messages. This is commonly called *ping flooding*. While IP accounting cannot do anything to prevent the problem, we can at least put accounting rules in place that will show us if anybody has been trying.

ICMP doesn't use ports as TCP and UDP do. Instead ICMP has ICMP message types. We an build rules to account for each ICMP message type. To do this, we place the ICMP message nd type number in place of the port field in the ipfwadm accounting commands.

An IP accounting rule to collect information about the volume of ping data that is being ent to you or that you are generating might look like :

```
# ipfwadm -A both -a -P icmp -S 0/0 8
# ipfwadm -A both -a -P icmp -S 0/0 0
# ipfwadm -A both -a -P icmp -S 0/0 0xff
```

or, with ipchains :

```
# ipchains -A forward -p icmp -s 0/0 8
# ipchains -A forward -p icmp -s 0/0 0
# ipchains -A forward -p icmp -s 0/0 -f
```

or, with iptables :

```
# iptables -A FORWARD -m icmp -p icmp —sports echo-request
# iptables -A FORWARD -m icmp -p icmp —sports echo-reply
# iptables -A FORWARD -m icmp -p icmp -f
```

The first rule collects information about the "ICMP Echo Request" datagrams (ping equests), and the second rule collects information about the "ICMP Echo Reply" datagrams ing replies). The third rule collects information about ICMP datagram fragments. This is a ick similar to that described for fragmented TCP and UDP datagrams.

4. Accounting by Protocol : Let's now imagine that we are interested in knowing how uch of the traffic on our link is TCP, UDP, and ICMP. We would use rules like the following :

```
# ipfwadm -A both -a -W ppp0 -P tcp -D 0/0
# ipfwadm -A both -a -W ppp0 -P udp -D 0/0
# ipfwadm -A both -a -W ppp0 -P icmp -D 0/0
```

or :

```
# ipchains -A forward -i ppp0 -p tcp -d 0/0
# ipchains -A forward -i ppp0 -p udp -d 0/0
# ipchains -A forward -i ppp0 -p icmp -d 0/0
```

or :

```
# iptables -A FORWARD -i ppp0 -m tcp -p tcp
# iptables -A FORWARD -o ppp0 -m tcp -p tcp
# iptables -A FORWARD -i ppp0 -m udp -p udp
# iptables -A FORWARD -o ppp0 -m udp -p udp
# iptables -A FORWARD -i ppp0 -m icmp -p icmp
# iptables -A FORWARD -o ppp0 -m icmp -p icmp
```

With these rules in place, all of the traffic flowing across the ppp0 interface will be analyzed determine whether it is TCP, UDP, or IMCP traffic, and the appropriate counters will be dated for each. The iptables example splits incoming flow from outgoing flow as its syntax mands it.

14. How can we use IP Accounting Results?

ns. It is all very well to be collecting this information, but how do we actually get to see it? To ew the collected accounting data and the configured accounting rules, we use our firewall

configuration commands, asking them to list our rules. The packet and byte counters fo of our rules are listed in the output.

The ipfwadm, ipchains, and iptables commands differ in how accounting data is ha so we will treat them independently.

1. Listing Accounting Data with ipfwadm : The most basic means of listi accounting data with the ipfwadm command is to use it like this :

```
# ipfwadm -A -l
IP accounting rules
   pkts bytes dir prot source              destination          ports
   9833 2345K i/o all 172.16.3.0/24        anywhere             n/a
   56527 33M i/o all 172.16.4.0/24         anywhere             n/a
```

This will tell us the number of packets sent in each direction. If we use the ex output format with the *-e* option, we are also supplied the options and applicable in names. Most of the fields in the output will be self-explanatory, but the following may

dir : The direction in which the rule applies. Expected values here are in, out, or i/o, m both ways.

prot : The protocols to which the rule applies.

opt : A coded form of the options we use when invoking ipfwadm.

ifname : The name of the interface to which the rule applies.

ifaddress : The address of the interface to which the rule applies.

By default, ipfwadm displays the packet and byte counts in a shortened form, rou the nearest thousand (K) or million (M). We can ask it to display the collected data i units by using the expanded option as follows :

```
# ipfwadm -A -l -e -x
```

2. Listing Accounting Data with ipchains : The ipchains command will not our accounting data (packet and byte counters) unless we supply it the -v argume simplest means of listing our accounting data with the ipchains is to use it like this:

```
# ipchains -L -v
```

Again, just as with ipfwadm, we can display the packet and byte counters in units the expanded output mode. The ipchains uses the -x argument for this:

```
# ipchains -L -v -x
```

3. Listing Accounting Data with iptables : The iptables command behaves very s to the ipchains command. Again, we must use the -v when listing to our rules to accounting counters. To list our accounting data, we would use :

```
# iptables -L -v
```

Just as for the ipchains command, you can use the -x argument to show the o expanded format with unit figures.

Q.15. What do you mean by IP Masquerade? [Imp

Ans. You don't have to have a good memory to remember a time when only large organ could afford to have a number of computers networked together by a LAN. Today technology has dropped so much in price that two things have happened. First, LANs commonplace, even in many household environments. Certainly many Linux users two or more computers connected by some Ethernet. Second, network resources, par IP addresses, are now a scarce resource and while they used to be free, they are n bought and sold.

Most people with a LAN will probably also want an Internet connection that every computer on the LAN can use. The IP routing rules are quite strict in how they deal with this situation. Traditional solutions to this problem would have involved requesting an IP network address, perhaps a class C address for small sites, assigning each host on the LAN an address from this network and using a router to connect the LAN to the Internet.

In a commercialized Internet environment, this is quite an expensive proposition. First, you'd be required to pay for the network address that is assigned to you. Second, you'd probably have to pay your Internet Service Provider for the privilege of having a suitable route to your network put in place so that the rest of the Internet knows how to reach you. This might still be practical for companies, but domestic installations don't usually justify the cost.

Q.16. What do you mean by Network Address Translation (NAT)? [Important]

Ans. NAT describes the process of modifying the network addresses contained with datagram headers while they are in transit. This might sound odd at first, but we'll show that it is ideal for solving the problem we've just described and many have encountered. IP masquerade is the name given to one type of network address translation that allows all of the hosts on a private network to use the Internet at the price of a single IP address.

IP masquerading allows you to use a private (reserved) IP network address on your LAN and have your Linux-based router perform some clever, real-time translation of IP addresses and ports. When it receives a datagram from a computer on the LAN, it takes note of the type of datagram it is, "TCP," "UDP," "ICMP," etc., and modifies the datagram so that it looks like it was generated by the router machine itself (and remembers that it has done so). It then transmits the datagram onto the Internet with its single connected IP address. When the destination host receives this datagram, it believes the datagram has come from the routing host and sends any reply datagrams back to that address. When the Linux masquerade router receives a datagram from its Internet connection, it looks in its table of established masqueraded connections to see if this datagram actually belongs to a computer on the LAN, and if it does, it reverses the modification it did on the forward path and transmits the datagram to the LAN computer.

A simple example is illustrated in Fig. 3:

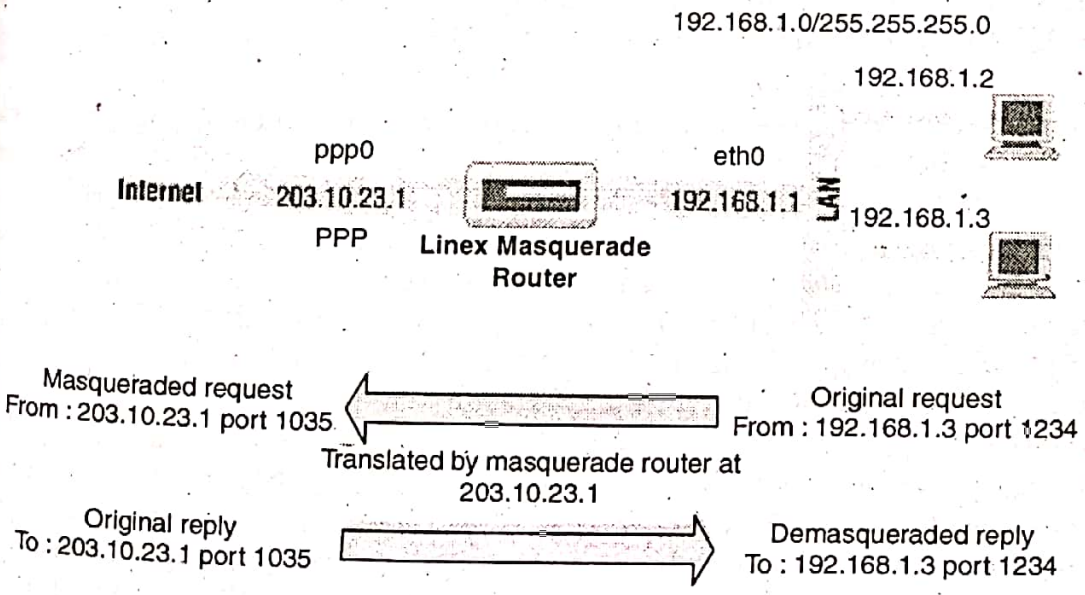


Fig. 3. A typical IP masquerade configuration

We have a small Ethernet network using one of the reserved network addresses. network has a Linux-based masquerade router providing access to the Internet. One of workstations on the network (192.168.1.3) wishes to establish a connection to the remote 209.1.106.178 on port 8888. The workstation routes its datagram to the masquerade which identifies this connection request as requiring masquerade services. It accepts datagram and allocates a port number to use (1035), substitutes its own IP address and number for those of the originating host, and transmits the datagram to the destination The destination host believes it has received a connection request from the Linux masquer host and generates a reply datagram. The masquerade host, upon receiving this datagr finds the association in its masquerade table and reverses the substation it performed on outgoing datagram. It then transmits the reply datagram to the originating host.

The local host believes it is speaking directly to the remote host. The remote host kn nothing about the local host at all and believes it has received a connection from the Li masquerade host. The Linux masquerade host knows these two hosts are speaking to e other and on what ports, and performs the address and port translations necessary to al communication.

Q.17. What are the Side Effects and Fringe Benefits of IP masquerade?

[Importa

Ans. The IP masquerade facility comes with its own set of side effects, some of which useful and some of which might become bothersome.

None of the hosts on the supported network behind the masquerade router are ever direc seen; consequently, you need only one valid and routable IP address to allow all hosts to m network connections out onto the Internet. This has a downside; none of those hosts are visi from the Internet and you can't directly connect to them from the Internet; the only h visible on a masqueraded network is the masquerade machine itself. This is important wh you consider services such as mail or FTP. It helps determine what services should be provi by the masquerade host and what services it should proxy or otherwise treat specially.

Second, because none of the masqueraded hosts are visible, they are relatively protect from attacks from outside; this could simplify or even remove the need for firewall configurat on the masquerade host. You shouldn't rely too heavily on this, though. Your whole netwo will be only as safe as your masquerade host, so you should use firewall to protect it if secur is a concern.

Third, IP masquerade will have some impact on the performance of your networking typical configurations this will probably be barely measurable. If you have large numbers active masquerade sessions, though, you may find that the processing required at t masquerade machine begins to impact your network throughput. IP masquerade must do good deal of work for each datagram compared to the process of conventional routing. Th 386SX16 machine you have been planning on using as a masquerade machine supporting dial-up link to the Internet might be fine, but don't expect too much if you decide you want use it as a router in your corporate network at Ethernet speeds.

Last, some network services just won't work through masquerade, or at least not without a lot of help. Typically, these are services that rely on incoming sessions to work, such as so types of Direct Communications Channels (DCC), features in IRC, or certain types of rea and audio multicasting services. Some of these services have specially developed kernel modul to provide solutions for these. For others, it is possible that you will find no support, so aware, it won't be suitable in all situations.

Q.18. How can we configure the Kernel for IP Masquerade?

Ans. To use the IP masquerade facility, your kernel must be compiled with masquerade support. You must select the following options when configuring a 2.2 series kernel :

Networking options :

- Network firewalls
- TCP/IP networking
- IP : firewalling
- IP : masquerading
  - Protocol-specific masquerading support will be built as modules.
- IP : ipautofw masq support
- IP : ICMP masquerading

Note that some of the masquerade support is available only as a kernel module. This means that you must ensure that you "make modules" in addition to the usual "make zImage" when building your kernel.

The 2.4 series kernels no longer offer IP masquerade support as a kernel compile time option. Instead, you should select the network packet filtering option :

Networking options :

- Network packet filtering (replaces ipchains)

In the 2.2 series kernels, a number of protocol-specific helper modules are created during kernel compilation. Some protocols begin with an outgoing request on one port, and then expect an incoming connection on another. Normally these cannot be masqueraded, as there is no way of associating the second connection with the first without peering inside the protocols themselves. The helper modules do just that; they actually look inside the datagrams and allow masquerading to work for supported protocols that otherwise would be impossible to masquerade. The supported protocols are :

Module	Protocol
ip_masq_ftp	FTP
ip_masq_irc	IRC
ip_masq_raudio	RealAudio
ip_masq_cuseeme	CU-See-Me
ip_masq_vdolive	For VDO Live
ip_masq_quake	IdSoftware's Quake

You must load these modules manually using the insmod command to implement them. Note that these modules cannot be loaded using the kerneld daemon. Each of the modules takes an argument specifying what ports it will listen on. For the RealAudio(TM) module you might use :

```
# insmod ip_masq_raudio.o ports=7070,7071,7072
```

The ports you need to specify depend on the protocol. An IP masquerade mini-HOWTO written by Ambrose Au explains more about the IP masquerade modules and how to configure them.

The *netfilter* package includes modules that perform similar functions. For example, to provide connection tracking of FTP sessions, you'd load and use the *ip_conntrack_ftp* and *ip_nat_ftp.o* modules.

Q.19. How can we configure IP Masquerade?

Ans. Masquerade rules are a special class of filtering rule. You can masquerade only datagra that are received on one interface that will be routed to another interface. To configure masquerade rule you construct a rule very similar to a firewall forwarding rule, but w special options that tell the kernel to masquerade the datagram. The ipfwadm command u the *-m* option, ipchains uses -j MASQ, and iptables uses -j MASQUERADE to indicate th datagrams matching the rule specification should be masqueraded.

Let's look at an example. A computing science student at Groucho Marx University ha number of computers at home internetworked onto a small Ethernet-based local area netwo She has chosen to use one of the reserved private Internet network addresses for her netwo She shares her accommodation with other students, all of whom have an interest in using t Internet. Because student living conditions are very frugal, they cannot afford to use permanent Internet connection, so instead they use a simple dial-up PPP Internet connectio They would all like to be able to share the connection to chat on IRC, surf the Web, and retrie files by FTP directly to each of their computers — IP masquerade is the answer.

The student first configures a Linux machine to support the dial-up link and to act a router for the LAN. The IP address she is assigned when she dials up isn't important. S configures the Linux router with IP masquerade and uses one of the private network address for her LAN: 192.168.1.0. She ensures that each of the hosts on the LAN has a default rou pointing at the Linux router.

The following ipfwadm commands are all that are required to make masquerading wo in her configuration :

```
# ipfwadm -F -p deny
# ipfwadm -F -a accept -m -S 192.168.1.0/24 -D 0/0
```

or with ipchains :

```
# ipchains -P forward -j deny
# ipchains -A forward -s 192.168.1.0/24 -d 0/0 -j MASQ
```

or with iptables :

```
# iptables -t nat -P POSTROUTING DROP
# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Now whenever any of the LAN hosts try to connect to a service on a remote host, the datagrams will be automatically masqueraded by the Linux masquerade router. The first ru in each example prevents the Linux machine from routing any other datagrams and also ad some security.

To list the masquerade rules you have created, use the -l argument the ipfwadm command.

To list the rule we created earlier we use :

```
# ipfwadm -F -l -e
```

which should display something like :

```
# ipfwadm -F -l -e
IP firewall forward rules, default policy: accept
pkts bytes type prot opt tosa tosx ifname ifaddress ...
   0 0 acc/m all —— 0xFF 0x00 any any ...
```

The "/m" in the output indicates this is a masquerade rule.

To list the masquerade rules with the ipchains command, use the -L argument. If we list the rule we created earlier with ipchains, the output will look like :

```
# ipchains -L
Chain input (policy ACCEPT) :
Chain forward (policy ACCEPT) :
target prot opt source destination ports
MASQ all —— 192.168.1.0/24 anywhere n/a

Chain output (policy ACCEPT):
Any rules with a target of MASQ are masquerade rules.
Finally, to list the rules using iptables you need to use:
# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target prot opt source destination

Chain POSTROUTING (policy DROP)
target prot opt source destination
MASQUERADE all — anywhere anywhere MASQUERADE

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```

Again, masquerade rules appear with a target of MASQUERADE.

Q.20. How can we set Timing Parameters for IP Masquerade?

Ans. When each new connection is established, the IP masquerade software creates an association in memory between each of the hosts involved in the connection. You can view these associations at any time by looking at the */proc/net/ip_masquerade* file. These associations will timeout after a period of inactivity, though.

You can set the timeout values using the ipfwadm command. The general syntax for this is :

```
ipfwadm -M -s <tcp> <tcpfin> <udp>
```

and for the ipchains command it is :

```
ipchains -M -S <tcp> <tcpfin> <udp>
```

The *iptables* implementation uses much longer default timers and does not allow you to set them.

Each of these values represents a timer used by the IP masquerade software and are in units of seconds. The following table summarizes the timers and their meanings :

Name	Description
tcp	TCP session timeout. How long a TCP connection may remain idle before the association for it is removed?
tcpfin	TCP timeout after FIN. How long an association will remain after a TCP connection has been disconnected?
udp	UDP session timeout. How long a UDP connection may remain idle before the association for it is removed?

❐❐❐

UNIT—4

THE NIS, NFS AND SYSTEM BACKUP AND RECOVERY

Q.1. Define NIS (Network Information System)?

Ans. When you're running a local area network, your overall goal is usually to provide an environment for your users that make the network transparent. An important stepping stone is keeping vital data such as user account information synchronized among all hosts. This provides users with the freedom to move from machine to machine without the inconvenience of having to remember different passwords and copy data from one machine to another. Data that is centrally stored doesn't need to be replicated, so long as there is some convenient means of accessing it from a network-connected host. By storing important administrative information centrally, you can make ensure consistency of that data, increase flexibility for the users by allowing them to move from host to host in a transparent way, and make the system administrator's life much easier by maintaining a single copy of information to maintain when required.

DNS serves a limited range of information, the most important being the mapping between hostname and IP address. For other types of information, there is no such specialized service. Moreover, if you manage only a small LAN with no Internet connectivity, setting up DNS may not seem to be worth the trouble.

This is why Sun developed the *Network Information System* (NIS). NIS provides generic database access facilities that can be used to distribute, for example, information contained in the *passwd* and *groups* files to all hosts on your network. This makes the network appear as a single system, with the same accounts on all hosts. Similarly, you can use NIS to distribute the hostname information from */etc/hosts* to all machines on the network.

NIS is based on RPC, and comprises a server, a client-side library, and several administrative tools. Originally, NIS was called *Yellow Pages*, or YP, which is still used to refer to it. Unfortunately, the name is a trademark of British Telecom, which required Sun to drop that name. As things go, some names stick with people, and so YP lives on as a prefix to the names of most NIS-related commands such as ypserv and ypbind.

Q.2. How can we get acquainted with NIS?

Ans. NIS keeps database information in files called *maps*, which contain key-value pairs. An example of a key-value pair is a user's login name and the encrypted form of their login password. Maps are stored on a central host running the NIS server, from which clients may retrieve the information through various RPC calls. Quite frequently, maps are stored in DBM files.

The maps themselves are usually generated from master text files such as */etc/hosts* or */etc/passwd*. For some files, several maps are created, one for each search key type. For instance, you may search the *hosts* file for a hostname as well as for an IP address. Accordingly, two NIS maps are derived from it, called *hosts.byname* and *hosts.byaddr*. Table 1 lists common maps and the files from which they are generated.

Table 1. Some Standard NIS Maps and Corresponding Files

Master File	Map(s)	Description
/etc/hosts	*hosts.byname, hosts.byaddr*	Maps IP addresses to host names
/etc/networks	*networks.byname, networks.byaddr*	Maps IP network addresses to network names
/etc/passwd	*passwd.byname, passwd.byuid*	Maps encrypted passwords to user login names
/etc/group	*group.byname, group.bygid*	Maps Group IDs to group names
/etc/services	*services.byname, services.bynumber*	Maps service descriptions to service names
/etc/rpc	*rpc.byname, rpc.bynumber*	Maps Sun RPC service numbers to RPC service names
/etc/protocols	*protocols.byname, protocols.bynumber*	Maps protocol numbers to protocol names
/usr/lib/aliases	*mail.aliases*	Maps mail aliases to mail alias names

You may find support for other files and maps in other NIS packages. These usually contain information for applications, such as the *bootparams* map that is used by Sun's bootparamd server.

For some maps, people commonly use *nicknames*, which are shorter and therefore easier to type. Note that these nicknames are understood only by ypcat and ypmatch, two tools for checking your NIS configuration. To obtain a full list of nicknames understood by these tools, run the following command :

```
$ ypcat -x
Use "passwd" for "passwd.byname"
Use "group" for "group.byname"
Use "networks" for "networks.byaddr"
Use "hosts" for "hosts.byaddr"
Use "protocols" for "protocols.bynumber"
Use "services" for "services.byname"
Use "aliases" for "mail.aliases"
Use "ethers" for "ethers.byname"
```

The NIS server program is traditionally called ypserv. For an average network, a single server usually suffices; large networks may choose to run several of these on different machines and different segments of the network to relieve the load on the server machines and routers. These servers are synchronized by making one of them the *master server*, and the others *slave servers*. Maps are created only on the master server's host. From there, they are distributed to all slaves.

We have been talking very vaguely about "networks." There's a distinctive term in NIS that refers to a collection of all hosts that share part of their system configuration data through NIS: the *NIS domain*. Unfortunately, NIS domains have absolutely nothing in common with the domains we encountered in DNS.

NIS domains have a purely administrative function. They are mostly invisible to users except for the sharing of passwords between all machines in the domain. Therefore, the name given to an NIS domain is relevant only to the administrators. Usually, any name will do, as long as it is different from any other NIS domain name on your local network. For instance, the administrator at the Virtual Brewery may choose to create two NIS domains, one for the Brewery itself, and one for the Winery, which she names *brewery* and *winery* respectively. Another quite common scheme is to simply use the DNS domain name for NIS as well.

To set and display the NIS domain name of your host, you can use the domainname command. When invoked without any argument, it prints the current NIS domain name; to set the domain name, you must become the superuser :

```
# domainname brewery
```

NIS domains determine which NIS server an application will query. For instance, the login program on a host at the Winery should, of course, query only the Winery's NIS server (or one of them, if there are several) for a user's password information, while an application on a Brewery host should stick with the Brewery's server.

One mystery now remains to be solved: how does a client find out which server to connect to? The simplest approach would use a configuration file that names the host on which to find the server. However, this approach is rather inflexible because it doesn't allow clients to use different servers (from the same domain, of course) depending on their availability. Therefore, NIS implementations rely on a special daemon called ypbind to detect a suitable NIS server in their NIS domain. Before performing any NIS queries, an application first finds out from ypbind which server to use.

ypbind probes for servers by broadcasting to the local IP network; the first to respond is assumed to be the fastest one and is used in all subsequent NIS queries. After a certain interval has elapsed, or if the server becomes unavailable, ypbind probes for active servers again.

Dynamic binding is useful only when your network provides more than one NIS server. Dynamic binding also introduces a security problem. ypbind blindly believes whoever answers, whether it be a humble NIS server or a malicious intruder. Needless to say, this becomes especially troublesome if you manage your password databases over NIS. To guard against this, the Linux ypbind program provides you with the option of probing the local network to find the local NIS server, or configuring the NIS server hostname in a configuration file.

Q.3. Differentiate between NIS and NIS+?

Ans. NIS and NIS+ share little more than their name and a common goal. NIS+ is structured entirely differently from NIS. Instead of a flat namespace with disjoint NIS domains, NIS+ uses a hierarchical namespace similar to that of DNS. Instead of maps, so-called *tables* are used that are made up of rows and columns, in which each row represents an object in the NIS+ database and the columns cover properties of the objects that NIS+ knows and cares about. Each table for a given NIS+ domain comprises those of its parent domains. In addition, an entry in a table may contain a link to another table. These features make it possible to structure information in many ways.

[Important]

NIS+ additionally supports secure and encrypted RPC, which helps greatly to solve the security problems of NIS. Traditional NIS has an RPC Version number of 2, while NIS+ is Version 3.

Q.4. What is Client Side of NIS?

Ans. If you are familiar with writing or porting network applications, you may notice that most of the NIS maps listed previously correspond to library functions in the C library. For

[Important]

instance, to obtain *passwd* information, you generally use the *getpwnam* and *getpwuid* functions, which return the account information associated with the given username or numerical user ID, respectively. Under normal circumstances, these functions perform the requested lookup on the standard file, such as */etc/passwd*.

An NIS-aware implementation of these functions, however, modifies this behaviour and places an RPC call to the NIS server, which looks up the username or user ID. This happens transparently to the application. The function may treat the NIS data as though it has been appended to the original *passwd* file so both sets of information are available to the application and used, or as though it has completely replaced it so that the information in the local *passwd* is ignored and only the NIS data is used.

For traditional NIS implementations, there were certain conventions for which maps were replaced and which were appended to the original information. Some, like the *passwd* maps, required kludgy modifications of the *passwd* file which, when done incorrectly, would open up security holes. To avoid these pitfalls, NYS and the GNU *libc* use a general configuration scheme that determines whether a particular set of client functions uses the original files, NIS, or NIS+, and in which order.

Q.5. How can we run an NIS Server?

Ans. If an NIS server is running on your network, you won't have to set up your own. Note that if you are just going to experiment with the server, make sure you don't set it up for an NIS domain name that is already in use on your network. This may disrupt the entire network service and make a lot of people very unhappy and very angry.

There are two possible NIS server configurations: master and slave. The slave configuration provides a live backup machine, should your master server fail. The server documentation will explain the differences, should you wish to configure a slave server.

There are currently two NIS servers freely available for Linux: one contained in Tobias Reber's *yps* package, and the other in Peter Eriksson's *ypserv* package. It doesn't matter which one you run.

After installing the server program (ypserv) in */usr/sbin*, you should create the directory that is going to hold the map files your server is to distribute. When setting up an NIS domain for the *brewery* domain, the maps would go to */var/yp/brewery*. The server determines whether it is serving a particular NIS domain by checking if the map directory is present. If you are disabling service for some NIS domain, make sure to remove the directory as well.

Maps are usually stored in DBM files to speed up lookups. They are created from the master files using a program called makedbm (for Tobias's server) or dbmload (for Peter's server).

Transforming a master file into a form that dbmload can parse usually requires some awk or sed magic, which tends to be a little tedious to type and hard to remember. Therefore, Peter Eriksson's *ypserv*package contains a Makefile (called *ypMakefile*) that manages the conversion of the most common master files for you. You should install it as *Makefile* in your map directory and edit it to reflect the maps you want the NIS server to share. Towards the top of the file, you'll find the *all* target that lists the services ypserv offers. By default, the line looks something like this :

all : ethers hosts networks protocols rpc services passwd group netid

If you don't want to produce, for example, the *ethers.byname* and *ethers.byaddr* maps, simply remove the *ethers* prerequisite from this rule. To test your setup, you can start with just one or two maps, like the *services.** maps.

After editing the *Makefile*, while in the map directory, type **make**. This will automatic generate and install the maps. You have to make sure to update the maps whenever change the master files, otherwise the changes will remain invisible to the network.

Q.6. What is the main security flaw in NIS Server? How can we overcome from the [Importa

Ans. NIS used to have a major security flaw: it left your password file readable by virtua anyone in the entire Internet, which made for quite a number of possible intruders. As long an intruder knew your NIS domain name and the address of your server, he could simply se it a request for the *passwd.byname* map and instantly receive all your system's encrypt passwords. With a fast password-cracking program like crack and a good dictionary, guessi at least a few of your users' passwords is rarely a problem.

This is what the *securenets* option is all about. It simply restricts access to your NIS serv to certain hosts, based on their IP addresses or network numbers. The latest versi of ypserv implements this feature in two ways. The first relies on a special configuration f called */etc/ypserv.securenets* and the second conveniently uses the */etc/hosts.allow* and /et *hosts.deny* files. Thus, to restrict access to hosts from within the Brewery, their netwo manager would add the following line to *hosts.allow* :

ypserv : 172.16.2.

This would let all hosts from IP network *172.16.2.0* access the NIS server. To shut out other hosts, a corresponding entry in *hosts.deny* would have to read :

ypserv : ALL

IP numbers are not the only way you can specify hosts or networks in *hosts.allow* an *hosts.deny*. However, be warned that you *cannot* use host or domain names for the *ypserv* entr If you specify a hostname, the server tries to resolve this hostname — but the resolver in tu calls ypserv, and you fall into an endless loop.

To configure *securenets* security using the */etc/ypserv.securenets* method, you need to creat its configuration file, */etc/ypserv.securenets*. This configuration file is simple in structure. Eac line describes a host or network of hosts that will be allowed access to the server. Any addres not described by an entry in this file will be refused access. A line beginning with a # will b treated as a comment. Example 1 shows what a simple */etc/ypserv.securenets* would look like

Example 1. Sample ypserv.securenets File

```
# allow connections from local host — necessary
host 127.0.0.1
# same as 255.255.255.255 127.0.0.1
#
# allow connections from any host on the Virtual Brewery network
255.255.255.0   172.16.1.0
#
```

The first entry on each line is the netmask to use for the entry, with *host* being treated a a special keyword meaning "netmask 255.255.255.255." The second entry on each line is the address to which to apply the netmask.

A third option is to use the secure portmapper instead of the *securenets* option in ypserv The secure portmapper (*portmap-5.0*) uses the *hosts.allow* scheme as well, but offers this fo all RPC servers, not just ypserv. However, you should not use both the *securenets* option an the secure portmapper at the same time, because of the overhead this authorization incurs

Q.7. What is Network File System (NFS)? Explain its main features.

Ans. The Network File System (NFS) is probably the most prominent network service using RPC. It allows you to access files on remote hosts in exactly the same way you would access local files. A mixture of kernel support and user-space daemons on the client side, along with an NFS server on the server side, makes this possible. This file access is completely transparent to the client and works across a variety of server and host architectures.

NFS offers a number of useful features :

- Data accessed by all users can be kept on a central host, with clients mounting this directory at boot time. For example, you can keep all user accounts on one host and have all hosts on your network mount */home* from that host. If NFS is installed beside NIS, users can log into any system and still work on one set of files.
- Data consuming large amounts of disk space can be kept on a single host. For example, all files and programs relating to LaTeX and METAFONT can be kept and maintained in one place.
- Administrative data can be kept on a single host. There is no need to use rcp to install the same stupid file on 20 different machines.

Q.8. Define the basic working of NFS. Explain.

Ans. Let's have a look at how NFS works. First, a client tries to mount a directory from a remote host on a local directory just the same way it does a physical device. However, the syntax used to specify the remote directory is different. For example, to mount */home* from host *vlager* to */users* on *vale*, the administrator issues the following command on *vale* :

```
# mount -t nfs vlager:/home /users
```

mount will try to connect to the rpc.mountd mount daemon on *vlager* via RPC. The server will check if *vale* is permitted to mount the directory in question, and if so, return it a file handle. This file handle will be used in all subsequent requests to files below */users*.

When someone accesses a file over NFS, the kernel places an RPC call to rpc.nfsd (the NFS daemon) on the server machine. This call takes the file handle, the name of the file to be accessed, and the user and group IDs of the user as parameters. These are used in determining access rights to the specified file. In order to prevent unauthorized users from reading or modifying files, user and group IDs must be the same on both hosts.

On most Unix implementations, the NFS functionality of both client and server is implemented as kernel-level daemons that are started from user space at system boot. These are the *NFS Daemon* (rpc.nfsd) on the server host, and the *Block I/O Daemon* (biod) on the client host. To improve throughput, biod performs asynchronous I/O using read-ahead and write-behind; also, several rpc.nfsd daemons are usually run concurrently.

The current NFS implementation of Linux is a little different from the classic NFS in that the server code runs entirely in user space, so running multiple copies simultaneously is more complicated. The current rpc.nfsd implementation offers an experimental feature that allows limited support for multiple servers. Olaf Kirch developed kernel-based NFS server support featured in 2.2 Version Linux kernels. Its performance is significantly better than the existing userspace implementation.

Q.9. How can we prepare NFS?

Ans. Before you can use NFS, be it as server or client, you must make sure your kernel has NFS support compiled in. Newer kernels have a simple interface on the *proc* filesystem for this, the */proc/filesystems* file, which you can display using cat :

```
$ cat /proc/filesystems
        minix
        ext2
        msdos
nodev proc
nodev nfs
```

If *nfs* is missing from this list, you have to compile your own kernel with NFS enabled perhaps you will need to load the kernel module if your NFS support was compiled as a mod

Q.10. How can we mount an NFS Volume?

Ans. The mounting of NFS volumes closely resembles regular file systems. Invoke mount u the following syntax :

```
# mount -t nfs nfs_volume local_dir options
```

nfs_volume is given as *remote_host:remote_dir*. Since this notation is unique to N filesystems, you can leave out the *-t nfs* option.

There are a number of additional options that you can specify to mount upon mountin NFS volume. These may be given either following the *-o* switch on the command line or in options field of the */etc/fstab* entry for the volume. In both cases, multiple options are separa by commas and must not contain any whitespace characters. Options specified on the comm line always override those given in the *fstab* file.

Here is a sample entry from */etc/fstab* :

```
# volume      mount point      type options
news:/var/spool/news /var/spool/news nfs timeo=14,intr
```

This volume can then be mounted using this command :

```
# mount news:/var/spool/news
```

In the absence of an *fstab* entry, NFS mount invocations look a lot uglier. For insta suppose you mount your users' home directories from a machine named *moonshot*, which u a default block size of 4 K for read/write operations. You might increase the block size to 8 obtain better performance by issuing the command :

```
# mount moonshot:/home /home -o rsize=8192,wsize=8192
```

The following is a partial list of options you would probably want to use :

rsize=n and *wsize=n*

These specify the datagram size used by the NFS clients on read and write reque respectively. The default depends on the version of kernel, but is normally 1,024 bytes.

timeo=n

This sets the time (in tenths of a second) the NFS client will wait for a request to compl The default value is 7 (0.7 seconds). What happens after a timeout depends on whether use the *hard* or *soft* option.

hard

Explicitly mark this volume as hard-mounted. This is on by default. This option causes server to report a message to the console when a major timeout occurs and continues tr indefinitely.

soft

Soft-mount (as opposed to hard-mount) the driver. This option causes an I/O error t reported to the process attempting a file operation when a major timeout occurs.

intr

Allow signals to interrupt an NFS call. Useful for aborting when the server doesn't respond.

Except for *rsize* and *wsize*, all of these options apply to the client's behaviour if the server should become temporarily inaccessible. They work together in the following way: Whenever the client sends a request to the NFS server, it expects the operation to have finished after a given interval (specified in the *timeout* option). If no confirmation is received within this time, a so-called *minor timeout* occurs, and the operation is retried with the timeout interval doubled. After reaching a maximum timeout of 60 seconds, a *major timeout* occurs.

Q.11. What do you mean by *soft-mounted* and *hard-mounted*? Define.

Ans. By default, a major timeout causes the client to print a message to the console and start all over again, this time with an initial timeout interval twice that of the previous cascade. Potentially, this may go on forever. Volumes that stubbornly retry an operation until the server becomes available again are called *hard-mounted*. The opposite variety, called *soft-mounted*, generate an I/O error for the calling process whenever a major timeout occurs. Because of the write-behind introduced by the buffer cache, this error condition is not propagated to the process itself before it calls the *write* function the next time, so a program can never be sure that a write operation to a soft-mounted volume has succeeded at all.

Whether you hard- or soft-mount a volume depends partly on taste but also on the type of information you want to access from a volume. For example, if you mount your X programs by NFS, you certainly would not want your X session to go berserk just because someone brought the network to a grinding halt by firing up seven copies of Doom at the same time or by pulling the Ethernet plug for a moment. By hard-mounting the directory containing these programs, you make sure that your computer waits until it is able to re-establish contact with your NFS server. On the other hand, non-critical data such as NFS-mounted news partitions or FTP archives may also be soft-mounted, so if the remote machine is temporarily unreachable or down, it doesn't hang your session. If your network connection to the server is flaky or goes through a loaded router, you may either increase the initial timeout using the *timeo* option or hard-mount the volumes. NFS volumes are hard-mounted by default.

Hard mounts present a problem because, by default, the file operations are not interruptible. Thus, if a process attempts, for example, a write to a remote server and that server is unreachable, the user's application hangs and the user can't do anything to abort the operation. If you use the *intr* option in conjuction with a hard mount, any signals received by the process interrupt the NFS call so that users can still abort hanging file accesses and resume work (although without saving the file).

Usually, the rpc.mountd daemon in some way or other keeps track of which directories have been mounted by what hosts. This information can be displayed using the showmount program, which is also included in the NFS server package :

```
# showmount -e moonshot
Export list for localhost:
/home <anon clnt> ,

# showmount -d moonshot
Directories on localhost:
/home
```

```
# showmount -a moonshot
All mount points on localhost:
localhost:/home
```

Q.12. What do you mean by NFS Daemons? Explain. [Import

Ans. If you want to provide NFS service to other hosts, you have to run the rpc.nfsd rpc.mountd daemons on your machine. As RPC-based programs, they are not managed by i but are started up at boot time and register themselves with the portmapper; therefore have to make sure to start them only after rpc.portmap is running. Usually, you'd use some like the following example in one of your network boot scripts :

```
if [ -x /usr/sbin/rpc.mountd ]; then
   /usr/sbin/rpc.mountd; echo -n " mountd"
fi
if [ -x /usr/sbin/rpc.nfsd ]; then
   /usr/sbin/rpc.nfsd; echo -n " nfsd"
fi
```

The ownership information of the files an NFS daemon provides to its clients us contains only numerical user and group IDs. If both client and server associate the same and group names with these numerical IDs, they are said to their share uid/gid space example, this is the case when you use NIS to distribute the *passwd* information to all on your LAN.

On some occasions, however, the IDs on different hosts do not match. Rather than upd the uids and gids of the client to match those of the server, you can use the rpc.ugidd map daemon to work around the disparity. Using the *map_daemon* option explained a little you can tell rpc.nfsd to map the server's uid/gid space to the client's uid/gid space with th of the rpc.ugidd on the client. Unfortunately, the rpc.ugidd daemon isn't supplied on all ma Linux distributions, so if you need it and yours doesn't have it, you will need to compile it source.

rpc.ugidd is an RPC-based server that is started from your network boot scripts like rpc.nfsd and rpc.mountd :

```
if [ -x /usr/sbin/rpc.ugidd ]; then
   /usr/sbin/rpc.ugidd; echo -n " ugidd"
fi
```

Q.13. What is *exports* File? What is its use in Linux?

Ans. The server determines the type of access that is allowed to the server's files. The *exports* file lists the filesystems that the server will make available for clients to mount use.

By default, rpc.mountd disallows all directory mounts, which is a rather sensible setti If you wish to permit one or more hosts to NFS-mount a directory, you must *export* it, thi specify it in the *exports* file. A sample file may look like this :

```
# exports file for vlager
/home vale(rw) vstout(rw) vlight(rw)
/usr/X11R6 vale(ro) vstout(ro) vlight(ro)
/usr/TeX vale(ro) vstout(ro) vlight(ro)
/ vale(rw,no_root_squash)
```

```
/home/ftp (ro)
```

Each line defines a directory and the hosts that are allowed to mount it. A hostname is usually a fully qualified domain name but may additionally contain the * and ? wildcards, which act the way they do with the Bourne shell. For instance, lab*.foo.com matches *lab01.foo.com* as well as *laboratory.foo.com*. The host may also be specified using an IP address range in the form *address/netmask*. If no hostname is given, as with the */home/ftp* directory in the previous example, any host matches and is allowed to mount the directory.

When checking a client host against the *exports* file, rpx.mountd looks up the client's hostname using the *gethostbyaddr* call. With DNS, this call returns the client's canonical hostname, so you must make sure not to use aliases in *exports*. In an NIS environment the returned name is the first match from the hosts database, and with neither DNS or NIS, the returned name is the first hostname found in the *hosts*file that matches the client's address.

The hostname is followed by an optional comma-separated list of flags, enclosed in parentheses. Some of the values these flags may take are :

secure : This flag insists that requests be made from a reserved source port, i.e., one that is less than 1,024. This flag is set by default.

insecure : This flag reverses the effect of the *secure* flag.

ro : This flag causes the NFS mount to be read-only. This flag is enabled by default.

rw : This option mounts file hierarchy read-write.

root_squash : This security feature denies the superusers on the specified hosts any special access rights by mapping requests from uid 0 on the client to the uid 65534 (that is, -2) on the server. This uid should be associated with the user *nobody*.

no_root_squash : Don't map requests from uid 0. This option is on by default, so superusers have superuser access to your system's exported directories.

link_relative : This option converts absolute symbolic links (where the link contents start with a slash) into relative links. This option makes sense only when a host's entire filesystem is mounted; otherwise, some of the links might point to nowhere, or even worse, to files they were never meant to point to. This option is on by default.

link_absolute : This option leaves all symbolic links as they are (the normal behaviour for Sun-supplied NFS servers).

map_identity : This option tells the server to assume that the client uses the same uids and gids as the server. This option is on by default.

map_daemon : This option tells the NFS server to assume that client and server do not share the same uid/gid space. rpc.nfsd then builds a list that maps IDs between client and server by querying the client's rpc.ugidd daemon.

map_static : This option allows you to specify the name of a file that contains a static map of uids and gids. For example, map_static=/etc/nfs/vlight.map would specify the */etc/nfs/vlight.map* file as a uid/gid map.

map_nis : This option causes the NIS server to do the uid and gid mapping.

anonuid* and *anongid : These options allow you to specify the uid and gid of the anonymous account. This is useful if you have a volume exported for public mounts.

Any error in parsing the *exports* file is reported to syslogd's *daemon* facility at level notice whenever rpc.nfsd or rpc.mountd is started up.

Q.14. Give various versions of Kernel-Based NFS for Server Support?

Ans. Kernel-Based NFSv2 Server Support : The user-space NFS server traditionally in Linux works reliably but suffers performance problems when overworked. This is prim because of the overhead the system call interface adds to its operation, and because it compete for time with other, potentially less important, user-space processes.

You need to build a 2.2.0 kernel with the kernel-based NFS daemon included in ord make use of the tools. You can check if your kernel has the NFS daemon included by lookin see if the */proc/sys/sunrpc/nfsd_debug* file exists. If it's not there, you may have to the rpc.nfsd module using the modprobe utility.

The kernel-based NFS daemon uses a standard */etc/exports* configuration file. The pac supplies replacement versions of the rpc.mountd and rpc.nfsd daemons that you start the same way as their userspace daemon counterparts.

Kernel-Based NFSv3 Server Support : The version of NFS that has been most comm used is NFS Version 2. Technology has rolled on ahead and it has begun to show weakn that only a revision of the protocol could overcome. Version 3 of the Network File Sy supports larger files and filesystems, adds significantly enhanced security, and offers a nu of performance improvements that most users will find useful.

Olaf Kirch and Trond Myklebust are developing an experimental NFSv3 server featured in the developer Version 2.3 kernels and a patch is available against the 2.2 k source. It builds on the Version 2 kernel-based NFS daemon.

Q.15. What do you mean by the term "Backups"? Why we make backup?

Ans. Bugs, accidents, natural disasters, and attacks on your system cannot be predicted. despite your best efforts, they can't be prevented. But if you have backups, you can com your current system and your backed-up system, and you can restore your system to a k state. Even if you lose your entire computer - to fire, for instance - with a good set of bac you can restore the information after you have purchased or borrowed a replacement mac Insurance can cover the cost of a new CPU and disk drive, but your data is something th many cases can never be replaced.

Importance of Make Backups : Backups are important only if you value the work you do on your computer. If you use your computer as a paperweight, then you don't ne make backups.

You also don't need to back up a computer that only uses read-only storage, such as ROM. A variant of this is Sun's "dataless" client workstations, in which the operating sy is installed from a CD-ROM and never modified. When configured this way, the compu local hard disk is used as an accelerator and a cache, but it is not used to store data you want to archive. Sun specifically designs its operating system for these machines so that do not need backups.

On the other hand, if you ever turn your computer on and occasionally modify data then you *must* make a copy of that information if you want to recover it in the event of a

A taxonomy of computer failures : Years ago, making daily backups was a com practice because computer hardware would often fail for no obvious reason. A backup w only protection against data loss.

Today, hardware failure is still a good reason to back up your system. In 1990, many disk companies gave their drives two- or three-year guarantees; many of those drive failing now. Even though today's state-of-the-art hard disk drives might come with five warranties, they too will fail one day!

Such a failure might not be years away, either. In the fall of 1993, one of the authors bought a new 1.7GB hard drive to replace a 1.0 GB unit. The files were copied from the older drive to the newer one, and then the older unit was reformatted and given to a colleague. The next week, the 1.7GB unit failed. Luckily, there was a backup.

Q.16. What are different reasons that make backups so important? [Important]

Ans. Backups are important for a number of other reasons :

User error : Users - especially novice users - accidentally delete their files. A user might type rm * -i instead of typing rm -i *. Making periodic backups protects users from their own mistakes, because the deleted files can be restored. Mistakes aren't limited to novices, either. More than one expert has accidentally overwritten a file by issuing an incorrect editor or compiler command, or accidentally trashed an entire directory by mistyping a wildcard to the shell.

System-staff error : Sometimes your system staff may make a mistake. For example, a system administrator deleting old accounts might accidentally delete an active one.

Hardware failure : Hardware breaks, often destroying data in the process: disk crashes are not unheard of. If you have a backup, you can restore the data on a different computer system.

Software failure : Application programs occasionally have hidden flaws that destroy data under mysterious circumstances. If you have a backup and your application program suddenly deletes half of your 500 x 500-cell spreadsheet, you can telephone the vendor and provide them with the dataset that caused the program to misbehave. You can also reload your data to try a different approach (and a different spreadsheet!).

Electronic break-ins and vandalism : Computer crackers sometimes alter or delete data. Unfortunately, they seldom leave messages telling you whether they changed any information - and even if they do, you can't trust them! If you suffer a break-in, you can compare the data on your computer after the break-in with the data on your backup to determine if anything was changed. Items that have changed can be replaced with originals.

Theft : Computers are expensive and easy to sell. For this reason, small computers - especially laptops - are often stolen. Cash from your insurance company can buy you a new computer, but it can't bring back your data. Not only should you make a backup, you should take it out of your computer and store it in a safe place, so that if the computer is stolen, at least you'll have your data.

Natural disaster : Sometimes rain falls and buildings are washed away. Sometimes the earth shakes and buildings are demolished. Fires are also very effective at destroying the places where we keep our computers. Mother Nature is inventive and not always kind. As with theft, your insurance company can buy you a new computer, but it can't bring back your data.

Other disasters : Sometimes Mother Nature isn't to blame: planes crash into buildings; gas pipes leak and cause explosions; and sometimes building-maintenance people use disk-drive cabinets as temporary saw horses. We even know of one instance in which EPA inspectors came into a building and found asbestos in the A/C ducts, so they forced everyone to leave within 10 minutes, and then sealed the building for several months!

Archival information : Backups provide archival information that lets you compare current versions of software and databases with older ones. This capability lets you determine what you've changed - intentionally or by accident. It also provides an invaluable resource if

you ever need to go back and reconstruct the history of a project, either as an academic or. to provide evidence in a court case.

Q.17. Which directories should we must backup and which not?

Ans. In general, there are some directories that you want to back up:

- **/etc**

contains all of your core configuration files. This includes your network config system name, firewall rules, users, groups, and other global system items.

- **/var**

contains information used by your systems daemons (services) includin configurations, DHCP leases, mail spool files, HTTP server files, db2 instance config and others.

- **/home**

contains the default user home directories for all of your users. This includes their p settings, downloaded files, and other information your users don't want to lose.

- **/root**

is the home directory for the root user.

- **/opt**

is where a lot of non-system software will be installed. IBM software goes i OpenOffice, JDKs, and other software is also installed here by default.

There are directories that you should consider *not* backing up.

- **/proc**

should never be backed up. It is not a real-file system, but rather a virtualized view running kernel and environment. It includes files such as /proc/kcore, which is a virtu of the entire running memory. Backing these up only wastes resources.

- **/dev**

contains the file representations of your hardware devices. If you are planning to to a blank system, then you can back up /dev. However, if you are planning to restor installed Linux base, then backing up /dev will not be necessary.

There are two schools of thought concerning computer-backup systems :

1. Back up everything that is unique to your system, including all user files, any databases that you might have modified (such as */etc/passwd* and */etc/tty* important system directories (such as */bin* and */usr/bin)* that are especially imp or that you may have modified.
2. Back up everything, because restoring a complete system is easier than restor incomplete one, and tape is cheap.

We recommend the second school of thought. While some of the information you bac already "backed up" on the original distribution disks or tape you used to load them ont hard disk, distribution disks or tapes sometimes get lost. Furthermore, as your system programs get installed in reserved directories such as */bin* and */usr/bin* , security hol discovered and patched, and other changes occur. If you've ever tried to restore your s after a disaster, you know how much easier the process is when everything is in the place.

Imagine having to reapply 75 vendor "jumbo patches" by hand, plus all the little se patches you got off the net and derived from this book, plus all the tweaks to opt performance - for each system you manage.

For this reason, we recommend that you store *everything* from your system (and that means everything necessary to reinstall the system from scratch - every last file) onto backup media at regular, predefined intervals. How often you do this depends on the speed of your backup equipment and the amount of storage space allocated for backups. You might want to do a total backup once a week, or you might want to do it only twice a year.

Q.18. Define various types of Backups possible. Explain in detail. **[Important]**

Ans. There are three basic types of backups :

A *day-zero backup*

Makes a copy of your original system. When your system is first installed, before people have started to use it, back up every file and program on the system. Such backups can be invaluable after a break-in.

A *full backup*

Makes a copy of every file on your computer to the backup device. This method is similar to a day-zero backup, except that you do it on a regular basis.

An *incremental backup*

Makes a copy to the backup device of only those items in a filesystem that have been modified after a particular event (such as application of a vendor patch) or date (such as the date of the last full backup).

Full backups and incremental backups work together. One common backup strategy is :

- Make a full backup on the first day of every other week.
- Make an incremental backup every evening of everything that has been modified since the last full backup.

Q.19. Discuss the logging system in Unix.

Ans. Unix systems have a very flexible and powerful logging system, which enables you to record almost anything you can imagine and then manipulate the logs to retrieve the information you require.

Many versions of UNIX provide a general-purpose logging facility called *syslog*. Individual programs that need to have information logged send the information to syslog.

Unix *syslog* is a host-configurable, uniform system logging facility. The system uses a centralized system logging process that runs the program **/etc/syslogd** or **/etc/syslog**.

The operation of the system logger is quite straightforward. Programs send their log entries to *syslogd*, which consults the configuration file /etc/syslogd.conf or /etc/syslog and, when a match is found, writes the log message to the desired log file.

There are four basic syslog terms that you should understand :

Term	Description
Facility	The identifier used to describe the application or process that submitted the log message. Examples are mail, kernel, and ftp.
Priority	An indicator of the importance of the message. Levels are defined within syslog as guidelines, from debugging information to critical events.
Selector	A combination of one or more facilities and levels. When an incoming event matches a selector, an action is performed.
Action	What happens to an incoming message that matches a selector. Actions can write the message to a log file, echo the message to a console or other device, write the message to a logged in user, or send the message along to another syslog server.

Q.20. What are the facilities and priorities of Syslog in Unix?

Ans. Syslog Facilities : Here are the available facilities for the selector. Not all facilities present on all versions of UNIX.

Facility	Description
Auth	Activity related to requesting name and password (getty, su, login)
Authpriv	Same as auth but logged to a file that can only be read by selected users
Console	Used to capture messages that would generally be directed to the system console
Cron	Messages from the cron system scheduler
Daemon	System daemon catch-all
ftp	Messages relating to the ftp daemon
Kern	Kernel messages
local0.local7	Local facilities defined per site
Lpr	Messages from the line printing system
Mail	Messages relating to the mail system
Mark	Pseudo event used to generate timestamps in log files
News	Messages relating to network news protocol (nntp)
Ntp	Messages relating to network time protocol
User	Regular user processes
Uucp	UUCP subsystem

Syslog Priorities : The syslog priorities are summarized in the following table :

Priority	Description
Emerg	Emergency condition, such as an imminent system crash, usually broadcast to all users
Alert	Condition that should be corrected immediately, such as a corrupted system database
Crit	Critical condition, such as a hardware error
Err	Ordinary error
Warning	Warning
Notice	Condition that is not an error, but possibly should be handled in a special way
Info	Informational message
Debug	Messages that are used when debugging programs
None	Pseudo level used to specify not to log messages.

Q.21. Define the logging actions and command in Unix.

Ans. Logging Actions : The action field specifies one of five actions :

1. Log message to a file or a device. For example, /var/log/lpr.log or /dev/console.

2. Send a message to a user. You can specify multiple usernames by separating them with commas (e.g., root, amrood).
3. Send a message to all users. In this case, the action field consists of an asterisk (e.g., *).
4. Pipe the message to a program. In this case, the program is specified after the UNIX pipe symbol (|).
5. Send the message to the syslog on another host. In this case, the action field consists of a hostname, preceded by an at sign (e.g., @tutorialspoint.com)

The logger Command : UNIX provides the **logger** command, which is an extremely useful command to deal with system logging. The **logger** command sends logging messages to the syslogd daemon, and consequently provokes system logging.

This means we can check from the command line at any time the **syslogd** daemon and its configuration. The logger command provides a method for adding one-line entries to the system log file from the command line.

The format of the command is :

```
logger [-i] [-f file] [-p priority] [-t tag] [message]...
```

Here is the detail of the parameters :

Option	Description
-f filename	Use the contents of file filename as the message to log.
-i	Log the process ID of the logger process with each line.
-p priority	Enter the message with the specified priority (specified selector entry); the message priority can be specified numerically, or as a facility.priority pair. The default priority is user.notice.
-t tag	Mark each line added to the log with the specified tag.
Message	The string arguments whose contents are concatenated together in the specified order, separated by the space

Q.22. What do you mean by dump and restore command in Unix? Explain their working. **[Important]**

Ans. Dump tends to look at file systems rather than individual files. dump examines files on an ext2 filesystem and determines which files need to be backed up. These files are copied to the given disk, tape, or other storage medium for safe keeping. A dump that is larger than the output medium is broken into multiple volumes. On most media, the size is determined by writing until an end-of-media indication is returned."

The companion program to dump is restore, which is used to restore files from a dump image.

The restore command performs the inverse function of dump. A full backup of a file system may be restored and subsequent incremental backups layered on top of it. Single files and directory subtrees may be restored from full or partial backups.

Both dump and restore can be run across the network, so you can back up or restore from remote devices. dump and restore work with tape drives and file devices providing a wide range of options. However, both are limited to the ext2 and ext3 file systems. If you are working with JFS, Reiser, or other file systems, you will need to use a different utility, such as tar.

Backing up with dump : Running a backup with dump is fairly straightforward. The following command does a full backup of Linux with all ext2 and ext3 file systems to a SCSI tape device :

```
dump 0f /dev/nst0 /boot
dump 0f /dev/nst0 /
```

In this example, our system has two file systems. One for /boot and another for / — a common configuration. They must be referenced individually when a backup is executed. The /dev/nst0 refers to the first SCSI tape, but in a non-rewind mode. This ensures that the volumes are put back-to-back on the tape.

An interesting feature of dump is its built-in incremental backup functionality. In the example above, the 0 indicates a level 0, or base-level, backup. This is the full system backup that you would do periodically to capture the entire system. On subsequent backups you can use other numbers (1-9) in place of the 0 to change the level of the backup. A level 1 backup would save all of the files that had changed since the level 0 backup was done. Level 2 would backup everything that had changed from level 1 and so on. The same function can be done with tar, using scripting, but it requires the script creator to have a mechanism to determine when the last backup was done. dump has its own mechanism, writing an update file (/etc/dumpupdates) when it performs a backup. The update file is reset whenever a level 0 backup is run. Subsequent levels leave their mark until another level 0 is done. If you are doing a tape-based backup, dump will automatically track multiple volumes.

Restoring with restore : To restore information saved with dump, the restore command is used. Like tar, dump has the ability to list (-t) and compare archives to current files (-C). Where you must be careful with dump is in restoring data. There are two very different approaches, and you must use the correct one to have predictable results.

Rebuild (-r)

Remember that dump is designed with file systems in mind more than individual files. Therefore, there are two different styles of restoring files. To rebuild a file system, use the -r switch. Rebuild is designed to work on an empty file system and restore it back to the saved state. Before running rebuild, you should have created, formatted, and mounted the file system. You should not run rebuild on a file system that contains files.

Here is an example of doing a full rebuild from the dump that we executed above.

```
restore -rf /dev/nst0
```

The above command needs to be run for each file system being restored.

This process could be repeated to add the incremental backups if required.

Etract (-x)

If you need to work with individual files, rather than full file systems, you must use the -x switch to extract them. For example, to extract only the /etc directory from our tape backup, use the following command :

```
restore -xf /dev/nst0 /etc
```

Interactive restore (-i)

One more feature that restore provides is an interactive mode. Using the command :

```
restore -if /dev/nst0
```

will place you in an interactive shell, showing the items contained in the archive. Typing "help" will give you a list of commands. You can then browse and select the items you wish to be extracted. Bear in mind that any files that you extract will go into your current directory.

Q.23. What are the other types of backup products available in Linux?

Ans. There are several commercial backup products available for Linux. Commercial products generally provide a convenient interface and reporting system, whereas with tools such as dump and tar, you have to roll your own. The commercial offerings are broad and offer a range of features. The biggest benefit you will gain from using a commercial package is a pre-built strategy for handling backups that you can just put to work. Commercial developers have already made many of the mistakes that you are about to, and the cost of their wisdom is cheap compared to the loss of your precious data.

1. Tivoli Storage Manager : Probably the best commercial backup and storage management utility available now for Linux is the Tivoli Storage Manager. Tivoli Storage Manager Server runs on several platforms, including Linux, and the client runs on many more platforms.

Essentially a Storage Manager Server is configured with the devices appropriate to back up the environment. Any system that is to participate in the backups loads a client that communicates with the server. Backups can be scheduled, performed manually from the Tivoli Storage Manager client interface, or performed remotely using a Web-based interface.

The policy-based nature of TSM means that central rules can be defined for backup behavior without having to constantly adjust a file list. Additionally, IBM Tivoli Storage Resource Manager can identify, evaluate, control, and predict the utilization of enterprise storage assets, and can detect potential problems and automatically apply self-healing adjustments.

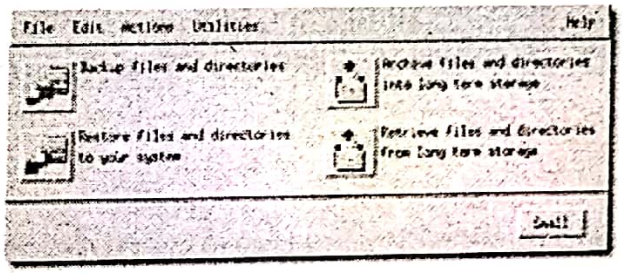


Fig. 1. Tivoli Storage Manager menu

Backups and restores are then handled through the remote device.

2. Using rsync to make a backup : The rsync utility is a very well-known piece of GPL'd software, written originally by Andrew Tridgell and Paul Mackerras. If you have a common Linux or UNIX variant, then you probably already have it installed. Rsync's specialty is efficiently synchronizing file trees across a network, but it works fine on a single machine too.

Suppose you have a directory called source, and you want to back it up into the directory destination. To accomplish that, you'd use :

```
rsync -a source/ destination/
```

(Note: I usually also add the -v (verbose) flag too so that rsync tells me what it's doing). This command is equivalent to :

```
cp -a source/. destination/
```

except that it's much more efficient if there are only a few differences.

Just to whet your appetite, here's a way to do the same thing as in the example above, but with destination on a remote machine, over a secure shell :

```
rsync -a -e ssh source/ username@remotemachine.com:/path/to/destination/
```

3. Using the —delete flag : If a file was originally in both source/ and destination/ (from an earlier rsync, for example), and you delete it from source/, you probably want it to be deleted

from destination/ on the nextrsync. However, the default behaviour is to leave the co at destination/ in place. Assuming you want rsync to delete any file from destination/ that not in source/, you'll need to use the —delete flag :

```
rsync -a —delete source/ destination/
```

4. Using cron : One of the toughest obstacles to a good backup strategy is human natur if there's any work involved, there's a good chance backups won't happen. (Witness, for exampl how rarely my roommate's home PC was backed up before I created this system). Fortunatel there's a way to harness human laziness: make cron do the work.

To run the rsync-with-backup command from the previous section every morning at 4:2 AM, for example, edit the root cron table: (as root)

```
crontab -e
```

Then add the following line :

```
20 4 * * * rsync -a —delete source/ destination/
```

Finally, save the file and exit. The backup will happen every morning at precisely 4:2 AM, and root will receive the output by email. Don't copy that example verbatim, though; yo should use full path names (such as /usr/bin/rsync and /home/source/) to remove any ambiguit

UNIT—5

ACTIVE DIRECTORY, LDAP

Q.1. Define Lightweight Directory Access Protocol (LDAP). Define its main goals. **[Important]**

Ans. The **Lightweight Directory Access Protocol** is an application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network. LDAP is defined in terms of ASN.1 and transmitted using BER.

Directory services may provide any organized set of records, often with a hierarchical structure, such as a corporate electronic mail directory. Similarly, a telephone directory is a list of subscribers with an address and a phone number.

LDAP was developed as a method of consolidating access, authentication, and authorization (AAA, or Triple-A) information. By itself, this is useful, because you are maintaining all of the information in one place rather than many. However, you could have accomplished the same thing using any old database. What makes LDAP especially suited to store your AAA information is that all LDAP operations take place within the context of the AAA information, rather than forcing the application to supply or interpret the context. Operations fail or succeed with no need for the application to understand the rules involved.

If you attempted to put all of the same AAA information into a database (which would be somewhat difficult because you would have to define all the standards for storage of the information, which LDAP has already done for you), then every one of your applications would have to parse that information and take it into account for each AAA operation. If you use LDAP, however, there are already methods for storage of the AAA information, although they are not yet RFC-defined, and the LDAP server rather than the application applies all of the AAA rules. This not only makes the lives of the application developers easier, it also eliminates the chance of rogue applications or users bypassing the AAA rules to directly access and modify the directory contents, except of course through traditional security compromises.

So, our goal here is to build such an AAA infrastructure, with LDAP at its core. The majority of our authentication information already exists, in the form of user accounts, and for the purposes of this article, we will assume that those user accounts are on Unix machines. Our goal, then, is to put that user account information into LDAP, manage it entirely from LDAP, and then use it as the root of AAA operations in other applications. Once we have the authentication information in place, we must then add the access and authorization information.

This article will deal with the first task, replacing the standard methods of maintaining Unix accounts with methods using LDAP. Later, we will hopefully provide examples of web-based maintenance of our LDAP data and some applications which might take further advantage of this newly centralized data.

Q.2. What are the tools of LDAP?

Ans. Nearly every modern language has an LDAP API; as such, there is a near infinite availability of tools. Fortunately, because of the simplicity of the protocol, most of the APIs

work quite similarly. To get started immediately, we're going to start with the command-l tools (of which there are multiple versions), because they're straightforward and ubiquito

There are only three basic types of LDAP operations, and each basic type has a few subtyp interrogation (search, compare), updating (add, modify, rename, delete), and binding/cont (bind, unbind, abandon). Notice there is no "read" operation; if you want to read an entry, y use a search operation to retrieve it.

To move our UNIX accounts to LDAP, we must convert them into a form the LDAP serv can understand. Because of the simplicity of conversion from passwd file to LDIF (see belo we will leave the conversion as an exercise for the reader. Instead, we will begin with already converted user account and add it to LDAP :

```
$ ldapadd -D "cn=Directory Manager" -h
server
password: ********
dn: uid=luke,ou=People,dc=domain,dc=com
objectclass: top
objectclass: posixAccount
uid: luke
cn: Luke A. Kanies
cn: Luke Kanies
cn: Kanies, Luke
uidNumber: 100
gidNumber: 14
homeDirectory: /home/luke
userPassword: {crypt}8SYYCOBH.aIII
gecos: Luke A. Kanies
^D
adding new entry uid=luke,ou=People,dc=domain,dc=com
$
```

This operation uses two of the three basic LDAP operations, an update operation and binding operation, along with an implicit unbind. We must bind as a privileged user in order modify the directory (this example uses a special user which all iPlanet Directory Servers have).

We then provide the data for the entry we want to add, which is in LDAP Date Interchange Format (LDIF), the standard text format for LDAP data. Notice that each line has an attribute name, then a colon and a space, and the value of the attribute. It certainly makes sense that LDAP search operations would return this format, so that it could be immediately fed back into an LDAP server.

Q.3. What are the various components of an LDAP entry? **[Important]**

Ans. Let's go through the various pieces to understand it.

1. Distinguished Names : The first line is what is called the "Distinguished Name" or dn. Because this is how we tell the server what object we are working with, the dn line must always be specified first.

The dn is how an entry is uniquely referred to within an LDAP server, similar to an absolute path name or a fully qualified domain name. Notice that the dn is represented similarly to

DNS names, with the most specific information first, as opposed to path names, which have the least specific information first. Contrary to how it looks, the root of this LDAP tree, also called the "naming context," isdc=domain,dc=com, not dc=com.

There are no requirements about what you name the root of your LDAP tree, but there are two standards: either the standard I've followed here, which breaks a domain into its various domain components, or one where an organization is referred to at the top level (for example, o=domain.com). Which one you should follow will be answered differently by everyone you ask, and why you should follow it will also be answered differently. I have chosen to follow the domain component standard because it seems to be more popular these days — Sun and Microsoft have both begun recommending/requiring it. Pick the one that seems easiest and fits best with how you plan on using the data.

The rest of the dn consists of a branch in the tree, ou=People, and an attribute-value pair uniquely identifying this entry, uid=luke. This attribute-value pair, when separate from the dn, is called the "Relative Distinguished Name," or rdn, and it uniquely identifies this object at this level of the tree.

Because path names and DNS names always refer to the name of the object in question, all they need to specify is that value, but LDAP can use any attribute to create an rdn, and as a result both the attribute and its value must be specified. The dn for this entry could also be cn=Luke A. Kanies,ou=People,dc=domain,dc=com, as long as there is no one else in ou=People with "Luke A. Kanies" as a value of cn.

We choose uid here, though, because we will always guarantee that it is unique for each entry — anything else would not function correctly on our Unix systems, and conveniently iPlanet's Directory Server can be set up to disallow duplicate uid values (or any other attribute). Apparently Microsoft's Active Directory is requiring that cn be used to create the rdn, which is annoying because the value of cn is almost guaranteed not to be unique, as it is the user's full name. Thanks!

2. Object classes : Next in our entry are two objectclass attributes. These attributes define what type of object the entry is. However, the concept of an object in LDAP is extremely simple: it merely defines what attributes an entry must have and what attributes an entry is allowed to have. All object classes inherit requirements from their parent object classes and add their own. The objectclass attribute isn't a special attribute, though — in all LDAP operations it is treated exactly like other LDAP attributes, but modifying object class does determine whether the object will be acceptable to the server after the operation.

The above definition of an LDAP object is important, because it is difficult to convince yourself how simple this definition really is. Again, an object merely defines what attributes must or can be stored with an entry. I can create an object which has both the posixAccount object class and a printer object class; this combination may seem quite contradictory to you and me, but to LDAP, the data is just data, and has no meaning on its own. One of the things that makes LDAP great is that the attributes mean something to humans, but it is up to the humans working with the data to retain that meaning by naming object classes and attributes intelligently and then creating objects that actually make sense. This is more difficult than it sounds, and deciding how all of this will be done is the first step to using LDAP. Fortunately, most of the object classes you will need are part of the LDAP specification (although posix Account is part of a later RFC), so at least initially you won't have to worry too much about that.

Note that you can't just willy-nilly make up object classes and add them to an object; the server must have each object class defined for it, in what is called itsschema. If you try to add

an entry with an undefined object class, you will get a schema violation and the operation will fail. How you define an object class for the server varies from server to server, but most of them document it quite well.

In this case, we have declared that our entry will be of types top and posixAccount. The top object class merely requires that the object class attribute be present, and by definition it is the parent object class for all other LDAP object classes. Given this fact, and the fact that objects always list the object classes, they are an instance of along with all of their parent object classes, every object in an LDAP database will list top as an object class. The posixAccount object class is defined in an RFC by Luke Howard, who has done a significant amount of the work involved in allowing LDAP to store Unix accounts, and provides a means to store all of the information from a passwd file in LDAP.

3. Attributes : Attributes can have multiple values. Whether an attribute supports multiple values is stated in the definition of the attribute, but most do.

Q.4. How we can use the data in LDAP?

Ans. If you want to read an entry, you have to search for it. So, to verify that our entry is present, we will do so. There are a total of eight (yes, eight) different options for every LDAP search, but most of them have reasonable defaults and they mostly make sense. We won't be using very many of these options to start, to keep it simple.

Here's an example search :

```
$ ldapsearch -h server -b "dc=domain,dc=com" "(uid=luke)"
uid=luke,ou=People,dc=madstop,dc=com
objectclass=top
objectclass=posixAccount
uid=luke
cn=Luke A. Kanies
uidnumber=100
gidnumber=14
homedirectory=/home/luke
gecos=Luke A. Kanies
$
```

This is on a Solaris box; horror of horrors — notice that the output is not in LDIF format (equal signs are used instead of colons and spaces). Notice also that the password is not printed; this is for security reasons, in the same way that /etc/shadow is only readable by privileged users.

In our search, we included two flags and an argument. The first flag is obvious: We specify the server. The second flag specified the base for the search, similar to how one specifies the start point of a search using the find command; we could have specified any branch including the object itself. The argument to the search is called the *filter*, and it's how we specify the specific entry or entries we want. All entries matching the filter will be returned. Fortunately, the filter format is somewhat sophisticated, so you shouldn't have problems performing complex searches to find exactly what you want, but I'm only going to use basic filters to get the job done. For more information, consult a reference of some kind.

Because we know the dn of the entry, we could have also performed our search thus :

```
$ ldapsearch -s base -h server -b "uid=luke,ou=People,dc=domain,dc=com
"(objectclass=*)"
```

Because we know the dn, we can set that dn as the base of our search. If we do that, however, can change the scope of the search. By default, an LDAP query searches for objects anywhere the heirarchy beginning at the base of the search. You can also specify a scope of "one", hich only searches the next lower level of the hierarchy, and base, which searches only the se itself. I have actually noticed some discrepencies here when using iPlanet Directory Server ; usually the base of the search is returned if it matches the filter, but I have seen instances ere that is not the case. The above method is the preferred method of retrieval if you know dn of the entry, because it is faster and requires less work from the server.

Notice also that our filter has changed. We previously searched for a specific value of uid, because we are using the entry's dn as the base of our search in order to retrieve it directly use what is called an *existence filter*. When an attribute is searched for with a value of *, LDAP server returns every entry which has any value for that attribute. Because all entries e values for objectclass, using a filter of objectclass=* is the standard method for returning entries matching a given base and scope. The reason for preferring an existence filter over ore specific filter is that a specific filter causes the LDAP server to work harder, and using existence filter here provides the same result with less work for the server.

How can we create a Unix account in LDAP?

. Well, assuming you have added this entry and all other desired entries to your LDAP er, all it takes is a simple shell script to convert this data to the passwd and shadow files:

```
#!/usr/bin/bash
#
# shell script to convert from LDAP to passwd/shadow files
IFS='
  # set the internal file separator to a carriage return, in case
  # attributes have spaces or tabs in them

# look for all entries which have the posixAccount objectclass
# we have to authenticate as a privileged user in order to read the
# password
for entry in $(ldapsearch -h server -D \
"uid=sysadmin,ou=People,dc=domain,dc=com" -w password \
-b dc=domain,dc=com "(objectclass=posixAccount)"); do

  # this is the logic that prints out the passwd entry for the
  # previous entry every time we hit a DN line
  echo $line | grep "dc=madstop,dc=com" > /dev/null
  if [ $? == 0 -a ! -z "$uid" ]; then
    echo $uid:+:$uidnumber:$gidnumber:$gecos:$homedirectory:$usershell \
    >> newpasswd
    echo $userpassword >> newshadow
  fi

# this uses some trickery to cause the shell to interpret
# ldapsearch output as variable assignments; this may need
```

```
    # to be modified depending on the output of the ldapsearch co
    # but this works with the ldapsearch supplied on Solaris 8
    newline=$(echo $line | sed "s/'/'\''/g
        s/=/='/
        s/$/'/")
    eval $newline
done
# this echo line puts our last entry into the files, because ou
# method of dumping the entries is triggered by the next entry,
# and there is no next entry for the last entry
echo $uid:+:$uidnumber:$gidnumber:$gecos:$homedirectory:$usersh
>> newpasswd
echo $userpassword >> newshadow

# backup the passwd and shadow files, and put the new ones in p
for file in passwd shadow; do
    cp $file $file.bak
    cp new$file $file
done
```

Q.6. Give the overview of Protocols used in LDAP.

Ans. A client starts an LDAP session by connecting to an LDAP server, called a Di System Agent (DSA), by default on TCP port 389. The client then sends an operation to the server, and the server sends responses in return. With some exceptions, the clie not need to wait for a response before sending the next request, and the server may s responses in any order.

The client may request the following operations :

- StartTLS — use the LDAPv3 Transport Layer Security (TLS) extension for a connection
- Bind — authenticate and specify LDAP protocol version
- Search — search for and/or retrieve directory entries
- Compare — test if a named entry contains a given attribute value
- Add a new entry
- Delete an entry
- Modify an entry
- Modify Distinguished Name (DN) — move or rename an entry
- Abandon — abort a previous request
- Extended Operation — generic operation used to define other operations
- Unbind — close the connection (not the inverse of Bind)

In addition the server may send "Unsolicited Notifications" that are not response request, e.g. before it times out a connection.

A common alternative method of securing LDAP communication is using an SSL This is denoted in LDAP URLs by using the URL scheme "ldaps". The default port fo over SSL is 636. The use of LDAP over SSL was common in LDAP Version 2 (LDAPv2

was never standardized in any formal specification. This usage has been deprecated along with LDAPv2, which was officially retired in 2003.

Q.7. What is the Directory structure in LDAP?

Ans. The protocol accesses LDAP directories, which follow the 1993 edition of the X.500 model :

- An entry consists of a set of attributes.
- An attribute has a name (an *attribute type* or *attribute description*) and one or more values. The attributes are defined in a *schema* (see below).
- Each entry has a unique identifier: its *Distinguished Name* (DN). This consists of its *Relative Distinguished Name* (RDN), constructed from some attribute(s) in the entry, followed by the parent entry's DN. Think of the DN as the full file path and the RDN as its relative filename in its parent folder (e.g. if */foo/bar/myfile.txt* were the DN, then *myfile.txt* would be the RDN).

Be aware that a DN may change over the lifetime of the entry, for instance, when entries are moved within a tree. To reliably and unambiguously identify entries, a UUID might be provided in the set of the entry's *operational attributes.*

An entry can look like this when represented in LDAP Data Interchange Format (LDIF) (LDAP itself is a binary protocol) :

```
dn: cn=John Doe,dc=example,dc=com
cn: John Doe
givenName: John
sn: Doe
telephoneNumber: +1 888 555 6789
telephoneNumber: +1 888 555 1232
mail: john@example.com
manager: cn=Barbara Doe,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
```

"dn" is the distinguished name of the entry; it's neither an attribute nor a part of the entry. "cn=John Doe" is the entry's RDN (Relative Distinguished Name), and "dc=example,dc=com" is the DN of the parent entry, where "dc" denotes 'Domain Component'. The other lines show the attributes in the entry. Attribute names are typically mnemonic strings, like "cn" for common name, "dc" for domain component, "mail" for e-mail address and "sn" for surname.

A server holds a subtree starting from a specific entry, e.g. "dc=example,dc=com" and its children. Servers may also hold references to other servers, so an attempt to access "ou=department,dc=example,dc=com" could return a *referral* or *continuation reference* to a server that holds that part of the directory tree. The client can then contact the other server. Some servers also support *chaining*, which means the server contacts the other server and returns the results to the client.

LDAP rarely defines any ordering: The server may return the values of an attribute, the attributes in an entry, and the entries found by a search operation in any order. This follows from the formal definitions - an entry is defined as a set of attributes, and an attribute is a set of values, and sets need not be ordered.

Q.8. How can we authenticate in LDAP server?

Ans. The Bind operation establishes the **authentication state** for a connection. Simp can send the user's DN and password in **plaintext**, so the connection should be pr using **Transport Layer Security** (TLS). The server typically checks the password a the userPassword attribute in the named entry. Anonymous Bind (with empty D password) resets the connection to anonymous state. **SASL** (Simple Authentication and S Layer) Bind provides authentication services through a wide range of mecha e.g. **Kerberos** or the client certificate sent with TLS.

Bind also sets the LDAP protocol version. The version is an integer and at present m either 2 (two) or 3 (three), although the standard supports integers between 1 and 127 (inc in the protocol. If the client requests a version that the server does not support, the must set the result code in the bind response to the code for a protocol error. Normally should use LDAPv3, which is the default in the protocol but not always in LDAP libra

Bind had to be the first operation in a session in LDAPv2, but is not required in LD (the current LDAP version).

Q.9. What are the various operations of LDAP? [Impor

Ans. 1. Search and Compare : The Search operation is used to both search for and entries. Its parameters are :

`baseObject`

The name of the base object entry (or possibly the root) relative to which the search is performed.

`scope`

What elements below the baseObject to search. This can be BaseObject (search ju named entry, typically used to read one entry), singleLevel (entries immediately belo base DN), or wholeSubtree (the entire subtree starting at the base DN).

`filter`

Criteria to use in selecting elements within scope. For example, the (&(objectClass=person) (|(givenName=John)(mail=john*))) will select "persons" (eleme objectClassperson) where the matching rules for givenName and mail determine whethe values for those attributes match the filter assertion. Note that a common misconcepti that LDAP data is case-insensitive, whereas in fact matching rules and ordering rules deter matching, comparisons, and relative value relationships. If the example filters were req to match the case of the attribute value, an *extensible match filter* must be used, for exa (&(object Class = person) (|(given Name: case Exact Match: = John) (mail: case E Substrings Match: = john*)))

`derefAliases`

Whether and how to follow alias entries (entries that refer to other entries),

`attributes`

Which attributes to return in result entries.

`sizeLimit, timeLimit`

Maximum number of entries to return, and maximum time to allow search to run. T values, however, cannot override any restrictions the server places on size limit and limit.

`typesOnly`

Return attribute types only, not attribute values.

The server returns the matching entries and potentially continuation references. These may be returned in any order. The final result will include the result code.

The Compare operation takes a DN, an attribute name and an attribute value, and checks if the named entry contains that attribute with that value.

2. Update data : Add, Delete, and Modify DN - all require the DN of the entry that is to be changed.

Modify takes a list of attributes to modify and the modifications to each: Delete the attribute or some values, add new values, or replace the current values with the new ones.

Add operations also can have additional attributes and values for those attributes.

Modify DN (move/rename entry) takes the new RDN (Relative Distinguished Name), optionally the new parent's DN, and a flag that says whether to delete the value(s) in the entry that match the old RDN. The server may support renaming of entire directory subtrees.

An update operation is atomic: Other operations will see either the new entry or the old one. On the other hand, LDAP does not define transactions of multiple operations: If you read an entry and then modify it, another client may have updated the entry in the meantime. Servers may implement extensions that support this, though.

3. Extended operations : The Extended Operation is a generic LDAP operation that can define new operations that were not part of the original protocol specification. StartTLS is one of the most significant extensions. Other examples include the Cancel and Password Modify.

`StartTLS`

The StartTLS operation establishes Transport Layer Security (the descendant of SSL) on the connection. It can provide data confidentiality (to protect data from being observed by third parties) and/or data integrity protection (which protects the data from tampering). During TLS negotiation the server sends its X.509 certificate to prove its identity. The client may also send a certificate to prove its identity. After doing so, the client may then use SASL/EXTERNAL. By using the SASL/EXTERNAL, the client requests the server derive its identity from credentials provided at a lower level (such as TLS). Though technically the server may use any identity information established at any lower level, typically the server will use the identity information established by TLS.

Servers also often support the non-standard "LDAPS" ("Secure LDAP", commonly known as "LDAP over SSL") protocol on a separate port, by default 636. LDAPS differs from LDAP in two ways: 1) upon connect, the client and server establish TLS before any LDAP messages are transferred (without a StartTLS operation) and 2) the LDAPS connection must be closed upon TLS closure.

It should be noted that some "LDAPS" client libraries only encrypt communication, they do not check the host name against the name in the supplied certificate.

LDAPS was used with LDAPv2, because the StartTLS operation had not yet been defined. The use of LDAPS is deprecated, and modern software should only use StartTLS.

4. Abandon : The Abandon operation requests that the server abort an operation named by a message ID. The server need not honor the request. Unfortunately, neither Abandon nor a successfully abandoned operation send a response. A similar Cancel extended operation does send responses, but not all implementations support this.

5. Unbind : The Unbind operation abandons any outstanding operations and closes the connection. It has no response. The name is of historical origin, and is *not* the opposite of the Bind operation.

Clients can abort a session by simply closing the connection, but they should Unbind. Unbind allows the server to gracefully close the connection and free resources that would otherwise keep for some time until discovering the client had abandoned the connection. It also instructs the server to cancel operations that can be canceled, and not to send respons for operations that cannot be canceled.

Q.10. Define the format of LDAP URL's. Give its example.

Ans. An LDAP URL format exists, which clients support in varying degrees, and servers retu in referrals and continuation references :

```
ldap://host:port/DN?attributes?scope?filter?extensions
```

Most of the components described below are optional.

- *host* is the FQDN or IP address of the LDAP server to search.
- *port* is the network port (default port 389) of the LDAP server.
- *DN* is the distinguished name to use as the search base.
- *attributes* is a comma-separated list of attributes to retrieve.
- *scope* specifies the search scope and can be "base" (the default), "one" or "sub".
- *filter* is a search filter.
- *extensions* are extensions to the LDAP URL format.

For example, "ldap://ldap.example.com/cn=John%20Doe,dc=example,dc=com" refers to user attributes in John Doe's entry in ldap.example.com, while "ldap:///dc=exampl dc=com??sub? (givenName=John)" searches for the entry in the default server (note the trip slash, omitting the host, and the double question mark, omitting the attributes). As in oth URLs, special characters must be percent-encoded.

There is a similar non-standard ldaps: URL scheme for LDAP over SSL. This should n be confused with LDAP with TLS, which is achieved using the StartTLS operation using th standard ldap:scheme.

Q.11. What do you mean by the term "Schema"? Define it.

Ans. The contents of the entries in a subtree are governed by a **schema** known as a **directo information tree (DIT).**

The schema of a Directory Server defines a set of rules that govern the kinds of informatio that the server can hold. It has a number of elements, including :

- **Attribute Syntaxes :** Provide information about the kind of information that can b stored in an attribute.
- **Matching Rules :** Provide information about how to make comparisons againt attribute values.
- **Matching Rule Uses :** Indicate which attribute types may be used in conjunctio with a particular matching rule.
- **Attribute Types :** Define an **object identifier** (OID) and a set of names that may b used to refer to a given attribute, and associates that attribute with a syntax and set matching rules.
- **Object Classes :** Define named collections of attributes and classify them into sets required and optional attributes.
- **Name Forms :** Define rules for the set of attributes that should be included in th RDN for an entry.
- **Content Rules :** Define additional constraints about the object classes and attribute that may be used in conjunction with an entry.

. **Structure Rule :** Define rules that govern the kinds of subordinate entries that a given entry may have.

Attributes are the elements responsible for storing information in a directory, and the hema defines the rules for which attributes may be used in an entry, the kinds of values that ose attributes may have, and how clients may interact with those values.

Clients may learn about the schema elements that the server supports by retrieving an propriate subschema subentry.

The schema defines *object classes*. Each entry must have an objectClass attribute, containing med classes defined in the schema. The schema definition of the classes of an entry defines hat kind of object the entry may represent - e.g. a person, organization or domain. The object ass definitions also define the list of attributes that must contain values and the list of tributes which may contain values.

For example, an entry representing a person might belong to the classes "top" and "person". embership in the "person" class would require the entry to contain the "sn" and "cn" attributes, nd allow the entry also to contain "userPassword", "telephoneNumber", and other attributes. ince entries may have multiple ObjectClasses values, each entry has a complex of optional nd mandatory attribute sets formed from the union of the object classes it represents. bjectClasses can be inherited, and a single entry can have multiple ObjectClasses values hat define the available and required attributes of the entry itself. A parallel to the schema of n objectClass is a **class** definition and an **instance** in **Object-oriented programming**, epresenting LDAP objectClass and LDAP entry, respectively.

Directory servers may publish the directory schema controlling an entry at a base DN iven by the entry's subschemaSubentry operational attribute. (An *operational ttribute* describes operation of the directory rather than user information and is only returned rom a search when it is explicitly requested.)

Server administrators can add additional schema entries in addition to the provided schema lements. A schema for representing individual people within organizations is termed a **white ages schema**.

Q.12. What are the Requirements for the LDAP Servers?

Ans. We must be able to get the schemas (table format definitions) for both operating systems nto the server.

For UNIX a complete set of schemas is provided in /etc/openldap/schema, and by using / usr/sbin/schema2ldif followed by slapadd or ldapadd you can import whichever ones you need. f you use/etc/openldap/slapd.conf, it has an "include" statement for schemas.

The Windows LDAP server is stuffed with its proper schema out of the box. On Windows, ldapadd can *probably* be used to add a foreign schema to an already working server, given the needed authentication codes. Conversely, ldapsearch *should* deliver a complete dump of the Windows schema in a form that could be added to the OpenLDAP server, provided the UNIX side could authenticate itself to Windows A.D.

Each operating system must be able to authenticate so as to read and write arbitrary table content, including for unattended operation (nightly housekeeping).

On Windows I'm not totally clear here but I believe that the host OS instance on each machine has a Kerberos secret key to authenticate itself ("joining the domain") and an authorization for fairly broad write access to the Windows portion of the LDAP database. Also there is probably a similar authorization for the global administrator. How to get such an

authorization for a UNIX machine and/or the UNIX root user is not too obvious, particula for unattended operation. However, there are rumors that a few people have made this wo

On UNIX, GSSAPI (Kerberos) can be used for authentication, but long-lived Kerbe authentication is considered to be a security hazard, making this form unpopular for unatten operation. Normally one uses an unencrypted password (secret key) protected by UNIX permissions. In reality this is equivalent to what Windows does, both functionally and security. If the UNIX LDAP server handled Windows as well, we would have to learn wh Windows was going to present for authentication (e.g. which Kerberos principal name), a how Windows expects the LDAP server to believe in the authorization for this principal. The are rumors that this can be made to work.

Useable dumps must be made of the LDAP database.

On Windows the LDAP content is included in standard dumps, but I don't know what for it takes or how much you can hack it to repair corruption.

On UNIX you would use /usr/sbin/slapcat to dump the database as LDIF, LDAP Da Interchange Format, basically key-value pairs in ASCII or UTF-8. With the normally us HDB and BDB backends for database storage, this can be done safely with the server runnin The output would go in master-site:/h1/backup and would end up in our regular filesyste backups, similar to what we do for other UNIX databases. LDIF can be hand-edited to repa corruption, and we can put back the repaired stanzas with ldapmodify, or in case of a total lo we can restore the whole database using slapadd, not requiring that the server be function (i.e. having schemas and authentication codes) before restoration.

We are going to have to transfer our NIS-based directory to LDAP. Our administrati tools (acctadmin) depend on UNIX directory flat files.

Just like NIS, on UNIX the LDAP tables are basically representations of the tradition UNIX directory flat files, e.g. /etc/passwd. While OpenLDAP does not include migration too to translate between the flat files and the tables, there is a simple flat to LDAP migration to available on the web, and I am writing a more complete one which will also translate a table a flat file, and it can do diffs. Both these tools are independent of the vendor, that is, shou work on a Windows LDAP server, given the needed schema and authentication codes.

Q.13. What do you mean by Active Directory (AD)? **[Importa**

Ans. Active Directory (AD) is a directory service created by Microsoft for Windo domain networks. It is included in most Windows Server operating systems. Server compute that run Active Directory are called domain controllers.

Active Directory provides a central location for network administration and securi It authenticates and authorizes all users and computers in a Windows domain type network assigning and enforcing security policies for all computers and installing or updating softwa For example, when a user logs into a computer that is part of a Windows domain, Acti Directory verifies the password and specifies whether the user is a system administrator normal user.

Active Directory uses Lightweight Directory Access Protocol (LDAP) versions 2 a 3, Kerberos and DNS.

Q.14. Discuss the structure of Active Directory in detail.

Ans. 1. Objects : An Active Directory structure is a hierarchical arrangement of informati about objects. The objects fall into two broad categories: resources (e.g., printers) and securit principals (user or computer accounts and groups). Security principals are assigne unique security identifiers (SIDs).

Each object represents a single entity—whether a user, a computer, a printer, or a group—and its attributes. Certain objects can contain other objects. An object is uniquely identified by its name and has a set of attributes—the characteristics and information that the object represents— defined by a schema, which also determines the kinds of objects that can be stored in Active Directory.

Each attribute object can be used to define multiple schema objects. The schema object lets administrators extend or modify the schema when necessary. However, because each schema object is integral to the definition of Active Directory objects, deactivating or changing these objects can fundamentally change or disrupt a deployment. Schema changes automatically propagate throughout the system. Once created, an object can only be deactivated—not deleted. Changing the schema usually requires planning.

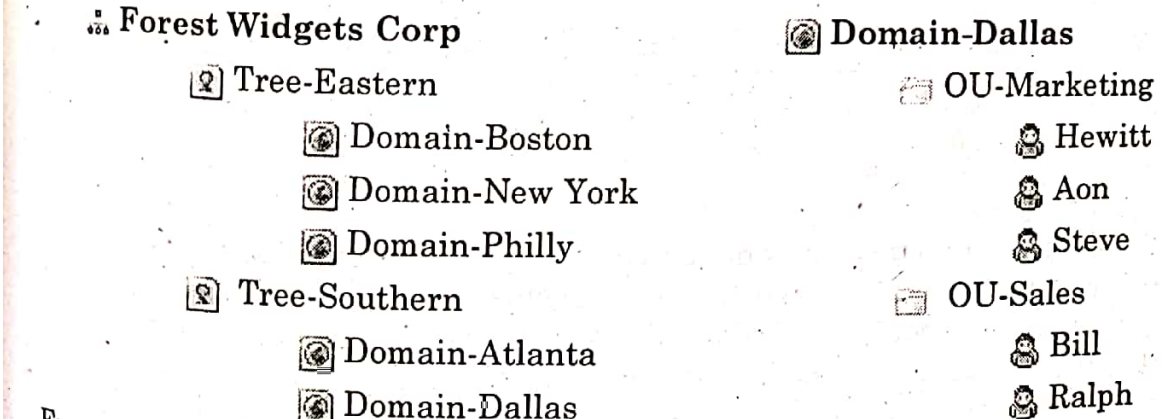
Sites : A **Site** object in Active Directory represents a geographic location that hosts networks.

2. Forests, trees, and domains : The Active Directory framework that holds the objects can be viewed at a number of levels. The forest, tree, and domain are the logical divisions in an Active Directory network.

Within a deployment, objects are grouped into domains. The objects for a single domain are stored in a single database (which can be replicated). Domains are identified by their DNS name structure, the namespace.

A tree is a collection of one or more domains and domain trees in a contiguous namespace, linked in a transitive trust hierarchy.

At the top of the structure is the *forest*. A forest is a collection of trees that share a common global catalog, directory schema, logical structure, and directory configuration. The forest represents the security boundary within which users, computers, groups, and other objects are accessible.



Example of the geographical organizing of zones of interest within trees and domains.

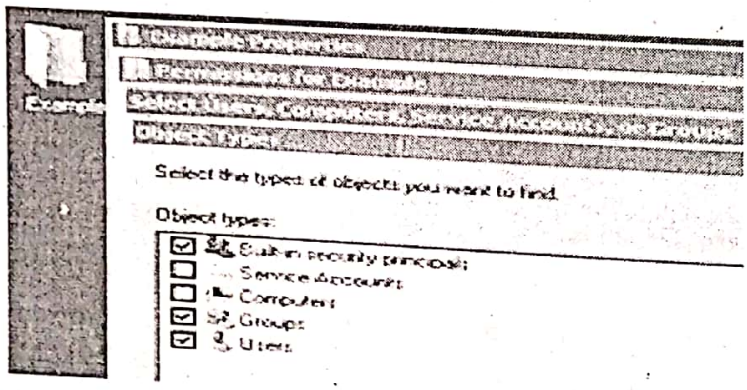
Organizational units : The objects held within a domain can be grouped into Organizational Units (OUs). OUs can provide hierarchy to a domain, ease its administration, and can resemble the organization's structure in managerial or geographical terms. OUs can contain other OUs—domains are containers in this sense. Microsoft recommends using OUs rather than domains for structure and to simplify the implementation of policies and administration. The OU is the recommended level at which to apply group policies, which are Active Directory objects formally named Group Policy Objects (GPOs), although policies can also be applied to domains or sites (see below). The OU is the level at which administrative powers are commonly delegated, but delegation can be performed on individual objects or attributes as well.

Organizational Units are an abstraction for the administrator and do not function as containers; the underlying domain is the true container. It is not possible, for example, to create user accounts with an identical username (sAMAccountName) in separate OUs, such as "fred.staff-ou.domain" and "fred.student-ou.domain", where "staff-ou" and "student-ou" are the OUs. This is so because sAMAccountName, a user object attribute, must be unique within the domain. However 2 users in different OUs can have the same Common Name (CN), the first component of the Distinguished Name (DN) of the user. Thus from the point of view of the DN, OUs do function as containers.

As the number of users in a domain increases, conventions such as "first initial, middle initial, last name" (Western order) or the reverse (Eastern order) fail for common family names like *Li (Ng)*, *Smith*or *Garcia*. Workarounds include adding a digit to the end of the username. Alternatives include creating a separate ID system of unique employee/student id numbers to use as account names in place of actual user's names, and allowing users to nominate their preferred word sequence within an acceptable use policy.

Because duplicate usernames cannot exist within a domain, account name generation poses a significant challenge for large organizations that cannot be easily subdivided into separate domains, such as students in a public school system or university who must be able to use any computer across the network.

Shadow groups : In Active Directory, organizational units cannot be assigned as owners or trustees. Only groups are selectable, and members of OUs cannot be collectively assigned rights to directory objects.



In Microsoft's Active Directory, OUs do not confer access permissions, and objects placed within OUs are not automatically assigned access privileges based on their containing OU This is a design limitation specific to Active Directory. Other competing directories such as Novell NDS are able to assign access privileges through object placement within an OU.

Active Directory requires a separate step for an administrator to assign an object in an OU as a member of a group also within that OU. Relying on OU location alone to determine access permissions is unreliable, because the object may not have been assigned to the group object for that OU.

A common workaround for an Active Directory administrator is to write a custom PowerShell or Visual Basic script to automatically create and maintain a *user group* for each OU in their directory. The scripts are run periodically to update the group to match the OU's account membership, but are unable to instantly update the security groups anytime the directory changes, as occurs in competing directories where security is directly implemented into the directory itself. Such groups are known as *Shadow Groups*. Once created, these shadow groups are selectable in place of the OU in the administrative tools.

Microsoft refers to shadow groups in the Server 2008 Reference documentation, but does not explain how to create them. There are no built-in server methods or console snap-ins for managing shadow groups.

The division of an organization's information infrastructure into a hierarchy of one or more domains and top-level OUs is a key decision. Common models are by business unit, by geographical location, by IT Service, or by object type and hybrids of these. OUs should be structured primarily to facilitate administrative delegation, and secondarily, to facilitate group policy application. Although OUs form an administrative boundary, the only true security boundary is the forest itself and an administrator of any domain in the forest must be trusted across all domains in the forest.

Q.15. What are the Physical matters of Active Directory?

Ans. *Sites* are physical (rather than logical) groupings defined by one or more IP subnets. AD also holds the definitions of connections, distinguishing low-speed (e.g., WAN, VPN) from high-speed (e.g.,LAN) links. Site definitions are independent of the domain and OU structure and are common across the forest. Sites are used to control network traffic generated by replication and also to refer clients to the nearest domain controllers. Microsoft Exchange Server 2007 uses the site topology for mail routing. Policies can also be defined at the site level.

Physically the Active Directory information is held on one or more peer domain controllers (DCs), replacing the NT PDC/BDC model. Each DC has a copy of the Active Directory. Servers joined to Active Directory that are not domain controllers are called Member Servers.

The Active Directory database is organized in *partitions*, each holding specific object types and following a specific replication pattern. AD synchronizes changes using multi-master replication. Microsoft often refers to these partitions as 'naming contexts'. The 'Schema' partition contains the definition of object classes and attributes within the Forest. The 'Configuration' partition contains information on the physical structure and configuration of the forest (such as the site topology). Both replicate to all domain controllers in the Forest. The 'Domain' partition holds all objects created in that domain and replicates only to Domain Controllers within its domain. So, for example, a user created in Domain X would be listed only in Domain X's domain controllers. A subset of objects in the domain partition replicates to domain controllers that are configured as global catalogs. Global catalog (GC) servers provide a global listing of all objects in the Forest. Global Catalog servers replicate to themselves all objects from all domains and hence, provide a global listing of objects in the forest. However, to minimize replication traffic and keep the GC's database small, only selected attributes of each object are replicated. This is called the partial attribute set (PAS). The PAS can be modified by modifying the schema and marking attributes for replication to the GC. Earlier versions of Windows used NetBIOS to communicate. Active Directory is fully integrated with DNS and requires TCP/IP—DNS. To be fully functional, the DNS server must support SRV resource records or service records.

Q.16. What is Active Directory Replication? Explain it. **[Important]**

Ans. Active Directory replication is 'pull' rather than 'push', meaning that replicas pull changes from the server where the change was effected. The *Knowledge Consistency Checker* (KCC) creates a replication topology of *site links* using the defined *sites* to manage traffic. Intrasite replication is frequent and automatic as a result of change notification, which triggers peers to begin a pull replication cycle. Intersite replication intervals are typically less frequent and do not use change notification by default, although this is configurable and can be made identical to intrasite replication.

Each link can have a 'cost' (e.g., DS3, T1, ISDN etc.) and the KCC alters the site link topology accordingly. Replication may occur transitively through several site links on same-protocol site *link bridges*, if the cost is low, although KCC automatically costs a direct site-to-site link lower than transitive connections. Site-to-site replication can be configured to occur between a *bridgehead server* in each site, which then replicates the changes to other DCs within the site. Replication for Active Directory zones is automatically configured when DNS is activated in the domain based by site.

Replication of Active Directory uses Remote Procedure Calls (RPC) over IP (RPC/IP). Between Sites SMTP can be used for replication, but only for changes in the Schema, Configuration, or Partial Attribute Set (Global Catalog) NCs. SMTP cannot be used for replicating the default Domain partition.

Q.17. Give an overview of Active Directory Database.

Ans. The Active Directory database, the *directory store*, in Windows 2000 Server uses the JET Blue-based Extensible Storage Engine (ESE98) and is limited to 16 terabytes and 2 billion objects (but only 1 billion security principals) in each domain controller's database. Microsoft has created NTDS databases with more than 2 billion objects. (NT4's Security Account Manager could support no more than 40,000 objects). Called NTDS.DIT, it has two main tables: the *data table* and the *link table*. In Windows Server 2003 a third main table was added for security descriptor single instancing.

Q.18. What are Single server operations in Active Directory? **[Important]**

Ans. Flexible Single Master Operations Roles (FSMO, sometimes pronounced "fizz-mo") operations are also known as operations master roles. Although domain controllers allow simultaneous updates in multiple places, certain operations are supported only on a single server. These operations are performed using the roles listed below :

Role Name	Scope	Description
Schema Master	1 per forest	Schema modifications
Domain Naming Master	1 per forest	Addition and removal of domains if present in root domain
PDC Emulator	1 per domain	Provides backwards compatibility for NT4 clients for PDC operations (like password changes). The PDC runs domain specific processes such as the Security Descriptor Propagator (SDPROP), and is the master time server within the domain. It also handles external trusts, the DFS consistency check, holds current passwords and manages all GPOs as default server.
RID Master	1 per domain	Allocates pools of unique identifiers to domain controllers for use when creating objects
Infrastructure Master	1 per domain/partition	Synchronizes cross-domain group membership changes. The infrastructure master cannot run on a global catalog server (GCS) (unless all DCs are also GCs, or environment consists of a single domain).

Q.19. What are the main types of Trust in Active Directory?

Ans. To allow users in one domain to access resources in another, Active Directory uses trusts. Trusts inside a forest are automatically created when domains are created. The forest sets the default boundaries of trust, and implicit, transitive trust is automatic for all domains within a forest.

Terminology :

One-way trust : One domain allows access to users on another domain, but the other domain does not allow access to users on the first domain.

Two-way trust : Two domains allow access to users on both domains.

Trusting domain : The domain that allows access to users from a trusted domain.

Trusted domain : The domain that is trusted; whose users have access to the trusting domain.

Transitive trust : A trust that can extend beyond two domains to other trusted domains in the forest.

Intransitive trust : A one way trust that does not extend beyond two domains.

Explicit trust : A trust that an admin creates. It is not transitive and is one way only.

Cross-link trust : An explicit trust between domains in different trees or in the same tree when a descendant/ancestor (child/parent) relationship does not exist between the two domains.

Shortcut : Joins two domains in different trees, transitive, one- or two-way.

Forest : Applies to the entire forest. Transitive, one- or two-way

Realm : Can be transitive or nontransitive, one- or two-way

External : Connect to other forests or non-AD domains. Nontransitive, one- or two-way.

Windows Server 2003 introduced the *forest root trust*. This trust can be used to connect Windows Server 2003 forests if they are operating at the 2003 forest functional level. Authentication across this type of trust is Kerberos based (as opposed to NTLM). Forest trusts are transitive for all the domains in the trusted forests. Forest trusts, however, are not transitive.

Q.20. What is Active Directory Lightweight Directory Service? **[Important]**

Ans. Active Directory Lightweight Directory Service (*AD LDS*), formerly known as *Active Directory Application Mode* (ADAM), is a light-weight implementation of Active Directory. AD LDS is capable of running as a service on computers running Microsoft Windows Server. AD LDS shares the code base with Active Directory and provides the same functionality as Active Directory, including an identical API, but does not require the creation of domains or domain controllers.

Like Active Directory, AD LDS provides a *Data Store* for storage of directory data and a Directory Service with an LDAP *Directory Service Interface*. Unlike Active Directory, however, multiple AD LDS instances can be run on the same server.

Q.21. How can we integrate Unix with Active Directory?

Ans. Varying levels of interoperability with Active Directory can be achieved on most Unix-like operating systems through standards-compliant LDAP clients, but these systems usually do not interpret many attributes associated with Windows components, such as Group Policy and support for one-way trusts.

Third-parties offer Active Directory integration for Unix platforms (including UNIX, Linux, Mac OS X, and a number of Java- and UNIX-based applications), including :

- *Centrify DirectControl* (Centrify Corporation) – Active Directory-compatible centralized authentication and access control

- *Centrify Express* (Centrify Corporation) – A suite of free Active Directory-compli services for centralized authentication, monitoring, file-sharing and remote acces
- *UNAB* (Computer Associates)
- *TrustBroker* (CyberSafe Limited) – An implementation of Kerberos
- *PowerBroker Identity Services*, formerly *Likewise* (BeyondTrust, formerly Likew Software) – Allows a non-Windows client to join Active Directory
- *Authentication Services* (Quest Software)
- *ADmitMac* (Thursby Software Systems)
- Samba – Can act as a domain controller

The schema additions shipped with Windows Server 2003 R2 include attributes that ma closely enough to RFC 2307 to be generally usable. The reference implementation of RFC 230 nss_ldap and pam_ldap provided by PADL.com, support these attributes directly. The defau schema for group membership complies with RFC 2307bis (proposed). Windows Server 200 R2 includes aMicrosoft Management Console snap-in that creates and edits the attributes.

An alternate option is to use another directory service such as 389 Directory Server (formerl Fedora Directory Server, FDS), eB2Bcom ViewDS v7.1 XML Enabled Directory or Su Microsystems Sun Java System Directory Server, which can perform two-way synchronizatio with AD and thus provide a "deflected" integration, as Unix and Linux clients authenticate t FDS and Windows Clients authenticate to AD. Another option is to use OpenLDAP wit its *translucent* overlay, which can extend entries in any remote LDAP server with additiona attributes stored in a local database. Clients pointed at the local database see entries containin both the remote and local attributes, while the remote database remains completely untouched

Active Directory can be automated by Powershell.

B.TECH.

SEVENTH SEMESTER EXAMINATION 2012-13

TCS-701

SYSTEM ADMINISTRATION

Time: 3 Hours *Total Marks: 100*

Note: Attempt all sections.

SECTION – A

Q.1. Attempt any four parts of the following:

(a) What is meant by multi user and multitasking operating system? **(5 × 4 = 20)**

Ans. Multiuser Operating System: A multi-user operating system is a computer operating system that allows multiple users on different computers or terminals to access a single system with one OS on it. These programs are often quite complicated and must be able to properly manage the necessary tasks required by the different users connected to it. The users will typically be at terminals or computers that give them access to the system through a network, as well as other machines on the system such as printers. A multi-user operating system differs from a single-user system on a network in that each user is accessing the same OS at different machines.

The operating system on a computer is one of the most important programs used. It is typically responsible for managing memory and processing for other applications and programs being run, as well as recognizing and using hardware connected to the system, and properly handling user interaction and data requests. On a system using a multi-user operating system this can be even more important, since multiple people require the system to be functioning properly simultaneously. This type of system is often used on mainframes and similar machines, and if the system fails it can affect dozens or even hundreds of people.

Multitasking Operating System: A multitasking operating system is any type of system that is capable of running more than one program at a time. Most modern operating systems are configured to handle multiple programs simultaneously, with the exception of some privately developed systems that are designed for use in specific business settings. As with most types of communications technology, the multitasking operating system has evolved over time, and is likely to continue evolving as communication demands keep growing in many cultures.

With older examples of the multitasking operating system, managing two or more tasks normally involved switching system resources back and forth between the two running processes. The system would execute tasks for one, freeze that program for a few seconds, and then execute tasks for the other program. While this approach did create a short time lag for the operator, this lag was usually no more than a few seconds and still offered considerable more efficiency than the older single-task operating system.

Over time, popular incarnations of the multitasking operating system were developed that used a different approach to allocating resources for each active program. This created a situation where virtually no time lag occurred at all, assuming that the equipment driving the system had adequate resources. For the end user, this meant the ability to perform several tasks simultaneously without any waiting for the system to release or redirect resources as each task completed in turn.

(b) How are file organized in UNIX? What are different file system exist in UNIX?

Ans. Please See Q. 1. of Unit-2

(c) What is the use of Inode? What information is stored in Inode?

Ans. Please See Q. 9. of Unit-2

(d) Explain various information which password file contain. How do we provide password security?

Ans. Please See Q. 20 and Q. 30 of Unit - I

(e) What are the various step involved in logging into UNIX?

Ans. Please See Q. 19 of Unit - 4

(f) What is the function of Kernel? What are different kernel boot files?

Ans. The Linux boot process consists of several stages, each represented by a different component. The following list briefly summarizes the boot process and features all the major components involved.

1. BIOS. After turning on the computer, the BIOS initializes the screen and keyboard and tests the main memory. Up to this stage, the machine does not access any mass storage media. Subsequently, the information about the current date, time, and the most important peripherals are loaded from the CMOS values. When the first hard disk and its geometry are recognized, the system control passes from the BIOS to the boot loader. If the BIOS supports network booting, it is also possible to configure a boot server that provides the boot loader. On x86 systems, PXE boot is needed. Other architectures commonly use the BOOTP protocol to get the boot loader.

2. Boot Loader. The first physical 512-byte data sector of the first hard disk is loaded into the main memory and the *boot loader* that resides at the beginning of this sector takes over. The commands executed by the boot loader determine the remaining part of the boot process. Therefore, the first 512 bytes on the first hard disk are referred to as the *Master Boot Record* (MBR). The boot loader then passes control to the actual operating system, in this case, the Linux Kernel. For a network boot, the BIOS acts as the boot loader. It gets the image to start from the boot server and starts the system. This is completely independent of local hard disks.

3. Kernel and initramfs. To pass system control, the boot loader loads both the Kernel and an initial RAM–based file system (initramfs) into memory. The contents of the initramfs can be used by the Kernel directly. initramfs contains a small executable called init that handles the mounting of the real root system. If special hardware drivers are needed b the mass storage can be accessed, they must initramfs. If the system does not have a local disk, the initramfs must provide the root file sy to the Kernel. This can be done with the help network block device like iSCSI or SAN, but it is possible to use NFS as the root device.

4. Init on initramfs. This program perform actions needed to mount the proper root file sy like providing Kernel functionality for the needed system and device drivers for mass storage contro with udev. After the root file system has been fo it is checked for errors and mounted. If thi successful, the initramfs is cleaned and the program on the root file system is executed.

5. Init. init handles the actual booting of system through several different levels provid different functionality.

Q.2. Attempt any four parts of the following: (5 × 4 =

(a) Explain IP Masqurade. How it is implemented

Ans. Please See Q. No. 15 and Q. No. 19 of Uni

(b) What is process? How UNIX identifies a proce How do you send a process in background?

Ans. For Process and identifying a proces unix Please See Q. No 5 of Unit - I

To place a foreground process in the backgro suspend the foreground process (with Ctrl-z) enter the bg command to move the process into background.

Show the status of all background and suspend jobs: **jobs**

Bring a job back into the foreground **%jobnumber**

Bring a job back into the background **%jobnumber**

(c) What is RUN Level in Unix O/S? Describe R level Script

Ans. Please See Q. No. 16 and Q. No. 17 of Uni

(d) What are Device files, Raw and Block file Unix O/S?

Ans. Please See Q. No. 6 and Q. No. 8 of Unit - 2

(c) What is the use of host and network file?

Ans. Please See Q. No. 20 of Unit - 2

(d) Explain inconfig command with example.

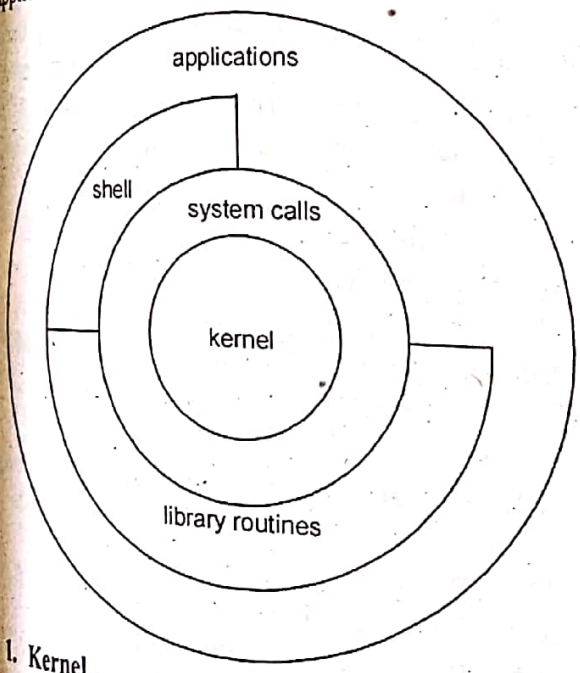
Ans. Please See Q. No. 21 and Q. No. 23 of Unit - 2

Q.3. Attempt any two parts of the following: **(10 × 2 = 20)**

(a) What are the different layers of Unix Architecture? Explain the intended purpose of each.

Ans. Unix Architecture

Unix is having Layered based architecture. It's having three important layer 1. Kernel 2. Shell 3. Application & User



1. Kernel

This is the first layer in Unix System to interact with Hardware. It's called as heart of the OS. Basic work of the kernel is process scheduling, memory managing and hardware interaction. Kernel can interact with the hardware through device drivers.

2. Shell

This is called as Command Interpreter. Interface between Kernel and User. Unix having different type of the Shell, every shell having unique feature.

3. User & Application

User interacts with the Unix system through Shell. All the application also running in this layer.

(b) What is the role of system Administrator? Explain various duties of system Administrator.

Ans. Please See Q. No. 1 of Unit - 1

(c) Explain name service in Linux operating. What is use of host and nsswitch configuration file?

Ans. Please See Q. 27 of Unit - 2

Q.4. Attempt any two parts of the following: **(10 × 2 = 20)**

(a) What are the main features of NIS? How it is configure? What is the functionality of export file in NFS?

Ans. Please See Q. 7, Q. 9, Q. 10, Q.13 of Unit - 4

(b) What you understand by IP filtering? How many ways we can do filtering? What are different utility to do IP filtering in linux? Explain with example.

Ans. Please See Q. 4 Q. 7, Q. 6 of Unit - 3

(c) Why backup is so important? Explain different tool available in Linux.

Ans. Please See Q. 15, Q. 16, Q. 23 of Unit - 4

Q.5. Attempt any two parts of the following: **(10 × 2 = 20)**

(a) How NIS is useful in local area network? What are different maps used in NIS.

Ans. Please See Q. 1, Q. 2 of Unit - 4

(b) What is use of Active Directory in windows 2000? What are its components? How LDAP worked in Active Directory?

Ans. Please See Q. 13, Q. 14, Q. 20 of Unit - 5

(c) What do mean by IP accounting? What are different ways to configure IP accounting.

Ans. Please See Q. 10, Q. 13 of Unit - 3

❑❑❑

B.Tech.

TCS-701

(UTU) SEVENTH SEMESTER EXAMINATION, 2017-18

SYSTEM ADMINISTRATION

Time : 03:00 Hours *Max Marks : 100*

1. **Attempt any Four. All questions carry equal marks.** [5×4]
   (a) What is mounting? Explain with the help of an example.
   (b) What is file system? Compare the file system of windows & UNIX .
   (c) What do you mean by the term System Administrator? Write the various goals that need to be taken care of for system and network administration.
   (d) Explain the concept of security and how we can provide security in password files.
   (e) Define IP Accounting and its configuration.

2. **Attempt any Four. All questions carry equal marks.** [5×4]
   (a) What do you mean by firewall? What different types of firewall exist? Also write what firewalls can block and what they cannot block.
   (b) Write a shell program to find out the greatest among the three numbers entered through the keyboard.
   (c) Write a shell script to accept two filenames and check if both exist. If the second filename exists, then the contents of the first filename should be appended to it. If the second filename does not exist then create a new file with the contents of the first file.
   (d) Explain User profiles, and the concept of switching the users & removing the users.
   (e) What is LINUX Scheduler? Explain the concept of Init program.

3. **Attempt any Four. All questions carry equal marks.** [5×4]
   (a) Explain the side effects and fringe benefits of IP Masquerading.
   (b) Write Short Note on following:
   (i) UFS, NFS & NTFS (ii) System Performance Testing
   (iii) Use of Make option
   (c) Explain IP masquerading in detail. How configuration of IP masquerading is done?
   (d) What is the name of main configuration file name for LDAP server? Why LDAP is called light weight?
   (e) What do you mean by multiuser and multitasking operating system?

4. **Attempt any two. All questions carry equal marks.** [10×2]
   (a) Explain different methods of attack in networking.
   (b) Define Network Information System in detail and explain maps and DBM used in NIS.
   (c) Explain the layers architecture of UNIX with the purpose of each layer.

5. **Attempt any two. All questions carry equal marks.** [10×2]
   (a) What are the main features of NFS and NFS Daemons? What is the functionality of export file in NFS?
   (b) What do you mean by system backup and recovery and how it is implemented in LINUX?
   (c) What are the differences between NIS and NIS+? Explain the client side of NIS.

B. TECH.

(UTU) SEVENTH SEMESTER THEORY EXAMINATION 2018-2019

SYSTEM ADMINISTRATION

Time: 03.00 Hours *Max Marks: 100*

Note: Attempt all the questions

1. Attempt any four questions. **[4 × 5 = 20]**

(a) What are the different types of permissions? Explain.

(b) What is the role of IP address and explain the process of assigning IP address.

(c) What is IP filtering, explain with suitable example.

(d) Briefly explain the duties of System Administrator.

(e) Describe about TCP/IP basics in UNIX.

2. Attempt any four questions. **[4 × 5 = 20]**

(a) Define system process. How can you kill a process through command line?

(b) Explain fork system call in brief.

(c) What is the difference between init and inittab?

(d) Explain the steps of checking the ARP tables.

(e) Write notes on the following:

(i) Network file system (ii) Mounting File

3. Attempt any two questions. **[2 × 10 = 20]**

(a) Write the UNIX command for:

i. Deleting a non-empty director recursively

ii. Printing the PID of current shell

iii. Shutting down the system

iv. Compiling the kernel

(b) What is boot disk? Explain about the procedure for configuring network in LINUX.

(c) Name the LDPA client utilities and application of LDPA.

4. Attempt any two questions. **[2 × 10 = 20]**

(a) Write short notes on the following:

(i) Client side of NIS (ii) Methods of Attack

(b) What is the role of proxy server in firewalls?

(c) Explain the file system hierarchy in Linux System.

5. Attempt any two questions. **[2 × 10 = 20]**

(a) Write the steps for installation and configuration process of LDPA.

(b) Explain briefly about working of NFS in Linux with neat block diagram.

(c) Explain the various run levels. Inittab file and switching users/groups detail.

□□□