Kumaraguru College of Technology

Department of Computer Science and Engineering
Coimbatore-641 006

LINUX NETWORK INTERFACE SECURITY SYSTEM

Project work done at P-1066

WINFOCOM SOLUTIONS Pvt. Ltd.
CHENNAI

PROJECT REPORT

Submitted in partial fulfillment of the

Requirements for the award of the degree of

Master of Science in Applied Science

Software Engineering

Bharathiar University, Coimbatore

Submitted by

U.BALAMANIKANDAN
Reg.No-0037S0083

INTERNAL GUIDE

Ms.P. Sudha B.E., MISTE.,
Dept.of Information Technology & Engineering,
Kumaraguru College of Technology,
Coimbatore.

EXTERNAL GUIDE

Mr. S. Karunanithi M.C.A.,
WINFOCOM SOLUTIONS Pvt. Ltd,
Chennai

Department of Computer Science and Engineering

KUMARAGURU COLLEGE OF TECHNOLOGY

Coimbatore – 641 006

CERTIFICATE

PROJECT REPORT 2003

Certified that this is a bonafide report of
the project work done by

U.BALAMANIKANDAN
(Reg. No. 0037S0083)

P. Sudha

Ms.P.Sudha,B.E.,MISTE.,
Project guide
Information Technology & Engineering

S. Thangasamy

Prof. S. Thangasamy , Ph.D.,
Head of the Department
Computer Science & Engineering

Place: Coimbatore
Date: 25.9.03

Submitted for viva-voce examination held on

29.09.03

S. Thangasamy 29/9

Internal Examiner

External Examiner

Winfocom Solutions Pvt. Ltd.

Nelson Towers, I Wing, 3rd Floor.
51, Nelson Manickam Road,
Chennai - 600 029. India.
Phone : 3743228 / 3745808

BONAFIDE CERTIFICATE

This is to certify that **U. Balamanikandan (2KSE05)** student of **Kumaraguru College of Technology** doing his seventh semester of M.Sc (Software Engineering) has successfully completed his project entitled **"Linux Network Interface Security System"** within the given duration from June 2003 to September 2003.

During his project duration his conduct and contribution has been excellent.

We wish him all the best for his future endeavors.

For ***Winfocom Solutions Pvt. Ltd.,***

V. Gopi Shankar
(Project Manager)

S. Karunanithi
(Project Guide)

WINFOCOM SOLUTIONS (P) LTD.,
I Wing, III Floor, Nelson Tower,
51, NELSON MANICKAM ROAD,
AMINJIKARAI, CHENNAI - 600 029.

DECLARATION

I here by declare that the project work entitled

" LINUX NETWORK INTERFACE SECURITY SYSTEM "

submitted to Kumaraguru College of Technology, Coimbatore affiliated to Bharathiar University as the project work of **Master of Science in Applied Science Software Engineering Degree**, is a record of original work done by me under the supervision and guidance of **Mr. S. Karunanithi, M.C.A**, WINFOCOM SOLUTIONS Pvt. Ltd, Chennai and **Ms. P. Sudha B.E., MISTE.**, IT Department, Kumaraguru College of Technology, Coimbatore and the project work has not found the basis for the award of any Degree. Diplamo/ Associateship/ Fellowship or similar title to any candidate of any university.

Place : Coimbatore

Date : 25.9.03

U. Bala

(U.BALAMANIKANDAN)

Reg. No: 0037S0083

Countersigned by

P. Sudha.

(Internal Guide)

Ms. P. Sudha B.E., MISTE.,

Kumaraguru College of Technology,

Coimbatore.

ACKNOWLEDGEMENT

I would like to express my sincere thanks to **Dr K.K. PADMANABHAN, B.Sc. (Engg), M.Tech., Ph.D.,** *Principal, Kumaraguru College of Technology, Coimbatore* and **Mr. V. GOPI SHANKAR** , Project Manager., **WINFOCOM SOLUTIONS** for giving me this opportunity to work on this project .

My sincere thanks to **Dr. S. THANGASWAMY, Ph.D.**, Professor, *Head of the Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore* for his support during my project. I also express my sincere and heart felt thanks to **Mrs S.Devaki BE., M.S** , Assistant Professor ,for her unfailing enthusiasm and guidance that paved me to the completion of this project.**Ms P. SUDHA, B.E., MISTE.,** *Lecturer in Department of Information & Technology, Kumaraguru College of Technology , Coimbatore* and my guide for her invaluable guidance and support throughout the project.

I sincerely thank **Mr. K. SARAVANA PRAKASH**, *Director*, **WINFOCOM SOLUTIONS** and my project guide **Mr. S. KARUNANITHI**, for sharing his precious time , for his consistent encouragement and support throughout the project.

I also express my gratitude towards my fellow trainees in **WINFOCOM SOLUTIONS** for their constant support and encouragement throughout the project . Last but never the least, my sincere thanks to all my family members and friends who helped and encouraged me during the entire course of this project work.

SYNOPSIS

Many organisations have a substantial number of computers often located far apart. Initially, each of these computers may have worked in isolation, but at some point the management may have decided to connect them to extract and correlate information about the entire organisation. Here the concept of resource sharing comes into play. More often when data is transferred across the Network there is possibility of illegal access of the data by intruders, when enough security is not provided.

The objective of our project is to provide extra-security features that are not available on LINUX. Our project will also provide necessary and easy interface to configure and monitor them staying locally and remotely. This software should also make administrator's work easier and more efficient. This software is highly flexible and efficient without the direct configuration of LINUX system .

This software should also incorporate features of configuring the network security settings staying in the network without interacting directly with the LINUX. This system should also provide network security and management through administrator using the basic concept called "Sniffing".

The clients on network should be disconnected whenever they are considered as out of time by changing the key value, which is distributed to the later connections, made by the clients.

The connected clients need to be monitored at every moment of time. Their actions on the server and across the network are monitored in order to avoid illegal actions.

CONTENTS

1. INTRODUCTION

1.1 CURRENT STATUS OF THE PROBLEM TAKEN UP

1.1.1 Existing System

The existing system concentrates only on connection to the remote server. This is found to be inefficient in many occasions where security during connection and transmission is required. The local client often needs access both local machines data as well as remote machines. This liabilities can be overcome by our project LNISS. The transaction of the files between systems requires the presence of a third party such as a web server to enable the FTP and for client to access. Using the web server we enable the FTP machine. The downloaded file is in a text format.

Disadvantages of the Existing System

- Log files are not maintained.
- Cannot hide the server for some of the clients.
- Cannot trace information about the current user.
- The transmission of files done through FTP is in a plain text format.

1.1.2 Proposed System

The **LNISS (LINUX Network Interface Security System)** provides additional security features, easy interface, configure, monitoring the transaction over network for safeguard of the data. The aim is to create a customized file transaction server to transfer files between domains. The system does not require a third party to transfer the file.

Advantages of the Proposed System

- The proposed system provides a secured channel for the transmission of the files.
- The file uploaded or downloaded is a cipher text and can be decrypted only with the suitable algorithm.
- It avoids access to unauthorized users.
- It denies permission to unauthorized access or modification.
- It maintains a trace of user access in a log file.
- It hides the server from the unauthenticated clients.

2. COMPANY PROFILE

WINFOCOM SOLUTIONS (P) LTD

- Merging technology with reality

[CORPORATE PROFILE]

NELSON TOWERS, I WING, III FLOOR
#51, NELSON MANICKAM ROAD
CHENNAI, INDIA -600029.
Phone 91- 44 - 3745808, 3743228.

WINFOCOM'S STRATEGIC PARTNERS

As the developments in wireless technologies assume momentum and bring in the mobile Internet era with unparalleled possibilities, Winfocom Solutions has tied up with Infocomm Solutions ltd, to offer solutions in the area of wireless technologies.

Infocomm Solutions, a total wireless solutions company has a strategic tie-up with AUSystem, Sweden, one of Europe's largest wireless IT consultancy group and partly owned by Ericcson to provide solutions in the wireless technologies of WAP, Bluetooth, GPRS, 3G, 4G, M-commerce and M-security.

General Packet Radio Service (GPRS) is a new technology to provide high capacity end to end IP Packet Services and Wireless Internet Applications over the GSM network. The next six months will see the wide spread adoption of GPRS technology into GSM networks around the world. For the end user, GPRS means always connected, always online and to access the internet as you are used to on your desktop over the mobile network at theoretical maximum speeds of 171.2 kbps.

Bluetooth is the codename for a technology specification for small form factor, low cost, and short-range radio links between Mobile PC's, Mobile Phones and other portable devices. It is estimated that, before year 2002, Bluetooth will be a built-in feature in more than 100 million mobile phones in several million other communication devices, ranging from headsets and portable PC's to desktop computers and notebooks.

THE FUTURE

Since today's networked economy is built on the information technology platform, real time and online processing dictate the speed of competition. Technology and Process are keys to organisational success.

Winfocom is focused on providing solutions in the areas of Web and Wireless Technologies and has a growing and dedicated team of software professionals to meet the challenging IT needs of any Industry. We have strived to perfect rigorous internal quality standards to deliver high quality products at the lowest possible time and thus creating maximum customer value.

Winfocom has a growing list of satisfied customers and is looking to add new customers across the globe.

OUR MISSION

"TO INNOVATE AND DEVELOP CUTTING EDGE SOFTWARE SOLUTIONS, AND TO PROVIDE WORLD CLASS SERVICES IN THE KNOWLEDGE ECONOMY ".

OUR THOUGHTS

We believe that Internet by facilitating free flow of intellectual capital is the ultimate driving force of the global economy. We can offer you the best

solution based on an approach that we have refined to create maximum customer value.

We are dedicated to total customer satisfaction and employ cutting edge technology to provide the highest quality solutions. We have instituted Quality Systems with a goal of achieving SEI Level 4 Certification in the shortest possible time.

ABOUT WINFOCOM

- Winfocom was incepted in 1998. Since then we have grown rapidly and forged relationship with companies in UK Singapore and US. Winfocom has strength of 18 dedicated software consultants with expertise and experience in multiple platforms.

- Winfocom's competency lies in providing innovative and high quality software solutions for Object oriented software product development, Wireless Solutions, Internet/Intranet applications, Web Business Solutions and Enterprise Solutions.

- Our team consists of high calibre software professionals with extensive skill sets and experience in different technologies.

- Winfocom has signed up with M/S INFOCOMM SOLUTIONS LTD, India to provide wireless solutions in the area of WAP, BLUETOOTH, GPRS, 3G etc. Infocomm solutions is a subsidiary of M/s Future Techno designs Pte ltd, Singapore and has a strategic tie up with M/s AUSYSTEMS of Sweden (Partly Owned By ERICSSON).

WINFOCOM'S MANAGEMENT TEAM

K.SARAVANA PRAKASH,

MANAGING DIRECTOR

Saravana Prakash has 9 years of management experience in the IT industry. He is a graduate in Mechanical Engineering and leads the Winfocom team. He is in charge of Marketing and Business Development.

CLARENCE ANTONY FERNANDO,

SENIOR SOFTWARE ARCHITECT

Clarence has 8 years of experience in software development in various technologies and has handled several key projects as a project manager. He leads the software team at Winfocom and has multi platform technical skills. Clarence has a Master's degree in Computer Applications.

M.VINU

DIRECTOR - TECHNICAL

Vinu has an Engineering degree in Electronics and Communication Engineering and has gone through all stages of software development cycle. She is in charge of wireless applications and is currently working on developing solutions in the wireless areas of WAP and GPRS.

WINFOCOM'S CLIENTELE

Clients to name a few,

- Future Techno Designs Pte Ltd - Singapore
- Gangakaveri Entertainment Plus Ltd - Chennai - India
- Infermax Limited - UK
- W3tech Limited - UK
- Shriram Group of Companies - India.
- PlacenSpace.com-India
- Kaivalya Technologies Pvt Ltd-Pondicherry

3. SOFTWARE REQUIREMENTS

3.1 HARDWARE SPECIFICATION:

PROCESSOR	: Pentium II (300 MHz) and Above
RAM	: 64 MB
FLOPPY DRIVE	: 3.5 inch
CD-ROM DRIVE	: 36x
DISPLAY DEVICE	: Monitor
KEYBOARD	: 105 Keys
HARD DISK	: 3 G B

3.2 SOFTWARE SPECIFICATION :

Programming Language	: C
Utilities	: TCP/IP Protocol, Socket programming, IPC, Signals
Platform	: LINUX 8.0

3.3 DESCRIPTION OF THE SOFTWARE PACKAGES USED

3.3.1 C — The Programming Language

C is what the world is breathing today. And books on C are filling everybody's book shelves, like believers converging on a holy bank. If you take a leisurely walk down the choked arteries that line computer paths, there's one horn you will hear blaring incessantly and that is C. Loud and sonorous, the booms have changed in to echoes that have transcended space and time boundaries.

In this absolutely crammed world, men who know C are the men who matter. For C is not merely a programming tool or a super efficient road roller that smoothens all the bumps that ever arose. Rather, it is philosophy and a way of life. Like a benevolent parent, it allows a long rope within a loosely woven web of rules and principles, encouraging bouts of exploration on one's own. C opens the coffers that lie under power-coated steel plates, hammered together to make the computer, lays it bare and gives you the freedom to meander, to carry the wealth in the direction you choose.

Characteristics of C

The following characteristics of C have lead to its popularity as a programming language.

1. Small size
2. Extensive use of function calls
3. Structured language
4. Low level (Bit wise) programming readily available
5. Pointer implementation

C has now become a widely used professional language for various reasons.

1. Generality and Robustness
2. High-level constructs.
3. Handle low-level activities.
4. Efficiency
5. Portability
6. Expandability

Uses of C

C was initially used for system development work, in particular the programs that make-up the operating system. Mainly because it produces code that runs nearly as fast as code written in assembly language. Some examples of the use of C might be:

- Operating Systems
- Language Compilers
- Assemblers
- Text Editors
- Network Drivers
- Language Interpreters
- Utilities

String Functions

The following header file is included for the string functions

#include <string.h>

➤ strcmp()

- compares one string with another

Syntax : int strcmp(string1, string2) ;

➤ strcat()

- appends one string to another

Syntax : char * strcat(string1, string2) ;

➤ strcpy()

- copies one string into another

Syntax : char * strcpy(string1, string2) ;

➤ strstr()

- scans a string for the occurrence of a given substring

Syntax : char * strstr(string1, string2) ;

➤ strtok()

- searches one string for tokens, that are separated by delimiters defined in the second string

Syntax : char * strtok(string1, string2) ;

The first call includes the original string as the first argument and the subsequent calls instead take NULL value.

3.3.2 LINUX- the multi-tasking operating system:

LINUX is the much maligned, debated and devoured entity today. An operating system that was possible only through C, has always been dealt with rather cursorily. LINUX is multi-tasking and multi-user operating system. But to think of it as a mere OS would be a gross under-statement. It is much more, a complete philosophy signifying that small and simple is beautiful. The entire OS is geared around this idea, giving us the ability to write small programs that do one task and connect them to others. The result is a synergy, a workbench of tools that are more than sum of their parts.

But over and above this, it is also an evolutionary process, moving constantly towards infinity. Reflecting not just an attitude but a path to the future. A future that is not stagnant but continuously growing. And as if to underline this sense of a process the word *process* it has been used to signify a program that runs under **LINUX.**

Advantages of LINUX

Why would you choose Linux over UNIX? Linux is free. Like UNIX, it is very powerful and is a real operating system. Also, it is fairly small compared to other UNIX operating systems, although to be honest, some versions of BSD UNIX, such as Open BSD, can be shoehorned onto a 60MB file system. Many commercial UNIX operating systems require 500MB or more, whereas some versions of Linux, such as the embedded uCLinux, can be run on as little as 2MB of file space and 2MB of RAM. You can even run Linux from a floppy disk!

Realistically, you will want room for development tools, data, and so on, which can take up 500MB or more, and your RAM should be 32–64MB (although the more, the merrier!).

- **Full multitasking**—Multiple tasks can be run in the background, and multiple devices, such as a modem, printer, and hard drive, can be accessed at the same time.
- **Virtual memory**—Linux safely uses a portion of your hard drive as virtual memory, which increases the efficiency of your system by keeping active processes in RAM and placing less frequently used or inactive portions of memory on disk. Virtual memory also utilizes all your system's memory and doesn't allow memory segmentation to occur.
- **Hardware support**—Linux, especially Intel-based versions, supports nearly all hardware architectures and devices, with the best support for legacy hardware. This is an advantage in that new versions of the operating system will not make your older hardware obsolete.
- **The X Window System**—The X Window System is a graphics system for UNIX machines. This powerful interface supports many applications and is the standard interface for the industry.

- **Built-in networking support**—Linux uses standard TCP/IP protocols, including Network File System (NFS), Network Information Service (NIS,

formerly known as YP), Session Message Block (SMB), and others. You can access the Internet by connecting your system with an Ethernet card, or a parallel-port, serial cable, or over a modem to another system.

- **Shared libraries**—because each command shares a common library of subroutines it can call at runtime, Linux helps saves memory and hard drive space.
- **Compatibility with the IEEE POSIX.1 standard**—Because of this compatibility, Linux supports many of the standards set forth for all UNIX systems.
- **Open Source code**—The Linux kernel uses no code from AT&T or any other proprietary source. This allows other organizations, the GNU project, hackers, and programmers from all over the world to develop and contribute software for Linux.
- **Documentation**—nearly every Linux distribution comes with more than 12,000 pages of documentation in the form of manual pages, info documents, or guides. You'll also find extra technical documentation for software packages under the /usr/share/doc directory. Unlike operating systems offered by the monopolistic software industry, Linux is fully documented—one problem might be that there is too much information!
- **Lower cost than most other UNIX systems and UNIX clones**—if you have a fast Internet connection and a CDR drive, you can freely download Linux off the Internet. Many books also come with a free copy (this book includes the latest version of Intel-based Red Hat Linux on the CD-ROMs).

GNU software support—Linux can run a wide range of free software available through the GNU project. This software includes everything from programming tools, such as compilers, assemblers, linkers, and loaders, to system administration utilities, such as stream editors, the venerable emacs editor, and even games.

C AND LINUX

Before proceeding further we need to explore one very basic question. Is C under DOS the same as C under LINUX?

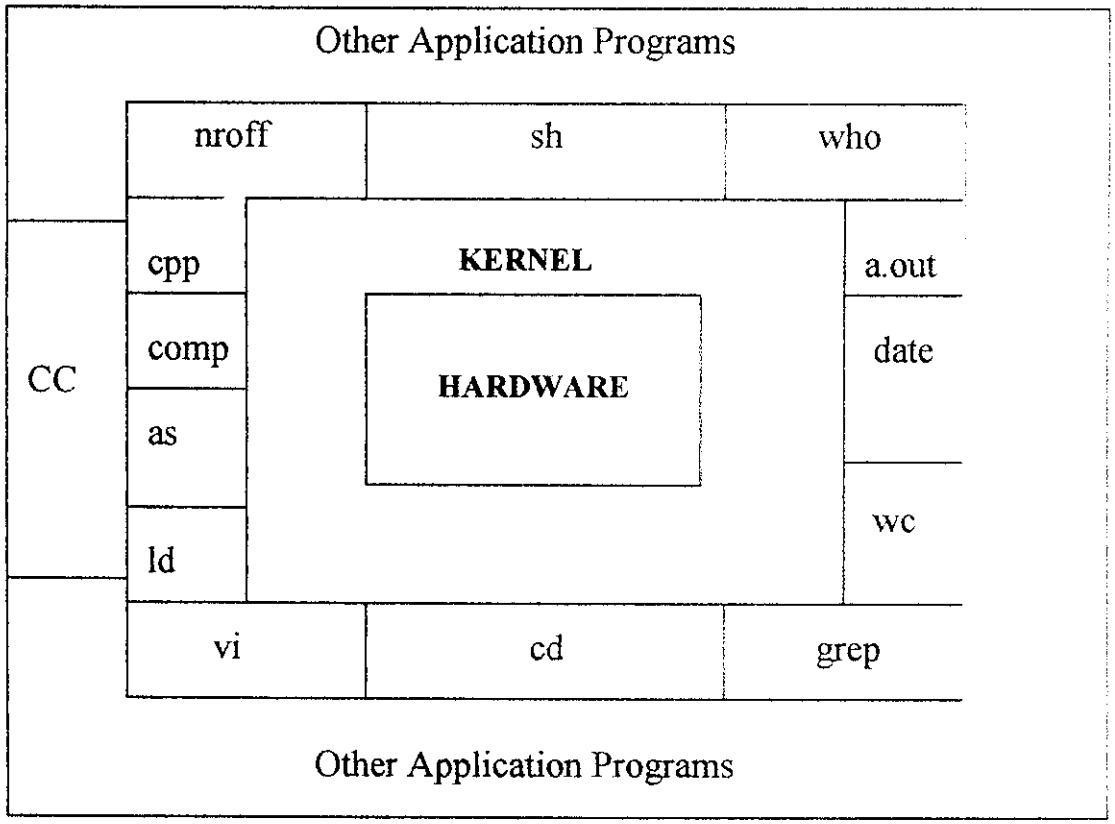
There has been lot of debate over the years about this topic. In the end it was decided that C remain an entity, no matter what the OS under which it was run. Why? the basic structure of C, its power, its data types, its control structures, its syntax all remain the same no matter which OS it run under.

But there must be a difference! The difference is that the DOS shell, COMMAND.COM, takes the commands or data the way it is present. The LINUX shell, however, is intelligent. It sometimes proceeds to deduce. So C runs the same under both OS, but their interpretation is different. So if there is no difference C under DOS and C under LINUX, why this project has been done using C under LINUX?

There are many reasons why LINUX is chosen as the platform for this project. C and LINUX that's the way the world seems to be moving .You know these two and your mobility are assured. These two are inseparable as newly wedded couples .LINUX is a multi –tasking, multi –user OS unlike DOS. And that is what we need to explore C through .It's ability to be used under LINUX for multi –tasking and in a multi –user environment.

One of the most important factors in masking the world village is the level of sophistication technology has reached. Communication in LINUX plays a very important role. We can send files to be printed at some distant location. More than just sharing data actual communication can also take place in the form of a computer based conference.

Architecture of the LINUX system



3.3.3 Concept of Networking:

A network is a communication system that allows users to access resources on other computers and exchange messages with other users. It allows users to share resources on their own systems with other network users and to access information on centrally located systems or systems that are located at remote offices.

A network is a data communication system that links two or more computers and peripheral devices. Users interact with network-enabled software applications to make a network request (such as to get a file or print on a network printer). The application communicates with the network

software and the network software interacts with the network hardware. The network hardware is responsible for transmitting information to other devices attached to the network.

Client – server model:

The standard model for Network application is client server model. A Server is a process waiting to be contacted by a client process so that the server can respond to that client request. The server process can classified into two types

1. Iterative Servers
2. Concurrent Servers

Iterative server:

When a client request can be handled by the server in a known, short amount time, the server process handles the request itself. These servers are called Iterative Servers.

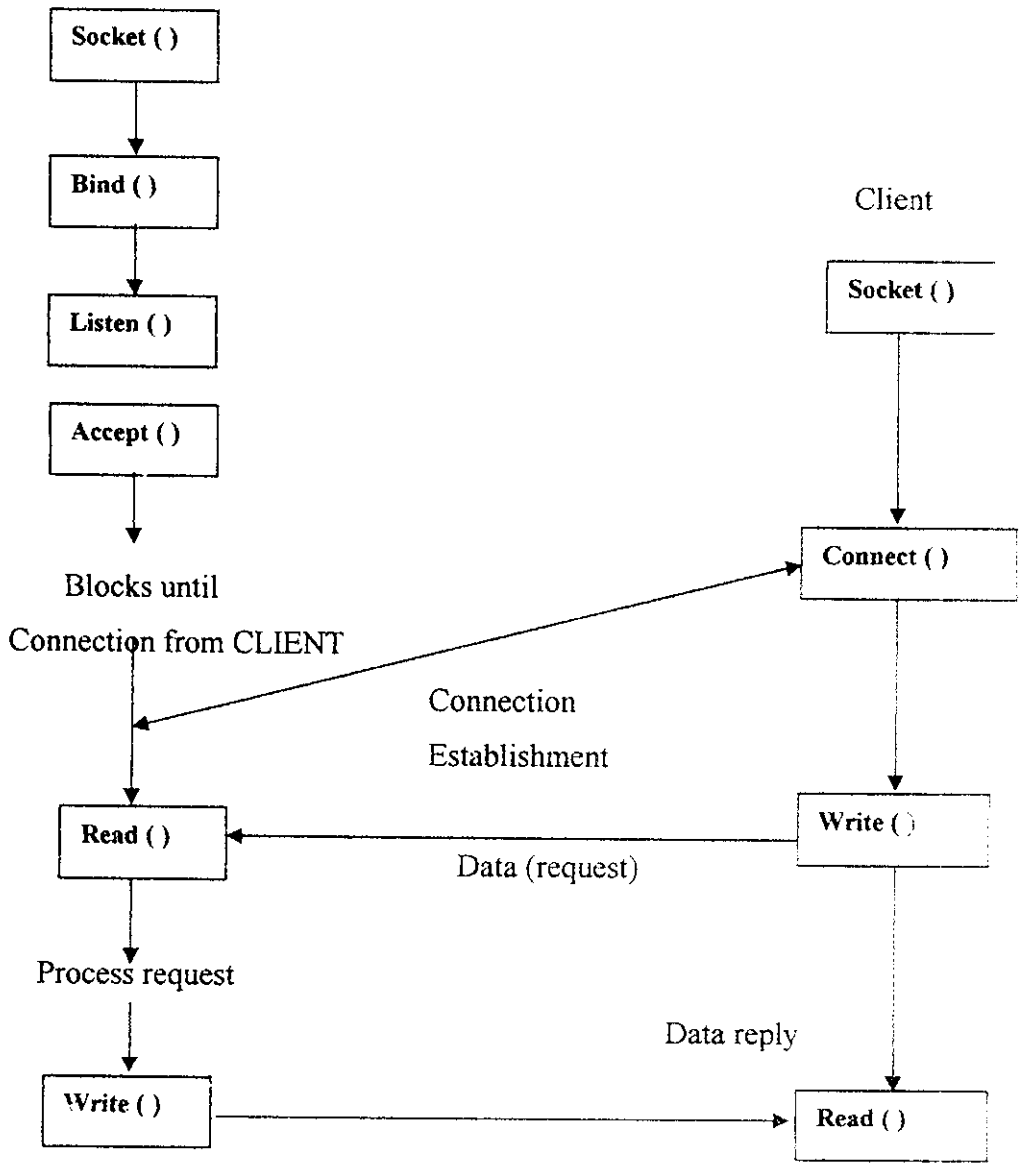
Concurrent servers:

A Concurrent Server invokes another process to handle each client's request, so that the original server process can go back to sleep, waiting for the next client request.

3.3.4 FUNCTIONAL DESCRIPTION

Server

(Connection oriented protocol)



ELEMENTARY SOCKET SYSTEM CALLS

SOCKET :

Prototype:

int socket(int *domain*, int *type*, int *protocol*)

Arguments:

domain :

The *domain* parameter specifies a communications domain within which communication will take place

PF_INET

IPv4 Internet protocols

PF_INET6

IPv6 Internet protocols

PF_IPX IPX - Novell protocols

PF_PACKET

Low level packet interface

type :

Specifies the semantics of communication.

SOCK_STREAM

Provides sequenced, reliable, two-way connection-based byte streams. An out-of-band data transmis-Supports datagrams (connectionless, unreliable mes-sages of a fixed maximum length).

SOCK_SEQPACKET

Provides a sequenced, reliable, two-way connection-based data transmission path for datagrams of fixed maximum length; a consumer is required to read an entire packet with each read system call.

SOCK_RAW

Provides raw network protocol access.

SOCK_RDM

Provides a reliable datagram layer that does not guarantee ordering.

SOCK_PACKET

Obsolete and should not be used in new programs

Return Value:

-1 is returned if an error occurs; otherwise the return value is a descriptor referencing the socket.

Description:

Socket creates an endpoint for communication and returns a descriptor. The descriptor is used for further communication on the socket.

BIND

The bind system call assigns an address to an unnamed socket.

Function:

int bind(int *sockfd*, struct sockaddr **my_addr*, int *addrlen*);

Arguments:

Sockfd : socket identifier.

my_addr: reference to local address of *addrlen* bytes.

Return Value:

On success, zero is returned. On error, -1 is returned

Description:

Before a SOCK_STREAM socket is put into the LISTEN state to receive connections, you usually need to first assign a local address using bind to make the socket visible.

LISTEN

The listen system call is used by a connection-oriented server to indicate it is willing to receive connections.

Function:

int listen(int *s*, int *backlog*)

Arguments:

s : socket identifier for servers.

backlog: defines the maximum length the queue of pending connections may grow to.

Return Value:

On success, zero is returned. On error, -1 is returned

Description:

Listen specifies the willingness to accept incoming connections and a queue limit for incoming connections are specified with listenThe listen call applies only to sockets of type SOCK_STREAM or SOCK_SEQPACKET.

ACCEPT

After the connection-oriented server executes a listen, it waits for connection requests from client(s) in the accept system call

Function:

int accept(int *s*, struct sockaddr **addr*, int **addrlen*)

Arguments:

s: Socket identifier in listening mode.

addr: Address of the client.

addrlen: length of the addr field.

Return Value:

On success, zero is returned. On error, -1 is returned

Description:

The accept function extracts the first connection request on the queue of pending connections, creates a new socket with the same properties of *s*, and allocates a new file descriptor for the socket.

CONNECT

A client process connects a socket descriptor after a socket system call to establish a connection with the server.

For a connection-oriented client, the connect (along with an accept at the server side) assigns all four addresses and process components of the association.

Function:

int connect(int *sockfd*, struct sockaddr **serv_addr*, int *addrlen*)

Arguments:

sockfd : socket identifier.

serv_add: address of the server to be contacted

addrlen: length of the serv_add

Return Value:

On success, zero is returned. On error, -1 is returned

Description:

If the socket is of type SOCK_DGRAM, this call specifies the peer with which the socket is to be associated; this address is that to which datagrams are to be sent, and the only address from which datagrams are to be received. If the socket is of type SOCK_STREAM, this call attempts to make a connection to another socket. The other socket is specified by serv_*addr*, which is an address in the communications space of the socket. Each communications space interprets the *serv_addr* parameter in its own way. Generally, stream sockets may successfully connect only once; datagram sockets may use connect multiple times to change their association. Datagram sockets may dissolve the association by connecting to an address with the *sa_family* sockaddr member set to AF_UNSPEC.

SEND

Function:

int send(int *s*, const void **msg*, int *len*, unsigned int *flags*);

int sendto(int *s*, const void **msg*, int *len*, unsigned int *flags*, const struct sockaddr **to*, int *tolen*);

Arguments:

s: socket identifer

msg: message to be transmitted

len: length of the message

flags: any control indicators to the lower layers.

to: destination address of in case of connectionless sockets.

tolen: length of the destination address field.

Return Value:

The calls return the number of characters sent, or -1 if an error occurred.

Description:

Send may be used only when the socket is in a *connected* state, while sendto and sendmsg may be used at any time. The address of the target is given by *to* with *tolen* specifying its size. The length of the message is given by *len*. If the message is too long to pass atomically through the underlying protocol, the error EMSGSIZE is returned, and the message is not transmitted.

RECV

Function:

int recvfrom(int *s*, void **buf*, int *len*, unsigned int *flags*
struct sockaddr **from*, int **fromlen*);

Arguments:

s: Socket identifier
buf: buffer to receive the data.
len: length of the buffer.
flags: flags if any.
from: source of the message.
fromlen: length of the from field.

Return Values:

These calls return the number of bytes received, or -1 if an error occurred.

CLOSE

Function:

close (int socketid)

Argument:

socketid: socket identifier.

Return Values

0 is returned on success else –1 is returned.

SHUT DOWN

shutdown(int socket, int how)

If the how field is 0, this will disallow further reading (recv) from the socket. If the how field is 1, subsequent writes (send) will be disallowed. The socket will still need to be passed to close.

4. PROPOSED APPROACH TO THE PRODUCT

System should provide an extra security feature that is not otherwise available on Linux system and provide necessary and easy interface to configure and monitor them staying locally and remotely so as to make administrator's work easier and more efficient. So software should be providing e security on these systems with high flexibility and efficiency without the direct configuration of Linux system.

This project should enhance the network security of a Linux system by checking the network request made on the system by different users and controlling them according to the configurations made on the software.

The software should also incorporate features of configuring the network security settings staying in the network without interacting directly with the Linux. System is aimed at enriching a Linux system to fulfill an administrator's need to track all the aspects of network security and management using the basic Concept "Packet Sniffing". This aimed at giving features like;

1. Login requests to the Linux System

This feature will help to track the successful and unsuccessful login request from a specific host thus enabling the software to decide whether to disable specific host for a specified period of time

2. Activities that is being performed on Linux System by the logged in Users

Log files generated by this feature can be used to track a users day to day activities on a regular basis.

3. Hiding specific data from specific users

This feature will be helpful if there is more than one user with the same privileges and still some special restrictions have to be implied on certain users.

4. Denial of Login requests by Users

These Features should provide administrators to enable or disable specific Linux users in logging on to the system. This feature should also provide feature of specifying the hosts capable of making a logging-in request to a Linux system

5. DETAILS OF THE DESIGN

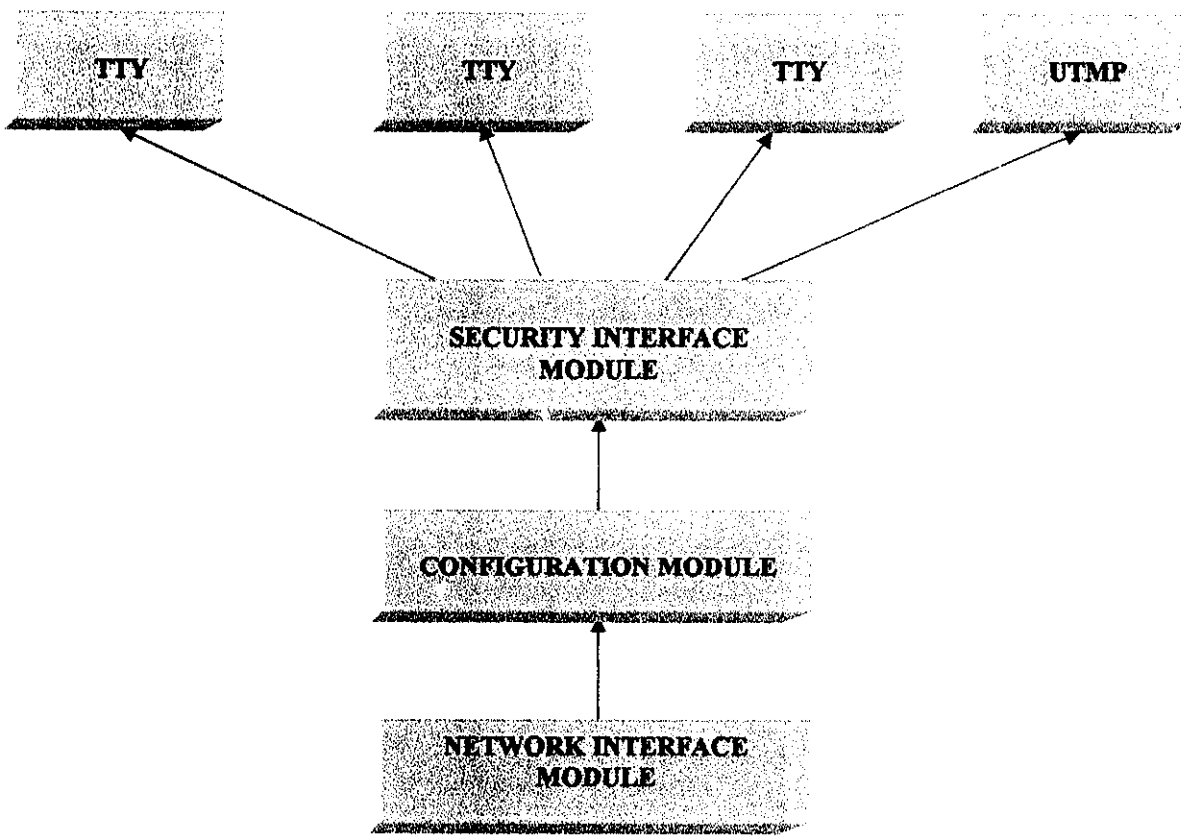
5.1 MODULE DESCRIPTION

The LNISS **(LINUX Network Interface Security System)** provides additional security features, easy interface, configure, monitoring the transaction over network for safeguard of the data.

This system can be modulated mainly as follows

- Security Interface.
- Configuration Module
- Network Interface Module

A Model Design:



Security Interface:

This Module as such will be running as a background process on LINUX system, like a sniffer, which will directly interact with it to provide necessary security as it is required. This program will listen to the entire request made to the LINUX system by the users to decide upon the actions to be performed on each request.

- Mandatory Requirements.
  - It should be capable of understanding the user who has initiated the request
  - It should be capable of understanding the host from which the request has be evolved
  - It should be capable of communicating with the user who has made the request (Necessary intimation)

This module will be always checking the ttys that is available on the system and checking utmp so that it can know about a request that has been made on the system and generate necessary steps to avoid illegal access to system.

Configuration Module:

This module will be responsible for performing all configurations on the SECURITY INTERFACE by communicating with it. This should be initiated by the administrator as and when a requirement of reconfiguring the security features of the System arises.

- Mandatory Requirements.
  - Should Validate the Administrator before letting him to configure the software
  - Should be having provision to add security setting for users-wise and host-wise
  - Should be having provision to view log files generated by the Security interface.

Network Interface:

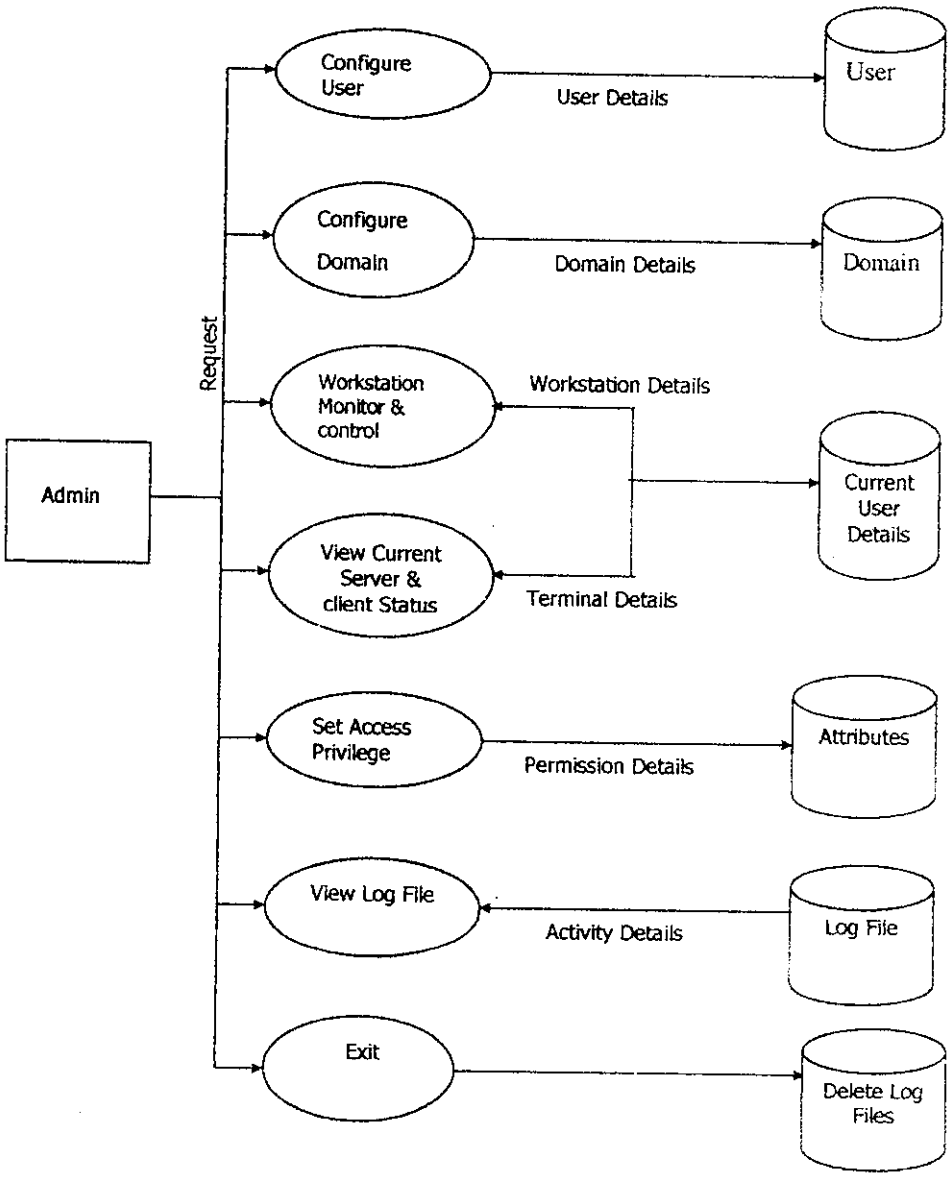
This module should have all features of Configuration module with capabilities of performing configuration of the software by staying on a network. This module should ensure that configuration happens from an LINUX network or a non-LINUX network. Therefore, it will have additional overhead of ensuring the configuration is being made from a specific network and by authorized users.

- Mandatory Requirements.
  - Should Validate the Administrator before letting him to configure the software
  - Should have feature of identifying the network from which the request has been made.
  - Should be having provision to add security setting for users-wise .
  - Should Generate log-files for the unsuccessful and successful attempt to access this interface

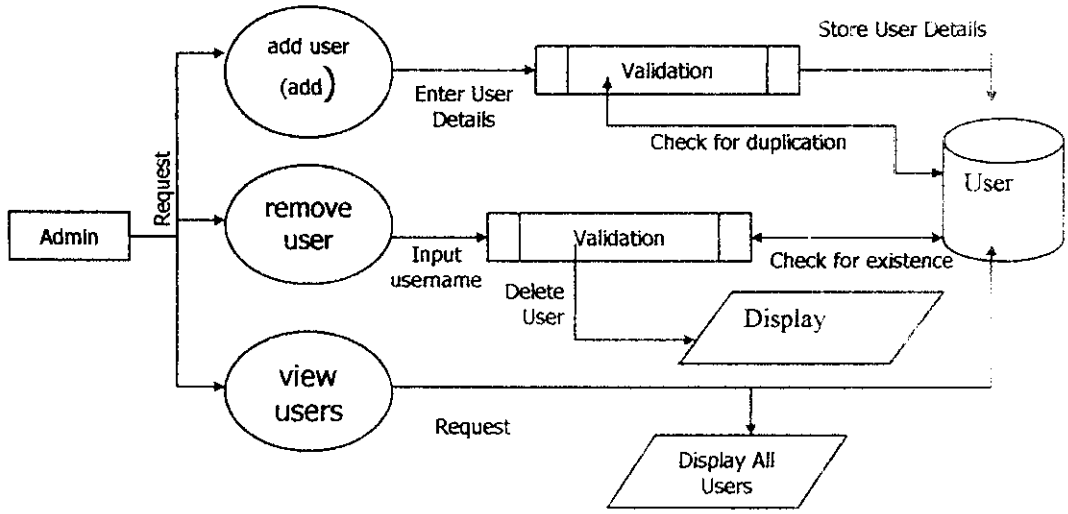
5. 2 ELEMENTS OF DESIGN

5.2.1 Data Flow Diagram

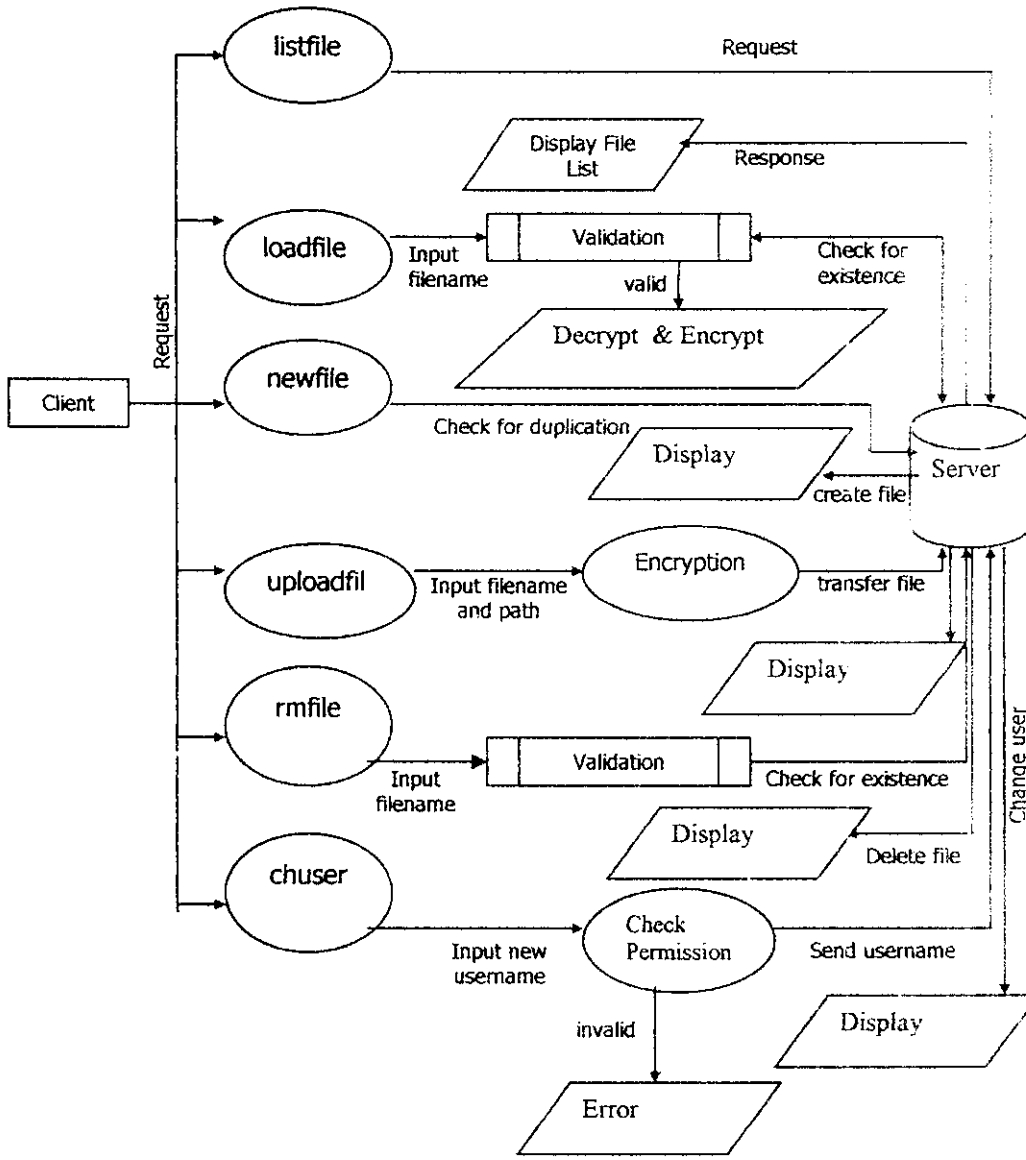
Administrator Process (DFD Level 1)



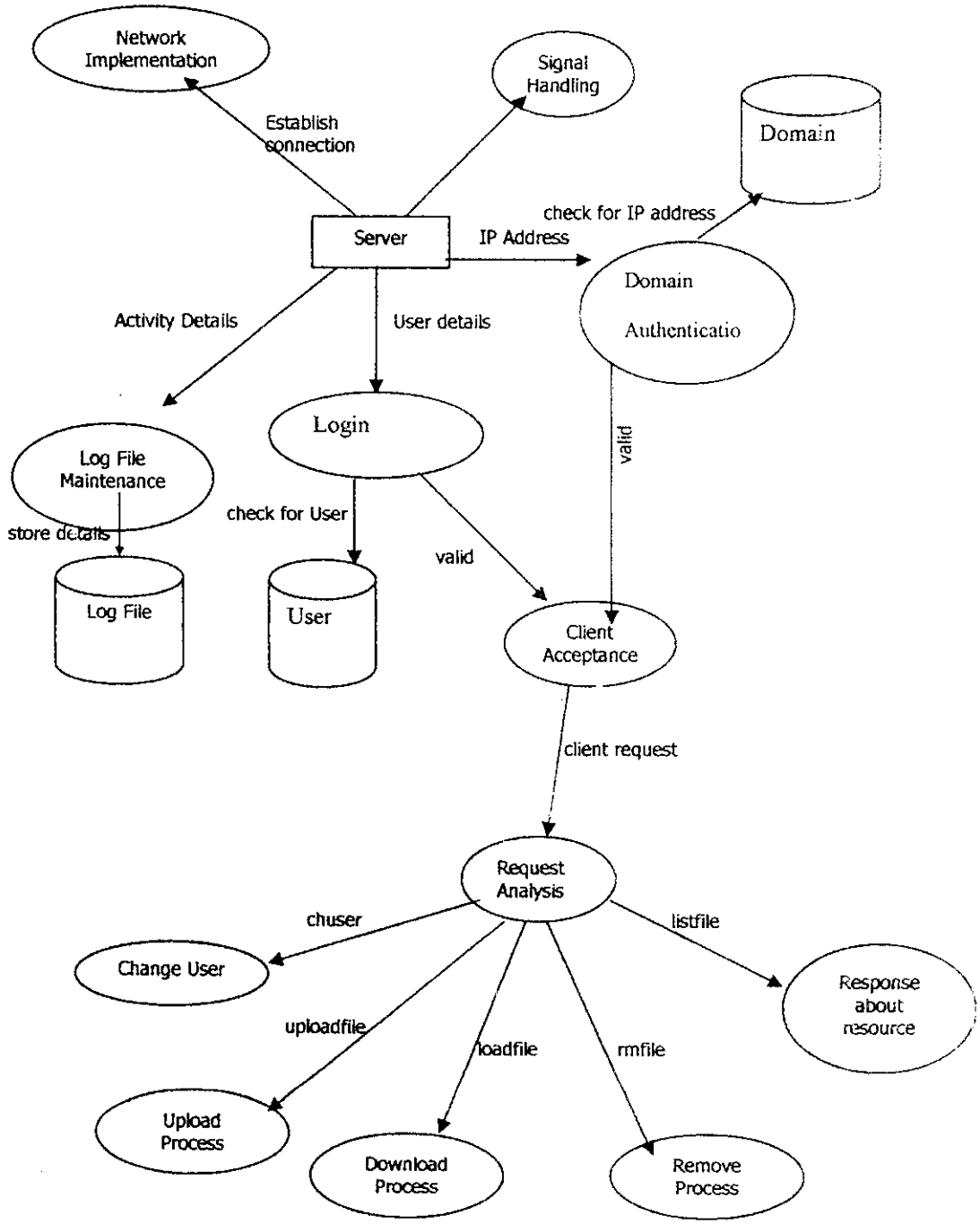
Administrator: User Process (DFD Level 2)



Client Process (DFD Level 1)

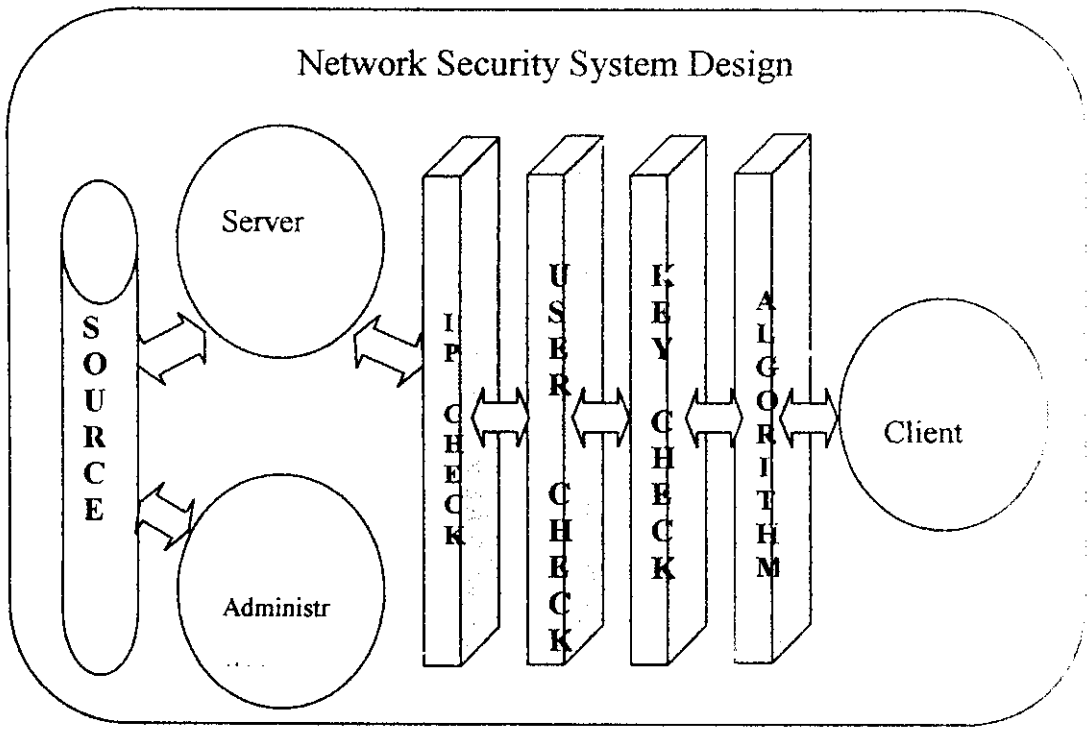


Server Process (DFD Level 1)

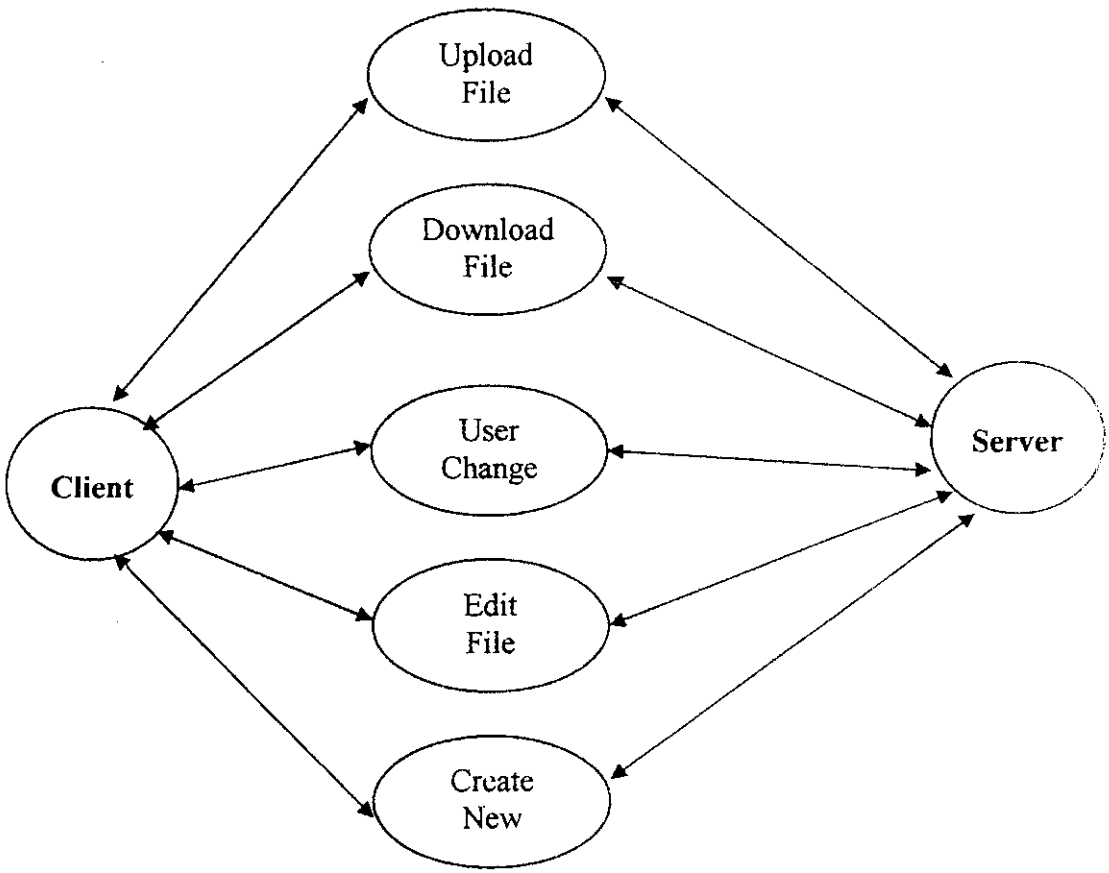


5.2.2 CONTROL FLOW DIAGRAM

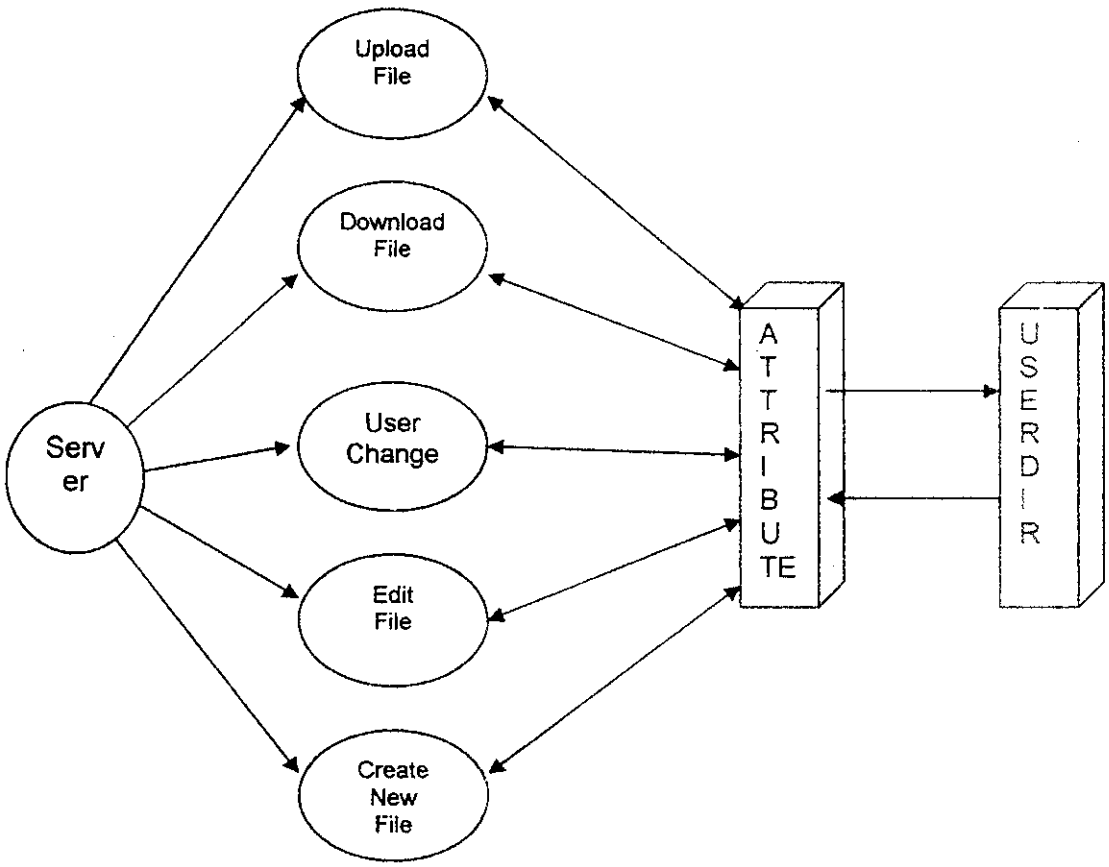
System Control Flow Diagram



Client Side Control Flow Diagram

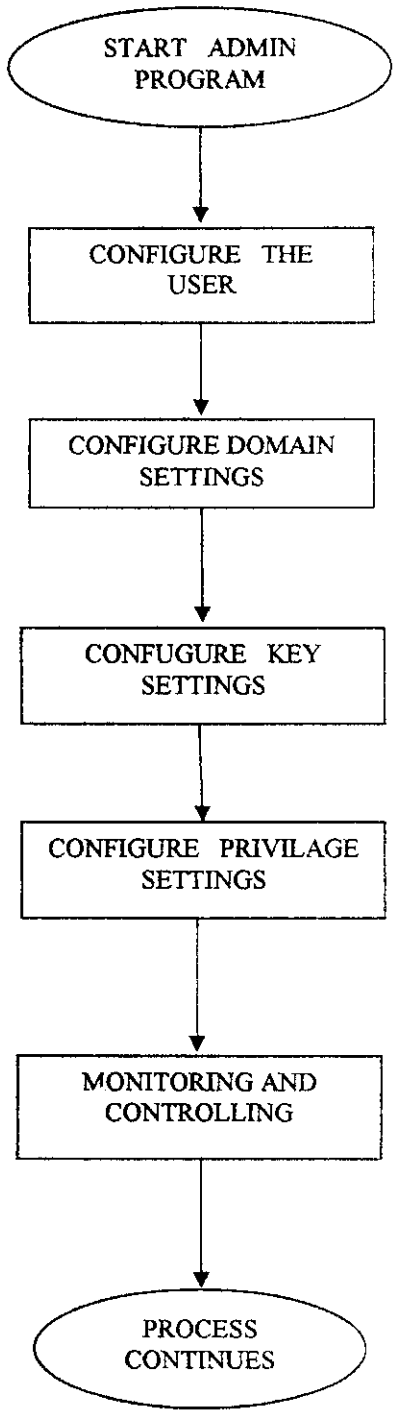


Server Side Control Flow Diagram

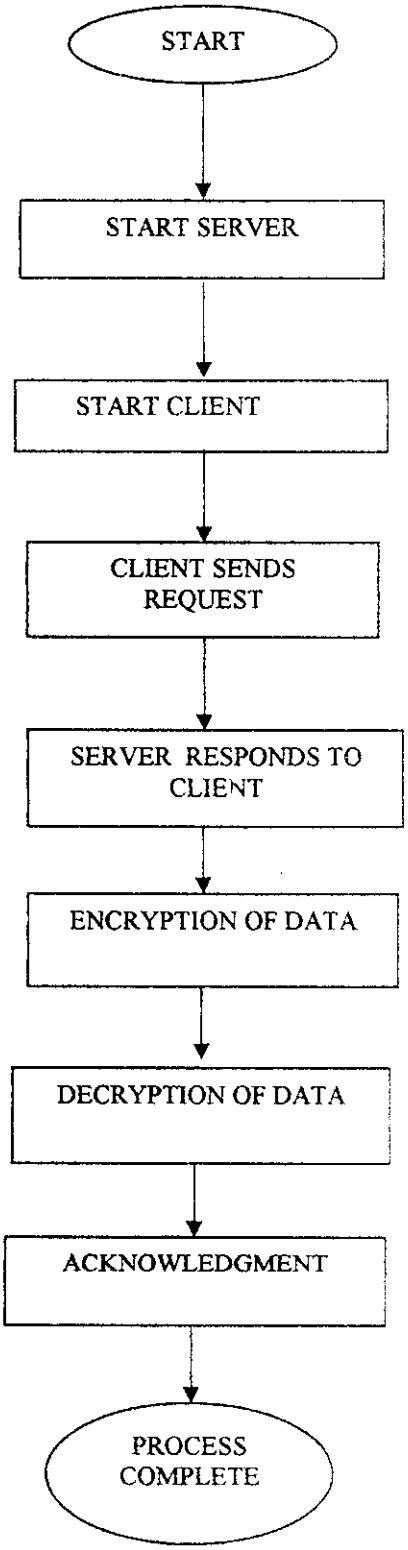


5.2.3 FLOW CHART

Administrator 's Part

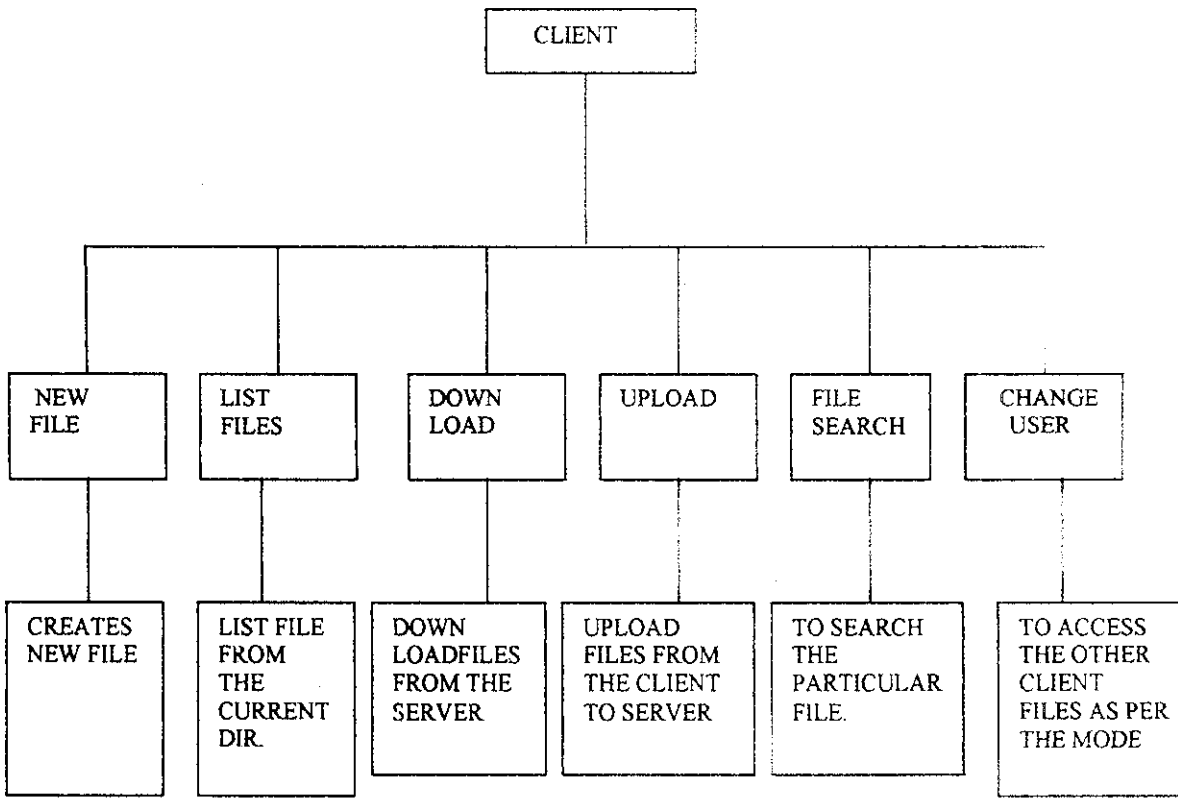


Server – Client Flow Chart

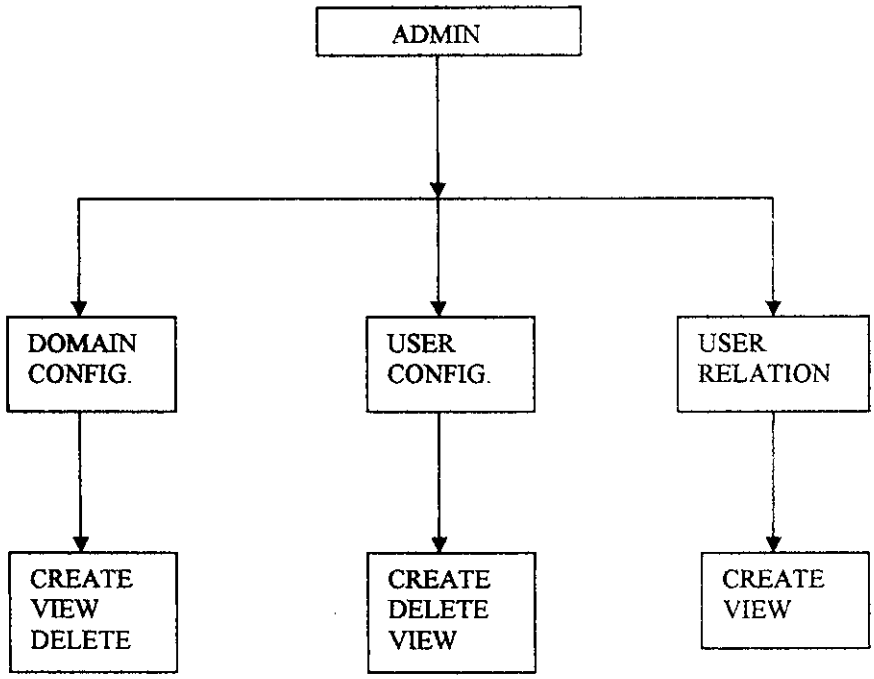


5.2.4 PROCESS LOGIC OF EACH MODULE

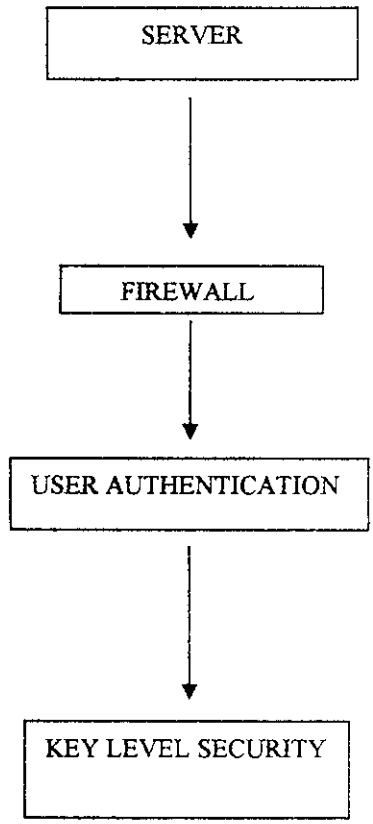
Client Module



Administrator Module



Server Module



6. IMPLEMENTATION DETAILS

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the users that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change over, an evaluation, of change over methods. Apart from planning, the major task of preparing the implementation is education and training the users. The more complex system being implemented, the more involved will be the system analysis and the design effort required just for implementation. An implementation coordinating committee based on policies of individual organization has been appointed. The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system.

Implementation is the final and important phase. The most critical stage in achieving a successful new system and in giving the users confidence that the new system will work effectively. The system can be implemented only after thorough testing is done and if it found to be working according to the specification. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain type of transactions while using the new system.

At the beginning of the development phase, a preliminary implementation plan is created to schedule and manage the many different activities that must be integrated into plan. The implementation plan is updated throughout the development phase, culminating in a change over plan for the operation phase. The major elements of implementation plan are test plan training plan, equipment installation plan and a conversion plan.

There are three types of implementation:

- Implementation of a computer system to replace a manual system.
- Implementation of a new system to replace the existing one.
- Implementation of a modified application to replace an existing one using the same computer.

After considering all the phases of the system life cycle, then the proposed system is now implemented successfully.

7. TESTING

Software Testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. Testing present an interesting anomaly for the software. Testing is vital to the success of the system. Errors can be injected at any stage during development. System Testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. During testing, the program to be tested is executed with set of data and the output of the program for the test data is evaluated to determine if the programs are performing as expected. A series of testing are performed for the proposed system before the system is ready for user acceptance testing. The testing steps are:

- Unit Testing
- Integration Testing
- Validation Testing
- Output Testing
- User Acceptance Testing

7.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of the software design. It comprises the set of sets performed by an individual programmer prior to the integration of the unit into a larger system. This testing is carried out during the coding itself. In this testing step each module is going to be working satisfactorily as the expected output from the module.

7.2 Integration Testing

Data can be lost across an interface; one module can have adverse effect on another, sub function when combined may not produced the desired function.

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated within the interface.

The objective is to take unit-tested module and build the program structure that has been dictated by design. All modules are combined in this testing. The entire program is tested as whole. Correction is difficult at this stage because the isolation of causes is complicated by the vast expense of the entire program. All the errors found in the system are corrected for the next testing step.

7.3 Validation Testing

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins.

Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of the two conditions exists.

- The function or performance characteristics confirm to specifications and are accepted.
- A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to the completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

7.4 Output Testing

After performing the validation testing, the next step is the output testing of the proposed testing. Since no system could be useful if it does not produced

the output in the specified formats. The output generator or displayed by the system under consideration is tested by asking the users about the format required by them. Here the output is considered in two ways: one is on the screen and the other one is in printed format. The output format on the screen is found to be correct as the format was designed in the system design phase according to the user needs. Hence the output testing does not result any correction in the system.

7.5 User Acceptance Testing

User acceptance of the system is a key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping touch with prospective system and user at the time of developing and making changes whenever required.

8. CONCLUSION AND FUTURE-OUTLOOK

CONCLUSION

Today in the world networking security plays the vital role. The network needs to be secured from intrusion and other illegal operations. These days connecting from one server to the remote server has become commonplace. There are many software's used to connect to the remote server, but none of them provide security while transmitting the data across the network. And also none of them have the ability to disconnect a particular user.

Our project aims at circumventing these problems and to provide a good interface to the administrator, to monitor and control the network. Our project overcomes the very important problem of providing security while transmitting the data.

This project is more a simulation of telnet, but it also provides some additional features which is otherwise not available in telnet. It will also be helpful for novice users who can explore the server with our interface. This project surely will have wide recognition and we hope it is in the right direction.

FUTURE OUTLOOK

System should provide extra features that is not otherwise available on LINUX system and provide necessary and easy interface to configure and monitor them staying locally and remotely so as to make administrator's work easier and more efficient .so software should be providing security on these with high flexibility and efficiency without the different configuration of Linux system.

1. Cryptography can be done using Developing algorithms
2. C++ an Object Oriented language can be used for programming.

9. REFERENCES

BOOKS

1. Maurice J.Bach, "The Design of the LINUX Operating System",Prentice Hall of India, 2001

2. Sumitabha Das, "UNIX , LINUX Concepts and Applications", Tata McGraw-Hill Publishing Company Limited,2002

3. Herbert Schildt , "The Complete Reference C", Tata McGraw-Hill ,2003

4. W. Richard Stevens, "UNIX Network Programming", Prentice Hall of India, 2002

WEBSITES

1. news:comp.os.linux.help , a useful site for Linux Basic helps.

2. news:comp.os.linux.networking , a definitive resource for what LINUX Network is and why it matters.

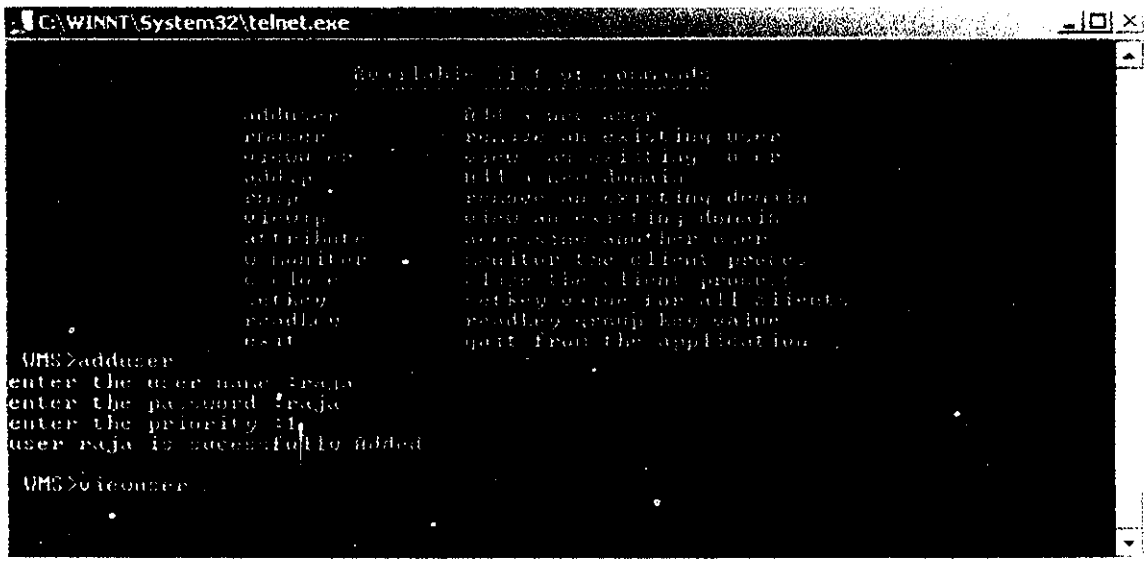
3. news:comp.os.linux.admin, an improved source for referring administrative details.

4. www.Linux.com , a well known site for Linux Links.

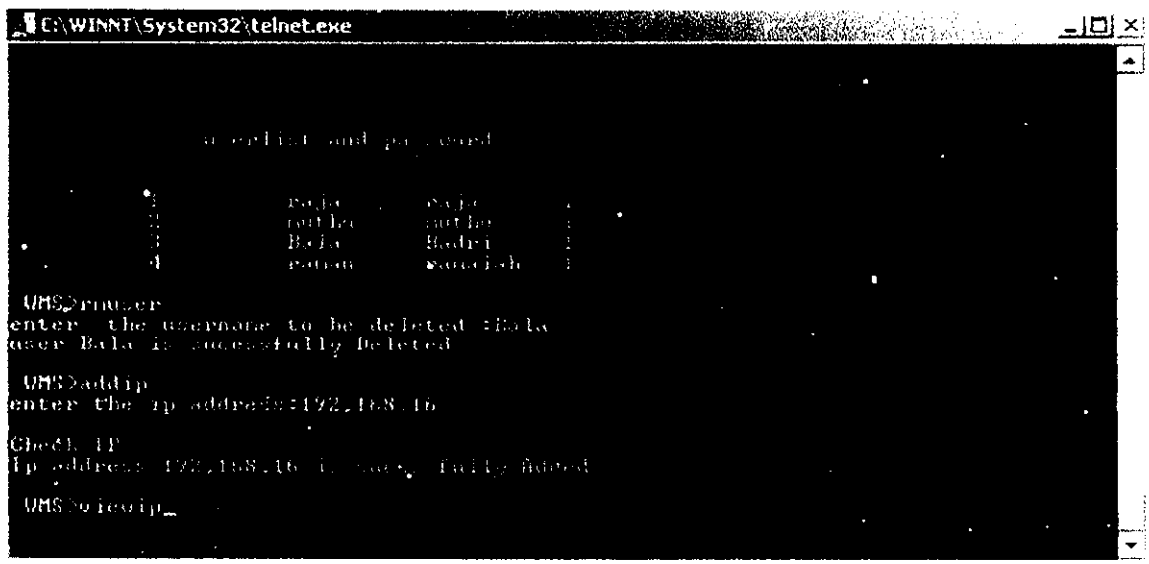
10. APPENDICES

10.1 OUTPUT SCREENS

Administrators Interface

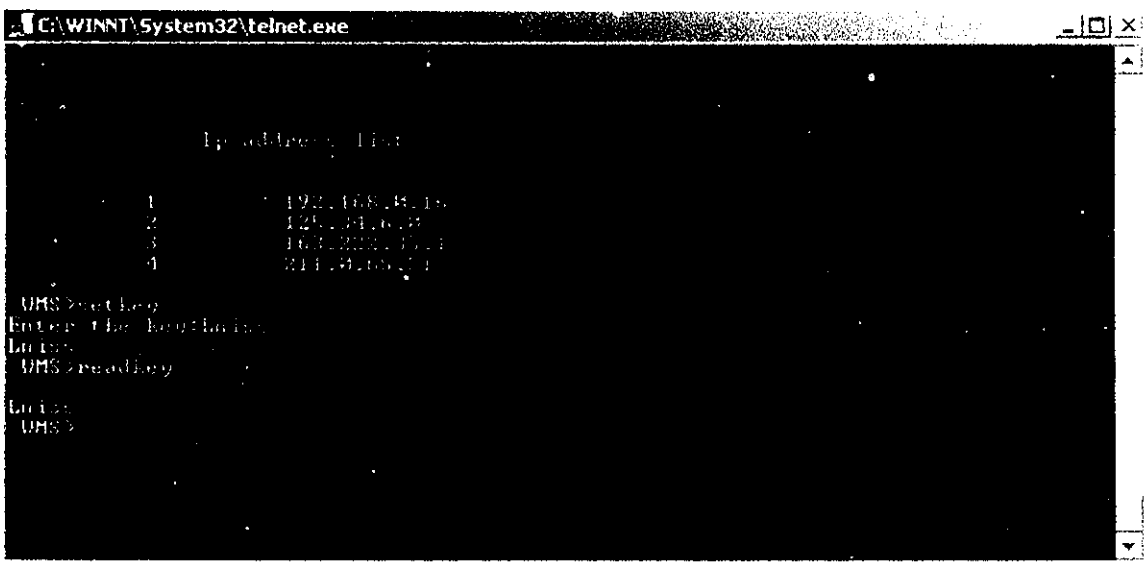


Viewing User's List

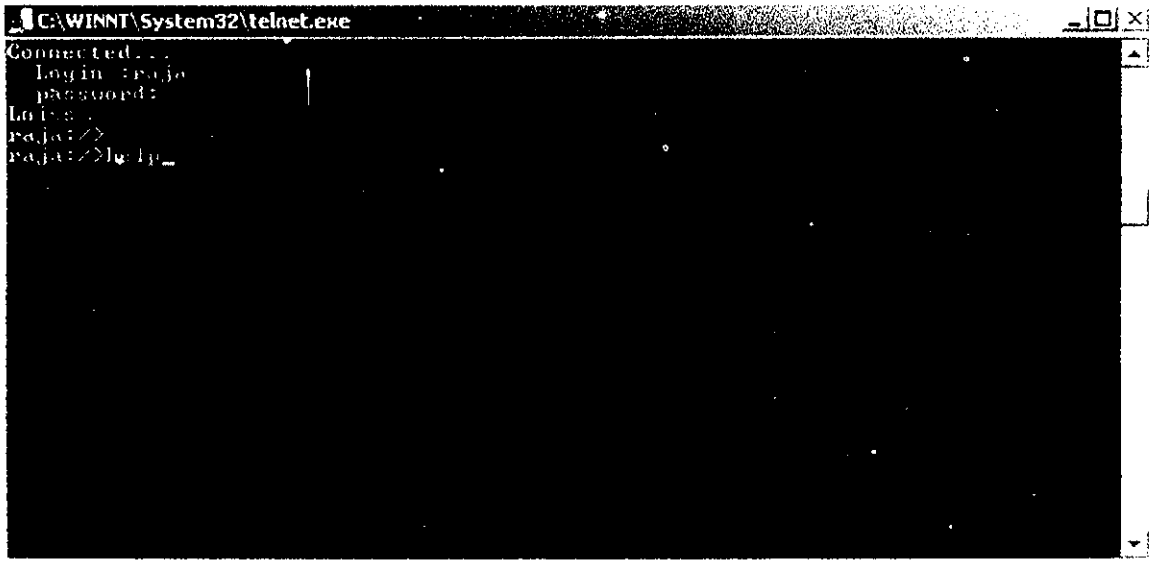


Reading The List Of Domains

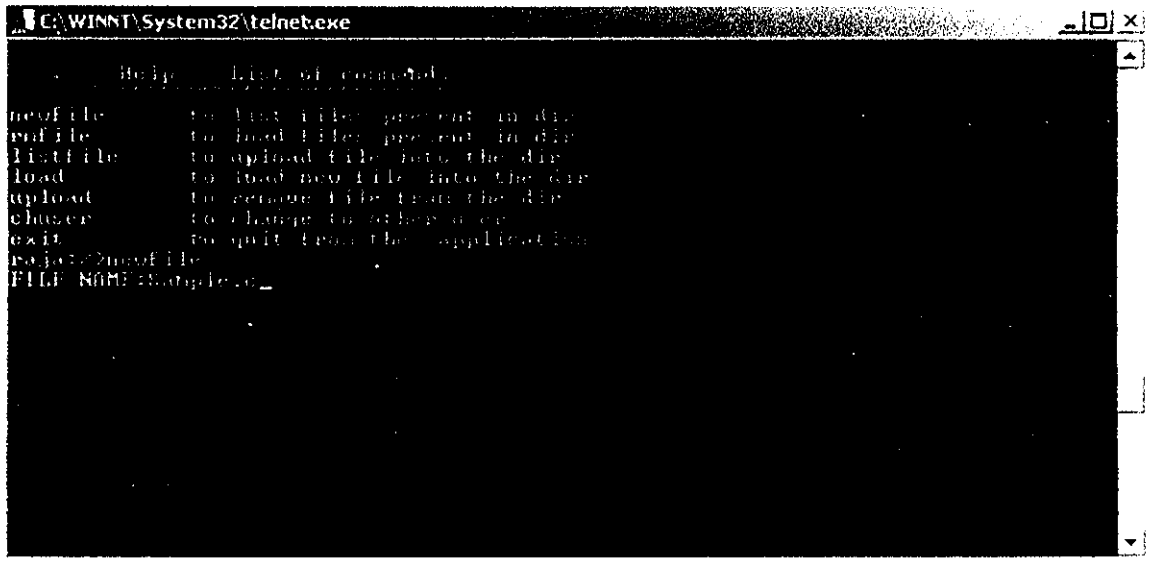
Importance Of Key



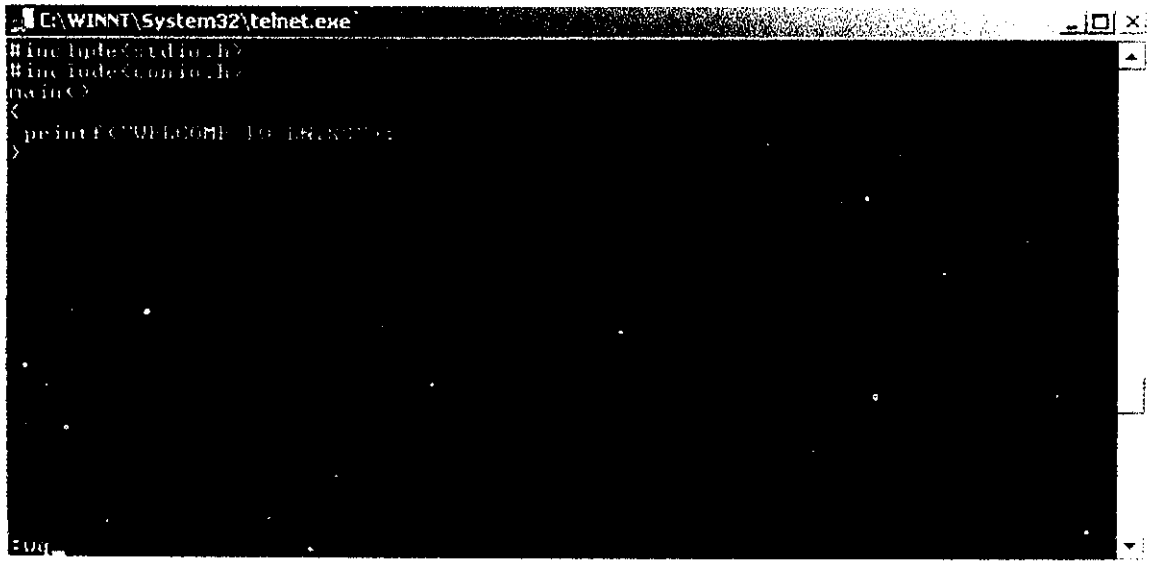
At The Start Of Clients Process During Connection



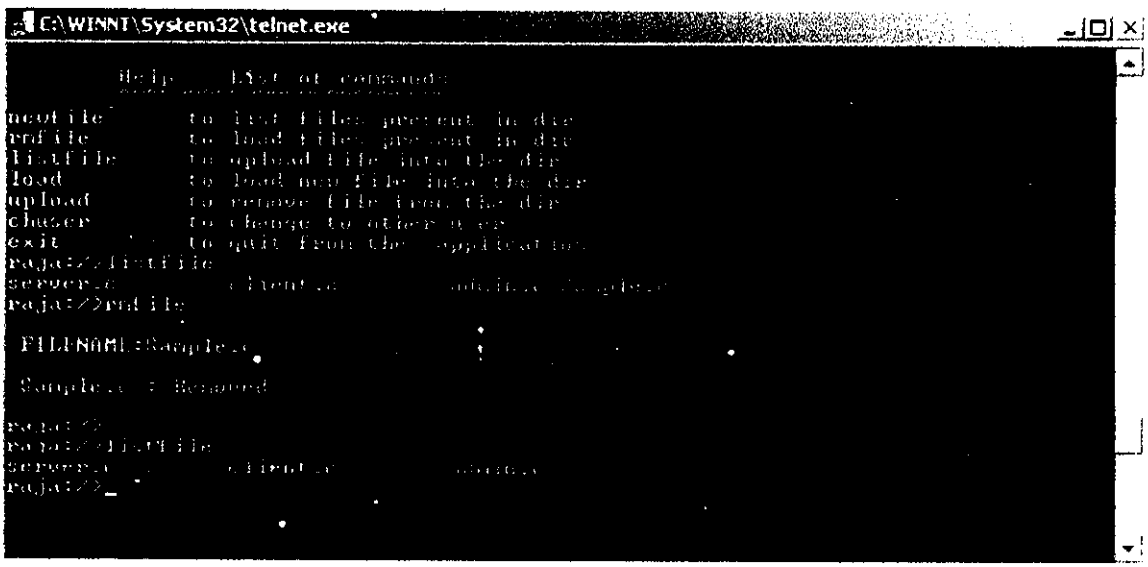
Clients Interface



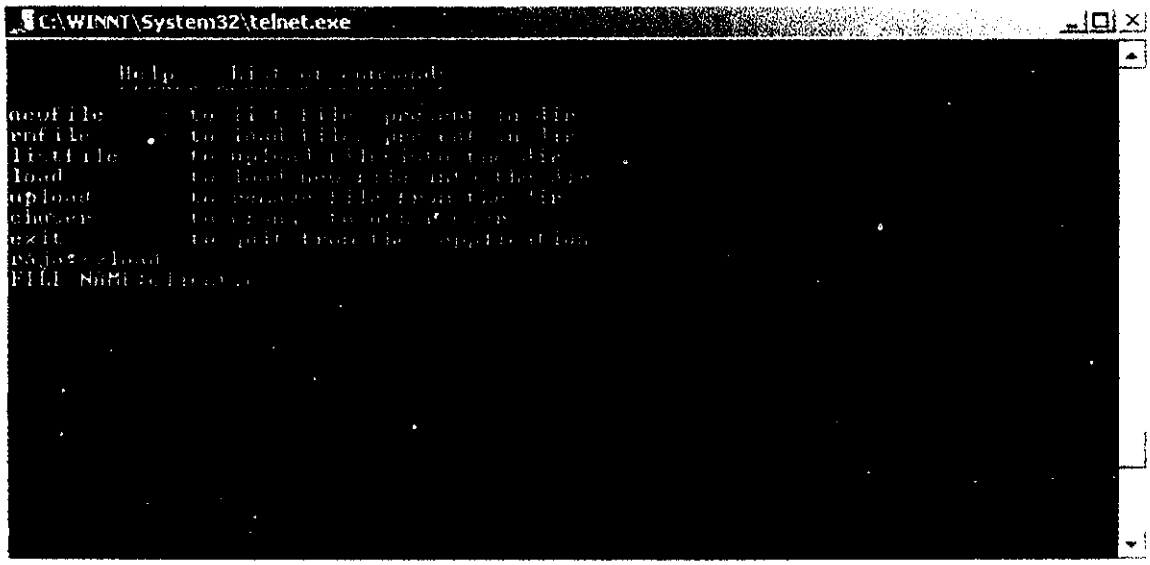
New File Creation

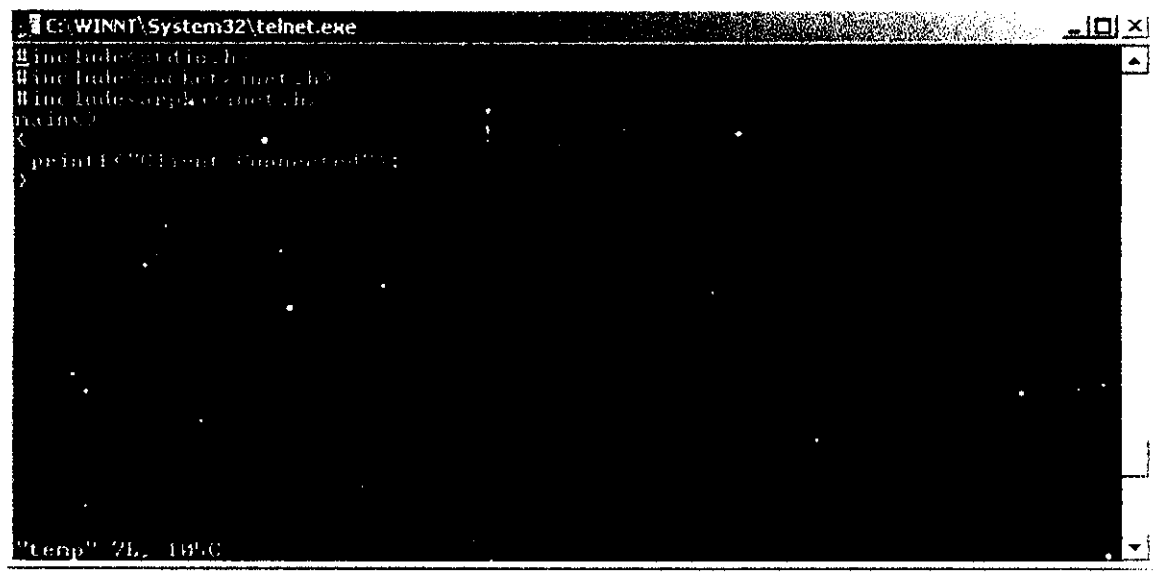


Manipulating The User Files

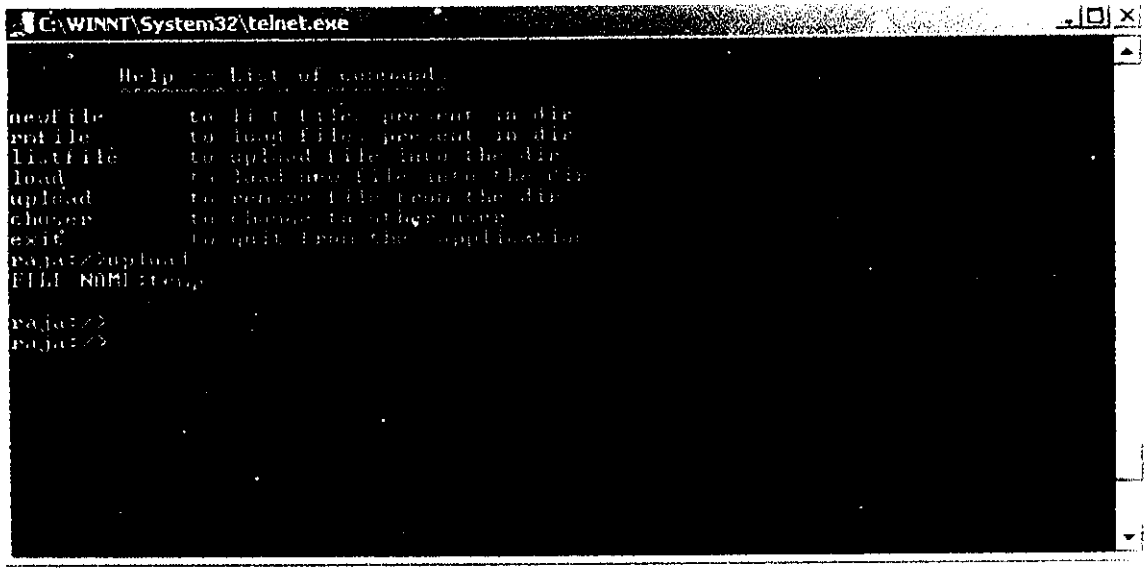


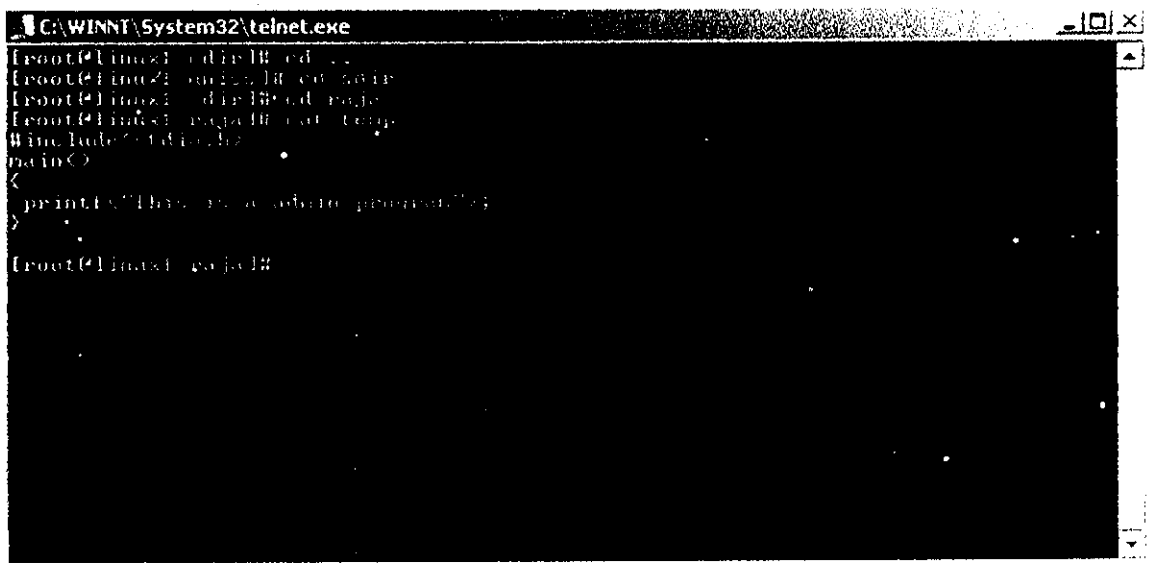
Loading File



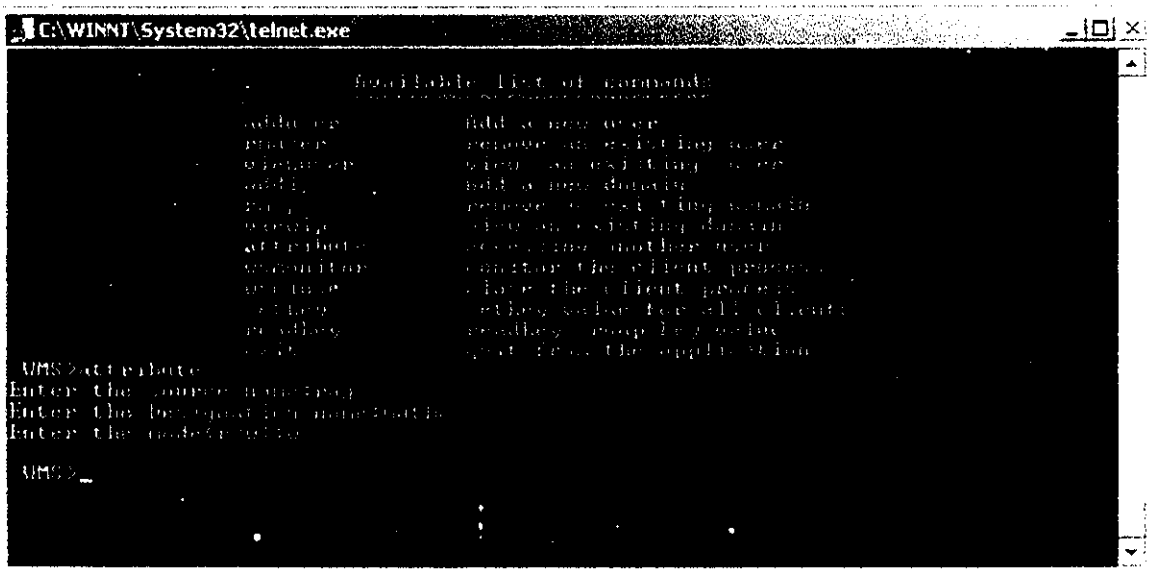


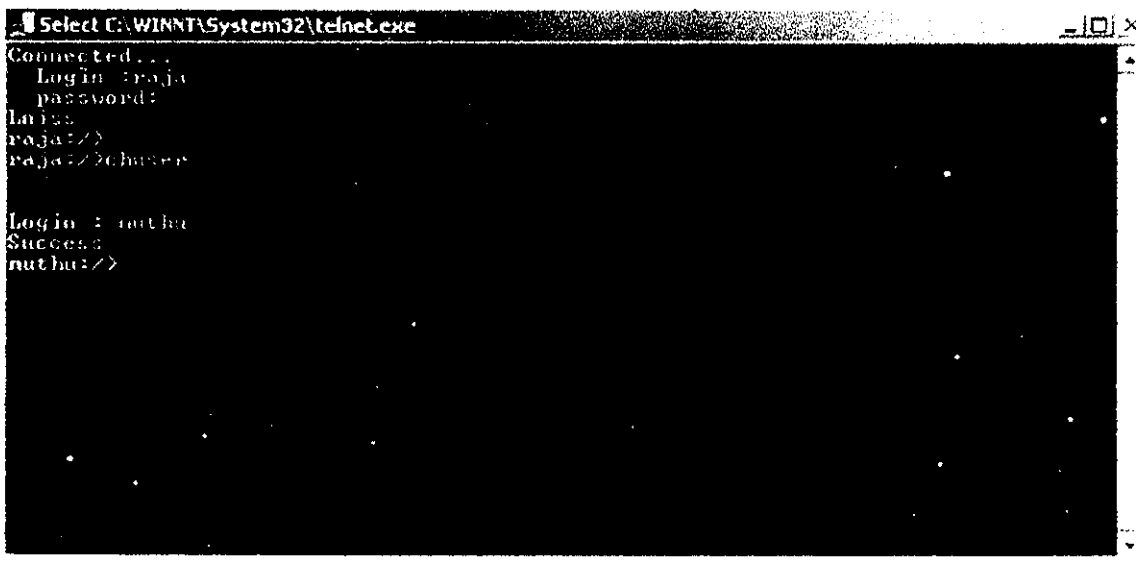
Uploading File



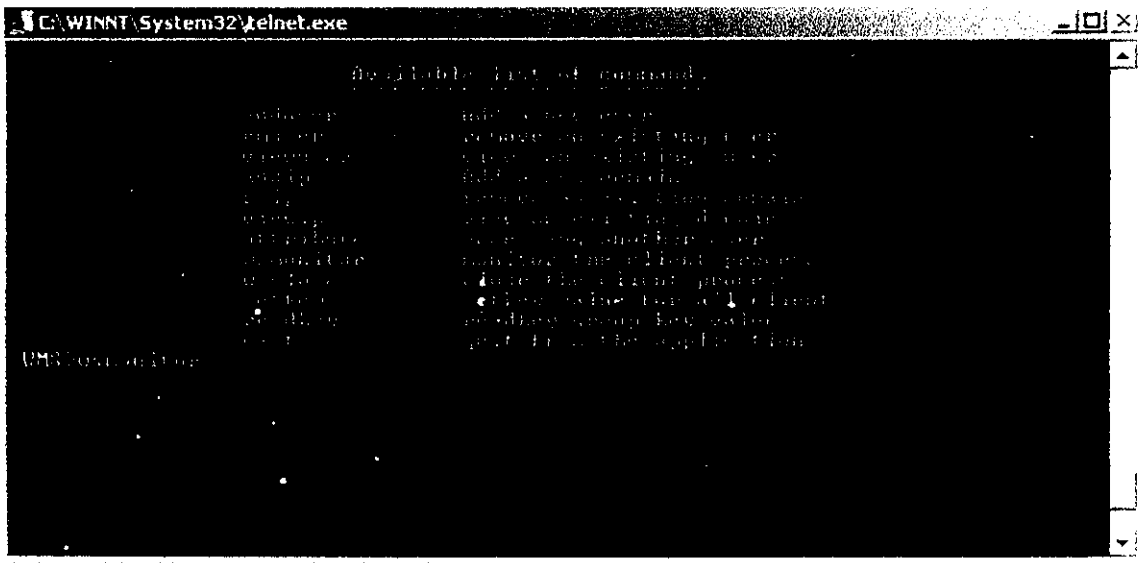


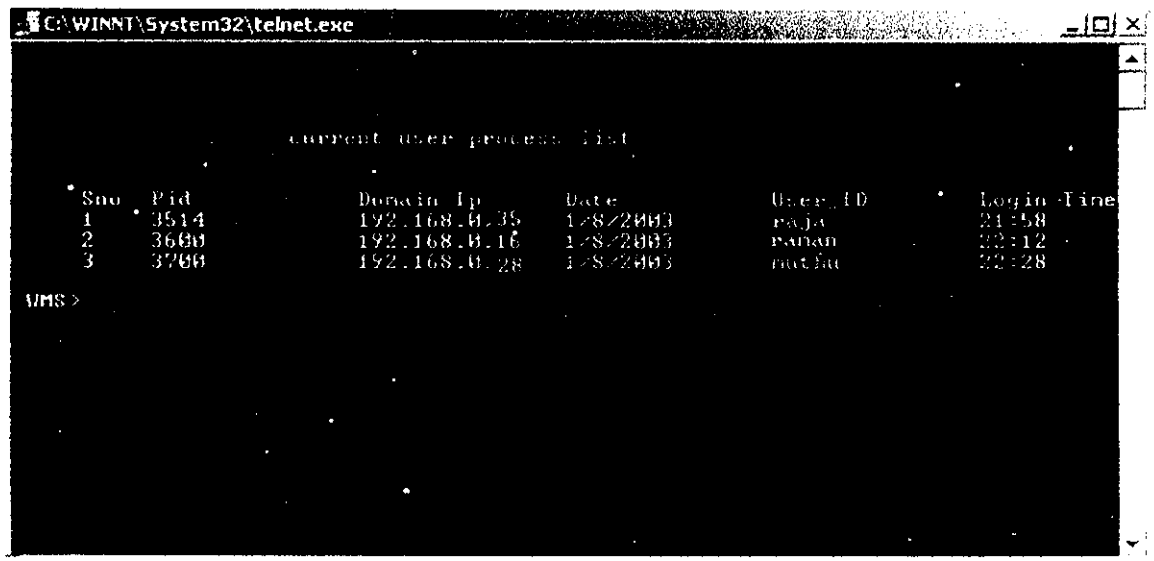
Attribute Setting



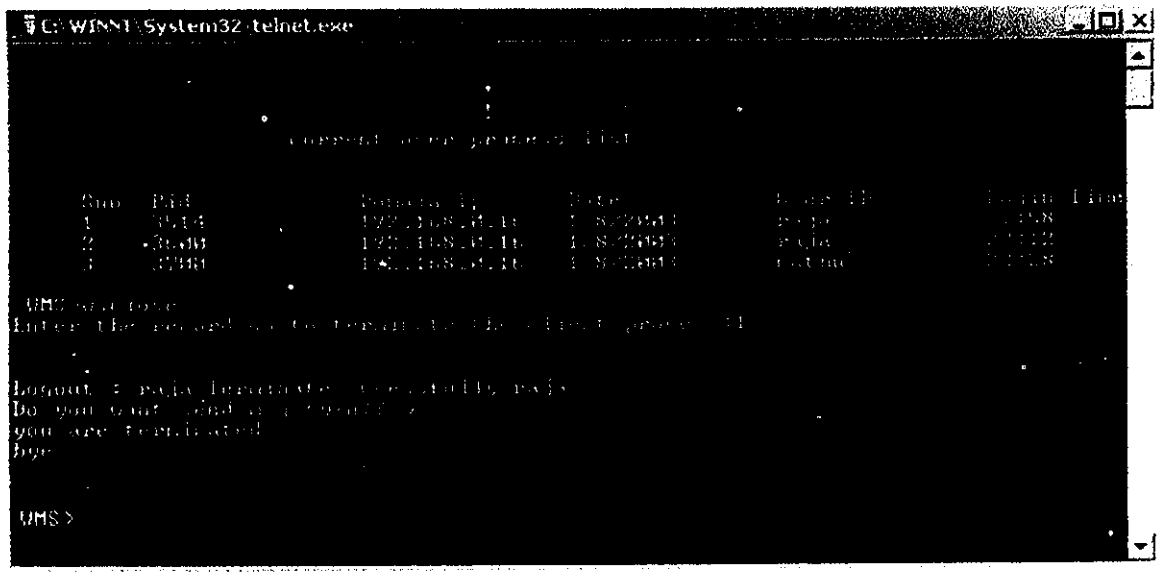


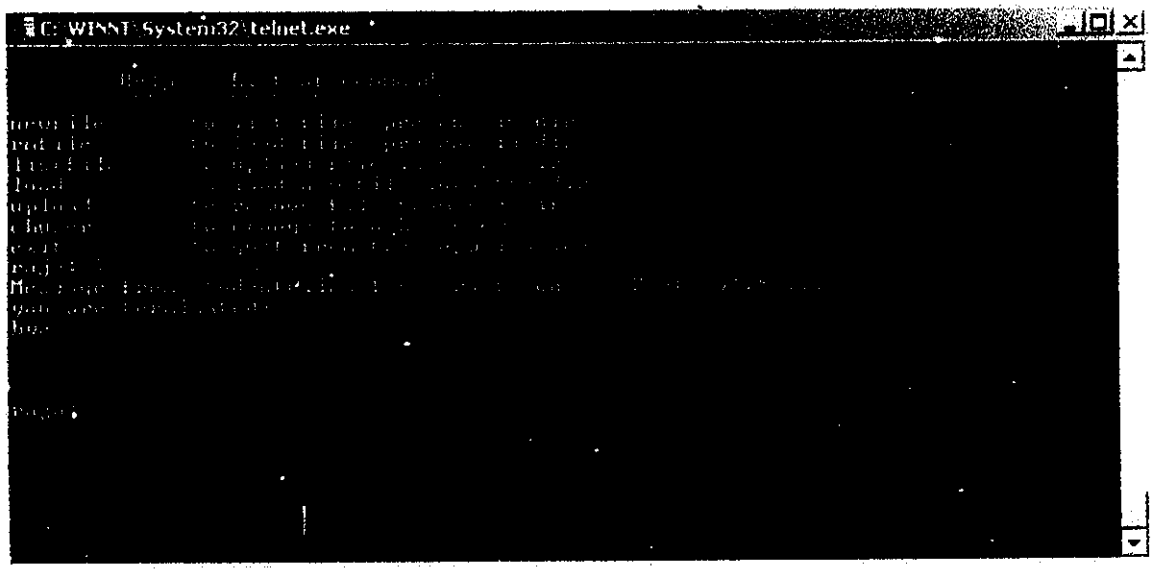
Viewing The Clients Connected In The Network

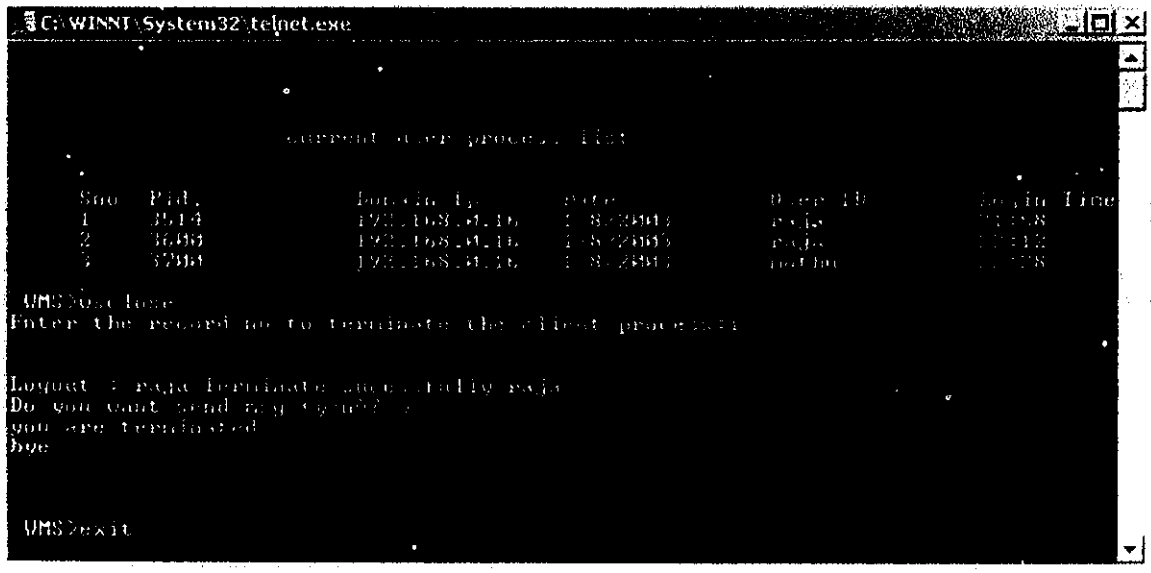




Terminating Specific Client In Network







Server After Connection

10.2 SOURCE CODE

The program here is classified in to three parts

1. Administrator part

2. Server part

3. Client part

Administrator part:

In the administrator part the following operations are carried out

1.User management

2.Firefall configuration

3.Key management

4.Monitoring and Controlling

5.Local Operations

6.Attribute Management

User management:

```
add()
 {
  fp=fopen("user.txt","a+");
  sprintf(md,"mkdir %s",var.name);
   system(md);
   chk = check(var.name);
  if(chk==1)
          printf("user %s already exist ",var.name);
  else
      {
          printf("enter the password :");
        scanf("%s",var.pwd);
```

```
        printf("enter the priority :");
      scanf("%d",&var.pri);

      fwrite(&var,sizeof(struct user),1,fp);
      printf("user %s is sucessfully Added\n",var.name);
    }
```

Description:

The above method is used to create a new user and store him permanently in the server.In this method FOPEN() function is used to create a new file or append details to an existing file.

When this function gets executed it prompts to enter the user name.If the username entered already exists it prompts for another name until the process is successful.

```
rm()
 {

   fp=fopen("user.txt","r");
   fp1=fopen("temp.txt","w");
   printf("enter  the id_no to be deleted :");
   scanf("%s",a);
   if(feof!=NULL)
    {

    while(1)
     {
         fread(&var,sizeof(struct user),1,fp);
              if(feof(fp)) break;
       if((strcmp(var.name,a)!=0))
```

```
        fwrite(&var,sizeof(struct user),1,fp1);
      else
        flag1='y';
    }
      if(flag1=='y')
      {
      printf("user %s is sucessfully Deleted \n",a);
         sprintf(buf,"rm -r %s",a);
         system(buf);
      }
    else
        printf("user %s is Not found",a);

     fclose(fp);
     fclose(fp1);
   remove("user.txt");
   rename("temp.txt","user.txt");
   }
}
```

Description:

By using this method the existing user is removed from the server.Here two files are opened one is the existing file and other one is the temporary file.The names from the original file is transferred to the temporary file.Then the temporary file is renamed.

Fopen()->To open a file for reading or writing

Fread()->To read records from the file

Fwrite()-> To write records in to the file

Remove()->To delete a file

Rename()-> To rename a file to another name

Firewall Configuration:

```
addip()
 {
    printf("enter the ip_address:");
   scanf("%s",var1.ip);
   ckip = chkip(var1.ip);
   printf("\nCheck IP\n");
    if(ckip==1)
       printf("Ip address %s is already exists\n",var1.ip);
    else
      {
       fp2=fopen("ip1.txt","ab");
       fwrite(&var1,sizeof(struct ipdet),1,fp2);
       fclose(fp2);
       printf("Ip address %s is sucessfully Added\n",var1.ip);
      }
}

 chkip(char * b)
    {
       struct ipdet var1;
      fp2=fopen("ip1.txt","rb");
```

```
if(fp2!=NULL)
 {
  while(1)
   {
     fread(&var1,sizeof(struct ipdet),1,fp2);
     if(feof(fp2)) break;
     if(strcmp(var1.ip,b)==0)
          {
           fclose(fp2);
           return 1;
          }
   }
  fclose(fp2);
 }
return 0;
}
```

Description:

In Firewall setting the network and the system from which the request must come is predetermined. In this setting a file is maintained in which the list of IP addresses which are allowed to access the information are maintained.

If any request arises from the client machine, the client machine is first checked for the legitimacy. If the IP address does not match with the existing IP address the client not allowed to connect to the server.
This firewall setting makes sure that the request from the authorised networks are served. This checking takes place before the userlevel checking.

Monitoring and Controling:

```
ws()
  {

   fp=fopen("cur_usr_data.txt","r");
   if(fp!=NULL)
    {
     istat=stat("cur_usr_data.txt",&buf);
     siz=buf.st_size;
  while(1)//!feof(fp))
    {
   fscanf(fp,"%s %s %s %s %s %s
%s",pid_v,domain,dt,username,tm,dname,tname);
        if(feof(fp)) break;
      // printf("\n%s %s %s %s %s %s %s",pid_v,domain,dt,username,tm);
       sprintf(buff,"%d",++i);
    }
   else
    {
       printf("\n No Client in  on ...");
    }
  fclose(fp);
 }
   else
    {
       printf("\n No Client in  on ...");
    }
}
   else
    {
```

```
        printf("\n No Client in  on ...");
    }
  fclose(fp);
 }
   else
    {
       printf("\n No Client in  on ...");
    }
}
```

Description:

This part of the program is used to monitor the network activity. With this method , the details about the client machines that are connected to the server,can be obtained. The other details about client, like username,login time ,tty,Pid ,login name etc. are maintained and can be obtained.

The procedure involved here is to store the details about the client when the client connects to the server.

The close window option is used by the administrator to shutdown a particular client who is misbehaving. This operation is performed by keeping track of the Process Id number. Once the PID value is known the process can be killed easily.

KILL()-> To kill a particular process.

Attribute Management:

```
setatt()
{
```

```
fp = fopen("attribute.txt","a");
 printf("Enter the source name:");
 scanf("%s",suser);
 wrdataflag=0;
 if(check(suser))
   {
     printf("Enter the Designation name:");
     scanf("%s",duser);
       if(check(duser))
         {
          printf("Enter the mode(r/w):");
          scanf ("%s",mode);
          if(mode[0] != 'r' && mode[0] != 'w')
           {
            printf("Invalid no");
            return 0;
           }
              wrdataflag=1;
        }
         fclose(fp);
    }
   if(wrdataflag==1)
      {
      fp = fopen("attribute.txt","r");
      if(fp != NULL)
       {
       wrdaflag=0;
       while(1)
        {
              fscanf(fp,"%s %s %s",fsu,fdu,fmod);
```

```
            if(feof(fp)) break;
            if(!strcmp(fsu,suser) && !strcmp(fdu,duser))
            {
                    wrdaflag = 1;
                    break;
            }
        }//while(1)
       }//if(fp !=NULL)

            }//while(1)
           fclose(fp1);
           fclose(fp2);
           unlink("attribute.txt");
           rename("attribute_1.txt","attribute.txt");
         }//else
   }//if(wrdataflag==1)
}
```

Description:

The attribute management function is updated by the system administrator according to the requirement. The main use of this feature is to access the files of the other users. A particular user can give access permissions to the other users through this feature.

The various permissions are

1. read permission
2. write permission

Read permission:

Read permission allows a particular user to access the files of other user,but he will not be allowed to update the file or to create a new file in that directory.

Write permission:

Write permission gives all the privileges to the user. He can read the existing data ,he can update the existing data, and also can create a new file in the server.

Local Operations:

This feature allows the system administrator to access the local machine in which he is currently operating. He can perform all the operations in the local machine as well as monitor the remote machine.

Server Process:

The server process takes care of the following operation:

1. Listing files from the server
2. Uploading the file in to the server
3. Downloading the file from the server
4. Removing the file from the server.

Listing Files:

```
int listfile(int cfd)
{
 struct dirent ** namelist;
  n = scandir(uname,&namelist,0,alphasort);
```

```
 strcpy(buff,"");
while(n--)
 {
   if(namelist[n]->d_name[0]!='.')
     {
      strcat(buff,namelist[n]->d_name);
      strcat(buff,"\t");
     }
    free(namelist[n]);
 }
 free(namelist);
 write(cfd,buff,sizeof(buff));
 return;
}
```

Description:

Once the listfile request is made by the client the control is transferred to the listfile function. In this function the server process gets the name user currently logged ing. It then moves to the login users directory.From that directory the list files is read recursively with the help of the SCANDIR() function.

The Scandir() function returns the value as string array.

Char * ScanDir(name,&namelist,0,alphasort)

Namelist array will hold the file names from that directory.

Uploading Files:

```
int newfile(int cfd)
{
   read(cfd,fname,sizeof(fname));
    strcpy(path,uname);
```

```
  strcat(path,"/");
  strcat(path,fname);
   read(cfd,(int*)&siz,sizeof(int));
  fp = fopen(path,"w");
  data=(char *) calloc(sizeof(char),siz);
  read(cfd,data,siz);
  write(1,data,siz);
  fwrite(data,siz,1,fp);
  fclose(fp);
  return;
}
```

Description:

During the uploading process the client machine encrypts the data and passes the encrypted data to the server machine. The server process reads the data and stores the data in the filename mentioned.

Read(int sockid, char * buff,int size)-> To read the data from a process.

Write(int sockid,char * buff, int size)-> To write data to a particular process

Client Process:

The operations involved with client process are:

1. To list file
2. To create a file
3. To load a file
4. To remove a file
5. To change to different user

List file:

```
int listfile(int sfd)
```

```
{
   read(sfd,buff,sizeof(buff));
      printf("%s",buff);
    return;
}
```

Description:

In this method just the request listfile is made the server recognises the requirement and gives a list file. The client process waiting for the response gets the string value.

Create File :

```
int newfile(int sfd)
{
   printf("FILE NAME:");
   scanf("%s",fname);
   strcpy(fna,fname);
   sprintf(buff,"vi %s",fna);
   system(buff);
   write(sfd,fna,sizeof(fna));
    sprintf(buffer,"./e %s temp2",fna);
    system(buffer);
   istat = stat("temp2",&buf);
   siz= buf.st_size;
   data = (char *) calloc(sizeof(char),siz);
   write(sfd,(int *)&siz,sizeof(int));

    return ;
}
```

Description:

This method prompts the user the type in the name of the file. After that, a vi editor screen is opened for the user. The user can type the required data and can store the data. Before passing the data to the server the file is encrypted with help of the algorithm. The encrypted format of the data is then passed on to the server.

sprintf(buffer,"./e %s temp2",fna);

In this function `. /e` the output file of the algorithm takes ordinary file as input and produces encrypted data as output.

Change User:

```
int chuser(sfd)
{
        printf("\n\nLogin : ");
        scanf("%s",uname);
        write(sfd,uname,sizeof(uname));
        read(sfd,(int *)&flag,sizeof(int));
     if(flag==1)
                printf("Success");
        else
                printf("Login Fail");
return flag;
}
```

Description:

The change user function allows a particular user to login as anothe user. Once the login process is successful the current user can access the files o other users.