UNIT 1.

Linux Overview

Structure

1.0 Introduction

The first version of UNIX was created in 1969 by Kenneth Thompson and Dennis Ritchie, system engineers at AT&T's Bell Labs. It went through many revisions and gained in popularity until 1977, when it was first made commercially available by Interactive Systems Corporation.

At the same time a team from the University of California at Berkeley was working to improve UNIX. In 1977 it released the first Berkeley Software Distribution, which became known as **BSD**. Over time this won favour through innovations such as the C shell.

Meanwhile the AT&T version was developing in different ways. The 1978 release of Version 7 included the Bourne Shell for the first time. By 1983 commercial interest was growing and Sun Microsystems produced a UNIX workstation. **System V** appeared, directly descended from the original AT&T UNIX and the prototype of the more widely used variant today.

1.1 Objectives

At the end of this unit, You would be able to

- Undestand the concept of operating system
- Define Memory management
- Know about Processes and Device drivers
- Distinguish about File system
- Explain Arrangement of Kernel Sources
- Exhibit Login,Logoff,Issue commands & switching to another account
- Describe the Pros & Cons of Linux

1.2 Welcome to Linux

Linux is a true 32-bit operating system that runs on a variety of different platforms, including Intel, Sparc, Alpha, and Power-PC (on some of these platforms, such as Alpha, Linux is actually 64-bit). There are other ports available as well, but I do not have any experience with them.

Linux was first developed back in the early 1990s, by a young Finnish then-university student named Linus Torvalds. Linus had a “state-of-the-art” 386 box at home and decided to write an alternative to the 286-based Minix system (a small unix-like implementation primarily used in operating systems classes), to take advantage of the extra instruction set available on the then-new chip, and began to write a small bare-bones kernel.

Eventually he announced his little project in the USENET group, asking for interested parties to take a look and perhaps contribute to the project. The results have been phenomenal!

The interesting thing about Linux is, it is completely free! Linus decided to adopt the GNU Copyleft license of the Free Software Foundation, which means that the code is protected by a copyright -- but protected in that it must always be available to others.

Linux can and should be considered a full-blown implementation of unix. However, it can not be called “Unix”; not because of incompatibilities or lack of functionality, but because the word “Unix” is a registered trademark owned by AT&T, and the use of the word is only allowable by license agreement.

Linux is every bit as supported, as reliable, and as viable as any other operating system solution (well, in my opinion, quite a bit more so!). However, due to its origin, the philosophy behind it, and the lack of a multi-million dollar marketing campaign promoting it, there are lot of myths about it. People have a lot to learn about this wonderful OS!

1.3 What is an Operating System?

Without software a computer is just a pile of electronics that gives off heat. If the hardware is the heart of a computer then the software is its soul. An operating system is a collection of system programs which allow the user to run application software. The operating system abstracts the real hardware of the system and presents the system's users and its applications with a virtual machine. In a very real sense the software provides the character of the system. Most PCs can run one or more operating systems and each one can have a very different look and feel. Linux is made up of a number of functionally separate pieces that, together, comprise the operating system. One obvious part of Linux is the kernel itself; but even that would be useless without libraries or shells.

Starting from hardware equipment, LINUX components include the:

- kernel
- shell
- file system
- commands (or user programs)

In order to start understanding what an operating system is, consider what happens when you type an apparently simple command:

```
$ ls
Mail            c               images          perl
docs            tcl
```

The $ is a prompt put out by a login shell (in this case `bash`). This means that it is waiting for you, the user, to type some command. Typing ls causes the keyboard driver to recognize that characters have been typed. The keyboard driver passes them to the shell which processes that command by looking for an executable image of the same name. It finds that image, in `/bin/ls`. Kernel services are called to pull the ls executable image into virtual memory and start executing it. The ls image makes calls to the file subsystem of the kernel to find out what files are available. The filesystem might make use of cached filesystem information or use the disk device driver to read this information from the disk. It might even cause a network driver to exchange information with a remote machine to find out details of remote files that this system has access to (filesystems can be remotely mounted via the Networked File System or NFS). Whichever way the information is located, ls writes that information out and the video driver displays it on the screen.

All of the above seems rather complicated but it shows that even most simple commands reveal that an operating system is in fact a co-operating set of functions that together give you, the user, a coherent view of the system.

Utilities

The utilities are software tools included with the Linux operating system that let you do work such as text editing, programming, and communications.

1.3.1 Memory management

With infinite resources, for example memory, many of the things that an operating system has to do would be redundant. One of the basic tricks of any operating system is the ability to make a small amount of physical memory behave like rather more memory. This apparently large memory is known as virtual memory. The idea is that the software running in the system is fooled into believing that it is running in a lot of memory. The system divides the memory into easily handled pages and swaps these pages onto a hard disk as the system runs. The software does not notice because of another trick, multi-processing.

1.3.2 Processes

A process could be thought of as a program in action, each process is a separate entity that is running a particular program. If you look at the processes on your Linux system, you will see that there are rather a lot. For example, typing `ps` shows the following processes on my system:

```
$ ps
 PID TTY STAT  TIME COMMAND
 158 tty1 1     0:00 -bash
 159 tty2 3     0:02 emacs intro/introduction.tex
```

If our system had many CPUs then each process could (theoretically at least) run on a different CPU. Unfortunately, there is only one so again the operating system resorts to trickery by running each process in turn for a short period. This period of time is known as a time-slice. This trick is known as multi-processing or scheduling and it fools each process into thinking that it is the only process. Processes are protected from one another so that if one process crashes or malfunctions then it will not affect any others. The operating system achieves this by giving each process a separate address space which only they have access to.

1.3.3 Device drivers

Device drivers make up the major part of the Linux kernel. Like other parts of the operating system, they operate in a highly privileged environment and can cause disaster if they get things wrong. Device drivers control the interaction between the operating system and the hardware device that they are controlling. For example, the filesystem makes use of a general block device interface when writing blocks to an IDE disk. The driver takes care of the details and makes device specific things happen. Device drivers are specific to the controller chip that they are driving which is why, for example, you need the NCR810 SCSI driver if your system has an NCR810 SCSI controller.

1.3.4 The Filesystems

In Linux, as it is for Unix ™, the separate filesystems that the system may use are not accessed by device identifiers (such as a drive number or a drive name) but instead they are combined into a single hierarchical tree structure that represents the filesystem as a single entity. Linux adds each new filesystem into this single filesystem tree as they are mounted onto a mount directory, for example `/mnt/cdrom`. One of the most important features of Linux is its support for many different filesystems. This makes it very flexible and well able to coexist with other operating

systems. The most popular filesystem for Linux is the EXT2 filesystem and this is the filesystem supported by most of the Linux distributions.

A filesystem gives the user a sensible view of files and directories held on the hard disks of the system regardless of the filesystem type or the characteristics of the underlying physical device. Linux transparently supports many different filesystems (for example MS-DOS and EXT2) and presents all of the mounted files and filesystems as one integrated virtual filesystem. So, in general, users and processes do not need to know what sort of filesystem that any file is part of, they just use them.

The block device drivers hide the differences between the physical block device types (for example, IDE and SCSI) and, so far as each filesystem is concerned, the physical devices are just linear collections of blocks of data. The block sizes may vary between devices, for example 512 bytes is common for floppy devices whereas 1024 bytes is common for IDE devices and, again, this is hidden from the users of the system. An EXT2 filesystem looks the same no matter what device holds it.

The File System

The file system is the structure that organizes and stores data on the computer system. Linux organizes files in a way that can be understood as a tree analogy. Simple Linux commands help you navigate and use the file system.

1.4 The Kernel

As its name implies, the kernel is at the core of each LINUX system and is loaded in whenever the system is started up - referred to as a boot of the system.

It manages the entire resources of the system, presenting them to you and every other user as a coherent system. You do not need to know anything about the kernel in order to use a LINUX system. This information is provided for your information only.

The ***functions*** performed by the kernel are:

- managing the machine's memory and allocating it to each process.
- scheduling the work done by the CPU so that the work of each user is carried out as efficiently as is possible.
- organising the transfer of data from one part of the machine to another.
- accepting instructions from the shell and carrying them out.
- enforcing the access permissions that are in force on the file system.

1.4.1 Kernel Data Structures

The operating system must keep a lot of information about the current state of the system. As things happen within the system these data structures must be changed to reflect the current reality. For example, a new process might be created when a user logs onto the system. The kernel must create a data structure representing the new process and link it with the data structures representing all of the other processes in the system.

Mostly these data structures exist in physical memory and are accessible only by the kernel and its subsystems. Data structures contain data and pointers; addresses of other data structures or the addresses of routines. Taken all together, the data structures used by the Linux kernel can look very confusing. Every data structure has a purpose and although some are used by several kernel subsystems, they are more simple than they appear at first sight.

Understanding the Linux kernel hinges on understanding its data structures and the use that the various functions within the Linux kernel makes of them. This book bases its description of the Linux kernel on its data structures. It talks about each kernel subsystem in terms of its algorithms, its methods of getting things done, and their usage of the kernel's data structures.

1.4.1.1 Linked Lists

Linux uses a number of software engineering techniques to link together its data structures. On a lot of occasions it uses *linked* or *chained* data structures. If each data structure describes a single instance or occurance of something, for example a process or a network device, the kernel must be able to find all of the instances. In a linked list a root pointer contains the address of the first data structure, or *element*, in the list and each data structure contains a pointer to the next element in the list. The last element's next pointer would be 0 or NULL to show that it is the end of the list. In a *doubly linked* list each element contains both a pointer to the next element in the list but also a pointer to the previous element in the list. Using doubly linked lists makes it easier to add or remove elements from the middle of list although you do need more memory accesses. This is a typical operating system trade off: memory accesses versus CPU cycles.

1.4.1.2 Hash Tables

Linked lists are handy ways of tying data structures together but navigating linked lists can be inefficient. If you were searching for a particular element, you might easily have to look at the whole list before you find the one that you need. Linux uses another technique, *hashing* to get around this restriction. A *hash table* is an *array* or *vector* of pointers. An array, or vector, is simply a set of things coming one after another in memory. A bookshelf could be said to be an array of books. Arrays are accessed by an *index*, the index is an offset into the array. Taking the bookshelf analogy a little further, you could describe each book by its position on the shelf; you might ask for the 5th book.

A hash table is an array of pointers to data structures and its index is derived from information in those data structures. If you had data structures describing the population of a village then you could use a person's age as an index. To find a particular person's data you could use their age as an index into the population hash table and then follow the pointer to the data structure containing the person's details. Unfortunately many people in the village are likely to have the same age and so the hash table pointer becomes a pointer to a chain or list of data structures each describing people of the same age. However, searching these shorter chains is still faster than searching all of the data structures.

As a hash table speeds up access to commonly used data structures, Linux often uses hash tables to implement *caches*. Caches are handy information that needs to be accessed quickly and are usually a subset of the full set of information available. Data structures are put into a cache and kept there because the kernel often accesses

them. There is a drawback to caches in that they are more complex to use and maintain than simple linked lists or hash tables. If the data structure can be found in the cache (this is known as a *cache hit*, then all well and good. If it cannot then all of the relevant data structures must be searched and, if the data structure exists at all, it must be added into the cache. In adding new data structures into the cache an old cache entry may need discarding. Linux must decide which one to discard, the danger being that the discarded data structure may be the next one that Linux needs.

1.4.1.3 Abstract Interfaces

The Linux kernel often abstracts its interfaces. An interface is a collection of routines and data structures which operate in a particular way. For example all network device drivers have to provide certain routines in which particular data structures are operated on. This way there can be generic layers of code using the services (interfaces) of lower layers of specific code. The network layer is generic and it is supported by device specific code that conforms to a standard interface.

Often these lower layers *register* themselves with the upper layer at boot time. This registration usually involves adding a data structure to a linked list. For example each filesystem built into the kernel registers itself with the kernel at boot time or, if you are using modules, when the filesystem is first used. You can see which filesystems have registered themselves by looking at the file `/proc/filesystems`. The registration data structure often includes pointers to functions. These are the addresses of software functions that perform particular tasks. Again, using filesystem registration as an example, the data structure that each filesystem passes to the Linux kernel as it registers includes the address of a filesystem specfic routine which must be called whenever that filesystem is mounted.

1.4.2 Arrangement of Kernel Sources

At the very top level of the source tree `/usr/src/linux` you will see a number of directories:

arch
: The `arch` subdirectory contains all of the architecture specific kernel code. It has further subdirectories, one per supported architecture, for example `i386` and `alpha`.

include
: The `include` subdirectory contains most of the include files needed to build the kernel code. It too has further subdirectories including one for every architecture supported. The `include/asm` subdirectory is a soft link to the real include directory needed for this architecture, for example `include/asm-i386`. To change architectures you need to edit the kernel makefile and rerun the Linux kernel configuration program.

init
: This directory contains the initialization code for the kernel and it is a very good place to start looking at how the kernel works.

mm
: This directory contains all of the memory management code. The architecture specific memory management code lives down in `arch/*/mm/`, for example `arch/i386/mm/fault.c`.

drivers
: All of the system's device drivers live in this directory. They are further sub-divided into classes of device driver, for example `block`.

ipc
: This directory contains the kernels inter-process communications code.

modules
: This is simply a directory used to hold built modules.

fs
: All of the file system code. This is further sub-divided into directories, one per supported file system, for example `vfat` and `ext2`.

kernel
: The main kernel code. Again, the architecture specific kernel code is in `arch/*/kernel`.

net
: The kernel's networking code.

lib
: This directory contains the kernel's library code. The architecture specific library code can be found in `arch/*/lib/`.

scripts
: This directory contains the scripts (for example awk and tk scripts) that are used when the kernel is configured.

Kernel
: The kernel is the heart of Linux. The kernel is reponsible for resource allocation, security, and low-level interfaces with hardware. As a beginning user, you're not going to touch the kernel.

1.5 The "Bash" shell

A "shell" is a program which interprets commands, either typed in directly by the user, or contained in a file called a "shell script", which is a simple interpreted program. The equivalents in Windows™ would be "command processor" for shell, "COMMAND.COM" or "CMD.EXE" instead of bash, and ".BAT files" instead of shell scripts.

Linux has a variety of different shells, but certainly the most popular is "bash", so it is this one which will be described here (even though many of these instructions apply to all shells). Some of the others are retained simply because there are lots of people who got used to them and don't wish to change, or because they are aimed at a specialised set of users.

Here are some examples of shells:

- Bourne (sh): this is the original Unix shell, available in nearly every flavor of Unix.
- Korn (ksh): a backwards-compatible upgrade to the Bourne shell, developed by David G. Korn at AT&T Bell Laboratories.
- Bash (bash): Bourne-again shell, a public domain shell containing features of the Bourne, Korn and c-shell.
- C-Shell (csh) and T-C-Shell (tcsh): The c-shell is similar to C programming language; t-c-shell extends csh.
- Z-Shell (zsh): an open-source Unix shell.

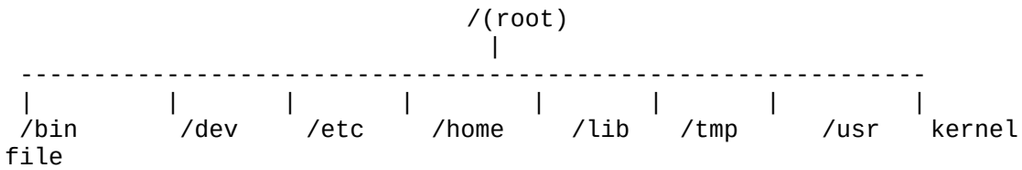
```
bash$ cat /etc/shells  #  to know the available shells in the system
bash$ echo $SHELL  #  to know what shell you are currently using
```

1.5.1 Shell Features and Benefits

- Interactive processing
    - Communicatoin between the user and the LINUX system takes the form of an interactive dialogue with the shell
- Background processing
    - Time-consuming, non-interactive tasks can proceed while the user continues with other interactive processing.
    - The system can perform many different tasks on behalf of a single user
- Input/Output redirection
    - Programs designed to interact with a user at a terminal can easily be instructed to take their input from another source, such as a file, and send their output to another destination, such as a printer or another application
- Pipes
    - Programs that perform simple functions can easily be connected together to perform more complex fuctions, minimizing the need to develop new programs
- Wild-card matching
    - The user can specify a pattern to select one or more files as a group for processing
    - Common file operations can thus be performed on a group of files with a single command
- Shell scripts
    - A commonly used sequence of shell commands can be stored in a file; the name of the file can be later used to execute the stored sequence with a single command
- Programming language constructs
    - The shell inclues features that allow it to be used as a programming language

1.6 LINUX system directories

The standard system directories are shown below. Each one contains specific types of file. The details may vary between, different LINUX systems but these directories should be common to all. Select one for more information on it.



Linux Directories

As you probably already know from working in graphics mode, in Linux the directories (aka "folders") use the slash (/) as a separator (Windows uses backslash (\)). In other words it works just like websites or ftp servers.

Any directory which starts with a slash, such as "/usr/bin", means it is an "absolute" name - the name specifies the entire sequence of directories from the "root" directory (/) up to the specific directory being requested (bin). Thus, it doesn't matter which directory is the "current" directory when you specify that name, it will always point to the /usr/bin directory.

On the other hand a directory which does not start with a slash is relative to the current directory. For example the directory "bin" will point to different directories depending on whether you are in the root directory (in which case it will point to "/bin"), in the "/usr" directory (in which case it will point to /usr/bin) or in the "/usr/local" directory (in which case it will point to /usr/local/bin).

The same applies to files - if you specify "file.txt" it is assumed to be in the current directory, while if you specify "/tmp/file.txt" it will always point to "file.txt" in the temporary directory.

Two special directory names are the current directory, represented by a single period (.) and the parent directory, represented by a double period (..). Thus, if you are in the /home/sandbox directory and type in `ls ..`, it will list the contents of the parent directory, which is /home.

Some System Directories

Below is a list of some common directories that are found in Linux and Linux systems, and what they are used for.

/

This is the root directory, inside which all other directories reside This is similar to the root directory of a drive in Windows (`C:\`), except that in Linux even different hard disks reside within this root.

/bin

This stands for "binary", and contains program (executable) files. This (and other "bin" directories) is where commands such as "ls" can be found.In Windows, the `c:\windows\commands` holds some of command-line programs, but others are scattered in various other directories.

/dev

This stands for "devices". It contains a number of special pseudo-files that are used to access the physical hardware that make up, or are connected to, your computer. For example the parallel-port would be a file called "lp0" in this directory, while the hard disk would be "hda", and its first partition would be "hda0". Windows/DOS uses a similar method, however in Windows these are not in any particular directory. Devices have names like `LPT1`, `COM1` or `CON` - any time you try to access a file with that name from any directory, you will get the parallel printer, serial port or console, respectively.

/etc

This is where (almost) all system-wide configuration information is stored. Almost all configuration information is stored in text files, so you can go into this directory and have a look around with a text viewer if you like. Some of the files are quite cryptic though.

There is no equivalent in Windows, where configuration data can be stored anywhere, including the registry, INI files and other data files in various directories.

/home

This is where users' home directories are usually found. Thus, if you created a user called "sandbox", there will be a directory with the same name in this directory, which will be that user's home directory.

The nearest equivalent in Windows is `c:\windows\profiles`, where some user-specific data is held, together with `c:\My Documents`, where user-created documents go. However other data can be written in many other directories.

/lib

This is where the library files are found. Libraries are files containing reusable functions and routines for programs to use.

There is no equivalent in Windows/DOS.

/mnt

This is where storage devices other than the hard disks are mounted. This directory usually contains subdirectories called "cdrom", "floppy", etc., which - when these devices are mounted - show the contents of the CD-ROM or floppy disk respectively. Your Windows drives may also be automatically mounted in this directory.

There is no equivalent in Windows/DOS.

/opt

This is where optional components of the system are installed. Products such as KDE, Gnome and Oracle may be installed into this directory. The nearest thing in Windows is the `c:\Program Files` directory.

/tmp

This is a temporary directory. All files placed in here will automatically be deleted eventually. The equivalent in Windows/DOS is `c:\windows\temp`.

/usr

Contains a copy of most of the directories in the root. For example, there is a "bin" directory containing programs, a "lib" directory containing libraries, etc. Usually, "core" Linux files are contained in the root direcories, while "non-core" files are in the "/usr" subdirectories. There is no equivalent in Windows/DOS.

/var

Stands for "various". Among the files stored here are the system log files, spool files and other data files. There is no equivalent in Windows/DOS.

1.7 How to Login, Issue commands, Switching to another account and Log Off

The purpose of this section is to quickly show you the way to login, issue a command, and log off a Linux system.

The fundamental things you need to know to use Linux are:

1. logging in
2. issuing commands
3. switching to another account
4. logging off

Getting a Linux Window

In order to log in to your Linux account, you need to get to a computer that can give you access to a Linux shell. You don't necessarily need to be at a Linux computer. You might be at a PC that allows you to open a Linux window, at a terminal that is dedicated to a Linux system, or a workstation that allows you to run Linux.

Check with whomever gave you your Linux account for your username and password. They will probably have written instructions for logging onto your account and a list of locations of computers where you can use Linux.

If you are using a personal computer with a Web browser, the most likely way you'll be able to login to your Linux account is by using telnet.

To telnet to your Linux account, you'll need to know your Linux host computer name. Your Linux host computer might have a name something like alphak.csd.uwm.edu. From your Web browser, use the "File / Open" option (or "Open URL" option) and, enter something this (use your correct Linux host computer name, of course) (use no spaces, use the : and the two /):

```
telnet://cnn.com or telnet:192.168.0.1(IP address)
```

This should cause a Linux window to come up on your personal computer. Now you are ready to login.

Log In

You need to know your username and password. When you know these, protect this information, as anyone who learns your password could log in to your account, change or delete your files, and send email in your name. You don't have to protect your username. In fact, your username will be readily apparent to anyone to whom you send email. It is the combination of your username and password that gives you access to your Linux account.

To login, enter your username where prompted, hit the Return or Enter key, then enter your password and hit the Return key again. Here is a sample session.

```
username: rao
password: *****
```

Linux Commands

The fundamental way that you communicate with Linux is through a command-line interface called the shell. In the shell, you issue commands at the command line. Linux systems vary in how their command lines work. In these lessons, I'll use the dollar sign ($) to stand for the Linux prompt. The Linux prompt is simply a symbol that appears at the start of a command line to let you know that you are in the shell and that Linux is ready and waiting for your command. As you look at the examples in this lesson, don't type the $ itself.

Let's issue a Linux command, the date command to tell us the date and time:

```
$ date
Tue Dec 19 21:16:03 CST 2000
$
```

Linux is case-sensitive. You must type the commands using lowercase letters. The command date is going to work. The commands DATE or Date are not.

Remember to hit the Return or Enter key on your keyboard after you are done typing in the Linux command. If you don't hit the Enter key or the Return key, Linux will just keep waiting until you do.

Switching to another account

To switch to using another account use the command

$ **su *username***

and enter your password when prompted.

You are still in the same working directory you were when you issued the command, but you now have access to the file system for your other account.

To stop using this account enter the command **exit**

Log Off

You need to know how to end your Linux session by logging off. Don't leave your Linux window open on a computer and just walk away. Someone could cause havoc with your files. To log off, just issue the exit command.

```
$ exit
```

1.8 Pros & cons of Linux

Some of the impordant terms:

- **Source Code:** Source code is programming commands. The thousands of lines of programming that tell the computer what to do. Source code must be interpreted by the computer or complied to be useable. *Also known as Source or Code.*
- **Compile:** To convert ASCII source code into binary, which the computer can understand.
- **Kernel:** The base of Linux. The bridge between the software and hardware. This is difficult to explain and I am not sure of a Windows example to compare it to.
- **Swap Space:** Scratch space used by an operating system when it runs out of memory. *Also known as Swap, the Swap File, or Virtual Memory.*

Some advantages of Linux are that it's...

- **Open Source** - To understand what I mean by this, let me explain a little bit about source code. Most commercial operating systems, for example Windows, are *not* open source. The source code is not available to anyone to be viewed or modified. The Linux source code and most of the software for Linux, however, is open source. Anyone can look at it and change or add to it. The fact that anyone can change the source code is good. For example, when a security problem or bug is discovered in Linux it is fixed by others much faster than Microsoft would release a patch. Also, unlike Windows, Linux constantly evolves. It is improved on every day. You can get a new release of the kernel nearly every week.
- **Free in Cost -** Linux being Open Source leads to it being free in cost. There are many distributions of Linux that can be downloaded free of cost from a FTP site. You may also see places where you can purchase Linux on CD. In these cases, you are paying for the cost of creating the CD, documentation, and possibly support.
- **Secure** - Windows 95/98 both have *terrible* security. Windows NT has been proven very bad also. On a Windows system, anyone can sit down and delete all the files on the entire hard drive. Linux uses the Unix idea of permissions. Only authorized users are able to modify, for example, global configuration files. (Settings that affect all users.)
- **No Viruses -** With Linux, you don't have to waste valuable system resources and money for a virus scanner. Since, as described above, Linux is so secure, a virus that attempts to delete a system critical file will receive a permission denied message the same as if a user without the proper privileges trys to delete one.
- **Stable** - How many times has Windows froze up on you? How many times have you been typing a large document or email message when you receive

that dreaded "This program has performed an illegal operation and will be shut down." and be forced to exit the program before getting a chance to save your document. Or perhaps, you use Windows NT for a webserver, for example. How long could it go without having problems? Not very long.

Linux is VERY stable. There have been reports of Linux systems staying up for a *year* and only going down for maintenance. Linux just does not need the workarounds that Windows does. It is a *huge* project always being worked on by many programmers. When a bug that could cause the system to crash is found, it is fixed very quickly by someone.

- **Powerful** - Linux is not a sissies operating system. Although Linux is becoming more user friendly, it is extremely powerful and does not sacrifice features for ease of use.

- **Very Flexible** - For starters, Linux comes in many different distributions. Each have their own differences I feel RedHat is more suited for new users.

- **Multiuser** - Windows 9*x* was a terrible multiuser system. Anyone could access anyone else's files. The password dialog box could be bypassed by clicking Cancel. Linux is a true Multiuser system. Each user has their own directory for their own files and can choose to deny access to other users. Users "own" certain files and can also belong to group(s) which have certain access restrictions taken down. For example, their could be a group called modem which users were permitted to dial out the modem. (With the exception of "root", the administration account which gets unrestricted access to everything!) Sometimes in Windows, users settings conflict. Email would get mixed together, etc. With Linux, Netscape creates a mail directory in each users /home/*user* directory. They get all their email in that one place and can't read each others email.

- **Fast** - Linux is highly optimizable. For example, once you become more advanced with Linux, you will probably want to recomplie the Kernel to optimize it for your CPU or to include support for your sound card. You can not recompile Windows to optimize it for your system, because it is not open source. However, since Linux is open source, you can recompile the Kernel to optimize it for your system.

Some disadvantages of Linux for newbies are that it can be...

- **Difficult to Learn** - Linux is not as easy to use as Windows 9*x*. It will not try and do everything for you. Hopefully, you will find it interesting to tamper with things, etc.

- **Difficult to Set Up & Install** - Both are being worked on and recently the situation has improved greatly.

- **Specific Software may not be available** - For example, Internet Explorer and MS Office are not available for Linux. However, replacements for most specific software is available, such as Netscape Navigator instead of Internet Explorer and WordPerfect instead of Office. There is also a suite available called StarOffice.

Some other notes that newbies should know are...

- **Lots of Docs** - Lots of books and documentation is available for Linux. The Linux attitude is RTFM (Read the (uhh..) friggen manual!) If you can't find

the answer you need there are lots of places where you can ask questions, such as the discussion boards on this site.

- **GNU/Linux** - You may hear it referred to as GNU/Linux. This just means the same thing as Linux alone.

- **Important notes about partitions** - Let me first say, that this is not a fault of Linux. If you had Linux installed and you wanted to install windows, you would still need to mess around with partitions. Partitions are what divide up the hard disks that will allow for organization or for multiple operating systems.

  Before installing Linux, you will need to know you need a Linux partition and a partition for the Linux's swap space. You could erase Windows and make two partitions when installing Linux. This is not recommended for new users. If you have a second hard drive, you could use that for Linux. (If you do, you will be responsible for backing up, moving, or otherwise archiving the files already on this drive.) However, what most people do, is use a program like FIPs or PowerQuest's Partition Magic to create room for Linux.

1.9 Summary

UNIX has a long history as an open development environment. More recently, it has become the system of choice for both commercial and some personal uses. LINUX performs the typical operating system tasks, but also includes a standard set of commands and library interfaces. The building-block approach of LINUX makes it an ideal system for creating new applications.

Linux is quite possibly the most important free software achievement since the original Space War, or, more recently, Emacs. It has developed into an operating system for busi-ness, education, and personal productivity. Linux is no longer only for UNIX wizards who sit for hours in front of a glowing console (although we assure you that many users fall into this category). This book will help you get the most from Linux.

Linux (pronounced with a short *i*,as in *LIH-nucks*) is a UNIX operating system clone which runs on a variety of platforms, especially personal computers with Intel 80386 or better processors. It supports a wide range of software, from T E X, to the X Window System, to the GNU C/C++ compiler, to TCP/IP. It’s a versatile, bona fide implementation of UNIX, freely distributed under the terms of the GNU General Public License.

Linux can turn any 80386 or better personal computer into a workstation that puts the full power of UNIX at your fingertips. Businesses install Linux on entire networks of machines, and use the operating system to manage financial and hospital records, distributed computing environments, and telecommunications. Universities worldwide use Linux to teach courses on operating system programming and design. Computing enthusiasts every-where use Linux at home for programming, productivity, and all-around hacking.

What makes Linux so different is that it is a free implementation of UNIX. It was and still is developed cooperatively by a group of volunteers, primarily on the Internet, who exchange code, report bugs, and fix problems in an open-ended environment.

1.10 Check Your Progress

I. Choose the correct answer:

1. Linux is invented by_____.
   a) Linus Torvalds b) Dennis Ritchie c) James Gosling d)None of these.
2. Operating system is a collection of _____.
   a) S/w programs b) system programs c) Application programs d) all of these
3. Linux components include the _____.
   a) Kernel b) shell c) file system & user programs d) all of these.
4. NFS means _____.
   a) Network File System
   b) Network FAT System
   c) NetBios System
   d) None of the above
5. Device Drivers control the interaction between the _____.
   a) I/p device & operating system
   b) Operating system & o/p device
   c) Operating system & hardware device
   d) All of the above
6. The standard way of mounting file systems onto the _____ directory.
   a) Dev b) mnt c) bin d) all of theseve
7. Linux supports _____ file systems.
   a) Single b) NFS c) multiple d) all of these
8. Examples for DATA STRUCTURES are _____.
   a) Linked List
   b) Doubky Link List
   c) Hash Tables
   d) All of the above.
9. In Linux the directories uses the ____ as a separator.
   a) / b) \ c) // d) all of these
10. binary directory contains _____.
    a) Executable files
    b) Special pseudo-files
    c) Library files
    d) System log files.

II. Say True or False

1. Linux is a 16-bit operating system. True/False
2. Linux is a multi-processor operating system. True/False
3. Linux supports many different file systems. True/False
4. Block size for floppy device is 1024 bytes. True/False
5. KERNEL accepts instructions from the shell and carries them out. True/False

III. Essay Type Questions

1) What is Linux? Write a brief history regarding it?
2) Explain Memory Management?
3) What is meant by Process?
4) What is meant by Scheduling?
5) Explain File System?
6) Write a note on Hash Tables?
7) How the Kernel sources arranged?
8) Explain all the Shell features and it's benefits briefly?
9) Explain the different types of access permissions?
10) Explain all the advantages & disadvantages of Linux?

IV. Further readings and other activities

1. The student shall get more information using man or info or any other help command for
   pwd, mkdir, cd, rmdir, sh, csh, ksh
   Ex: man pwd OR info ls OR --help cd OR apropos cp
2. Get new commands from the list given at the end of any man command information at the place *see also*.
3. Using man intro, get the information about different kinds of commands present in your kind of linux system.
4. For further readings you can refer the following books
    1. Title: *Linux in a Nutshell*
       Author: Jessica Perry Hekman and the staff of O'Reilly & Associates; ISBN 1-56592-167-4; 1997. A complete reference for Linux.

    2. Title: *Harley Hahn's Student Guide to UNIX*, second edition; McGraw-Hill; ISBN 0-07-025492-3; 1996. Not just for students, this is a complete and very readable guide to LINUX and networking.

Reference e-mails: raomvp@yahoo.com roopasindhe@lycos.com
URL / Web Site: http://www.raomvp.bravepages.com

Good-Luck