Kumaraguru College of Technology

Department of Computer Science and Engineering
Coimbatore-641 006

ISO 9001:200

P- 1075

FILE TRANSFER PROTOCOL (WITH FILE FIND AND FILE WATCH)

Project work done at

SRM SYSTEMS AND SOFTWARE LIMITED

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

B. POORNIMA
Reg.No-0037S0095

INTERNAL GUIDE

Mrs. V. VANITHA. M.E.,
Dept.of Computer Science& Engineering,
Kumaraguru College of Technology,
Coimbatore.

EXTERNAL GUIDE

Mr.S.Kamesh B.E,
SRM SYSTEMS AND SOFTWARE. Ltd,
CHENNAI

Department of Computer Science and Engineering
KUMARAGURU COLLEGE OF TECHNOLOGY
Coimbatore – 641 006

ISO 9001:200

CERTIFICATE
PROJECT REPORT 2003

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

B. POORNIMA
(Reg. No. 0037S0095)

Mrs. V. VANITHA. M.E.,
Project guide
Computer Science & Engineering

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

Place: Coimbatore
Date: 29.09.03

Submitted for University examination held on

29.09.2003

Internal Examiner

29/9

External Examiner

RM SYSTEMS AND SOFTWARE LIMITED
-, G.N. Chetty Road, T.Nagar, Chennai - 600 017.
: 91 - 44 - 8250771, 8258757, 8269471 Fax : 91 - 44 - 8283359
mail : srm@srmsoft.co.in Web Site : http://www.srmsoft.com
:gd. Off : 2, Veerasamy St., West Mambalam, Chennai - 600 033.

SRM
SYSTEMS AND SOFTWARE

23.09.2003

CERTIFICATE

This is to certify that the project work entitled "**FILE TRANSFER PROTOCAL**" was Analyzed, Designed and Developed by **Ms. B.POORNIMA** of **KUMARAGURU COLLEGE OF TECHNOLOGY (CBE),** submitted in partial fulfillment of the requirements of degree of 4th **Year M.Sc (S.E)** has been carried out in our organization from June 2003 to Sep 2003.This project has been developed using **VC++.**

We wish him success in all his future endeavors.

For SRM System And Software Limited

Manager-Projects

DECLARATION

I here by declare that the project entitled **"FILE TRANSFER PROTOCOL"**, submitted to **Kumaraguru College of Technology**, Coimbatore Affiliated to Bharathiar university as the project work of **Master of Science in Applied Science Software Engineering** ,is a record of original work done by me under the supervision and guidance of **Mr.S.Kamesh.B.E,** SRM Systems And Software Ltd., Chennai and **Mrs.V.Vanitha,M.E.,** Lecturer of Kumaraguru College of Technology, Coimbatore and the project work has not found the basis for the award of any Degree/Diploma/Associateship/Fellowship or similar title to any candidate of any University.

Place: Coimbatore

Date:

ACKNOWLEDGEMENT

I am immensely grateful to **Dr.K.K.Padmanaban BSc(Engg) , M.Tech., Ph.D.,** Principal , Kumaraguru College of Technology for his valuable support to come out with this project.

I really feel delighted in expressing my heartful thanks to **Dr.S.Thangaswamy Ph.D**, Prof & Head of Department of Computer Science and Engineering for his endless encouragement in carrying out this project successfully.

My heartfelt thanks to our project coordinator **Mrs.S.Devaki B.E., M.S,** Assistant Professor, for his unfailing enthusiasm, encouragement and guidance that paved me to the completion of this project.

I am indent to express my heartiest thanks to **Mrs.Vanitha.M.E** my project guide who rendered his valuable guidance and support to do this project work extremely well.

I am greatly indebted to chairman SRM Systems And Software Limited, chennai, for getting us into his esteemed institution. I also thank **Mr.S.Kamesh** B.E who was my guide and he has helped me a lot in my project.

I am also thankful to all the faculty members of the Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore for their valuable guidance, support and encouragement during the course of my project work..

My humble gratitude and thanks to my parents who have supported, to complete the project and to my friends, for lending me valuable tips, support and cooperation through out my project work.

SYNOPSIS

"FILE TRANSFER PROTOCOL" is an application software developed for the SRM Radiant Infotech.

With the development of computers file transfer among networks via intranet protocols has become an inevitable part of advanced computing.

This has given rise to the need for efficient and flexible file transfer applications for end-user computing.

The software is installed in the communicating systems. File sending is accomplished by browsing for the specific file and mentioning the IP address of the destination workstation. In case of transfer of random file the file find application is made use of to locate the specified file and then is sent across the LAN.

The File Transfer is enhanced by the presence of the status bar indicating percentage of transfer data and rate of transmission. The receiver can at any time exit the application by making use of the EXIT command.

On completion of the transfer the file watch application is made use of to assess the status of the file in case of any modification prior or after the transfer. The user has the flexibility to either accept or ignore these modifications thus providing file security.

The File Find and File Watch applications are bundled along with FTP to enhance efficiency and security.

CONTENTS

1. INTRODUCTION

1.1 CURRENT STATUS OF THE PROBLEM

The "FILE TRANSFER PROTOCOL" promotes sharing of files and encourages indirect use of remote computers. The main objective of the current system is to eliminate the use of commands for File Transfer and to provide Graphical User Interface(GUI). File Transfer protocol uses TCP/IP to transfer files across the network. The higher layer, Transmission control protocol, manages the assembling of a message or file into smaller packets that are transmitted over the Internet and received by a TCP layer that reassembles the packets into the original message. most unique features have been its robust stop and keep alive commands. The Exit command, similar in nature to the stop button found on many Web browsers, allows a user to stop any operation in progress while still maintaining the connection to the destination workststion. The Status Bar helps to ensure the accurate transfer of data without interruptions.

1.2. RELAVANCE AND IMPORTANCE

1.2.1. EXISTING SYSTEM

The Existing system uses Command User Interface(CUI).The commands are executed in MSDOS using TELNET protocol. The Trivial File Transfer Protocol(TFTP)is an internet utility software but it is less capable than File Transfer Protocol.The TFTP uses UDP(User Datagram Protocol) for file transfer which is not user interactive.The existing system does not provide any acknowledgement for the files that are transferred.
The Drawbacks of the Existing System are

- Commands and replies are all Line Oriented Text
- Uses lots of commands to transfer data
- It is used where user authentication and visibility are not required
- Time consuming

1.2.2. PROPOSED SYSTEM

In the Internet world of accessing sharing and transferring of files remotely becomes an important concept. In the proposed system,the files are found using the file find module and then the files are transferred to the destined work station. If the files are modified outside the application a message box pops up and lets the user choose between ignoring modifications or reloading the data thus discarding all changes. In order to increase the efficiency of file transfer the proposed system is developed using MFC classes in VC++
The advantages of using the MFC classes are:
Complete support for all Windows functions.
Small code and almost as fast as C implementation
Support for Component Object Models(COM)

The proposed system eliminates the drawbacks of the existing system

- Graphical User Interface
- Less Time Consuming
- User Friendly
- User Interactive
- Efficient Transfer

2. COMPANY PROFILE

SRM SYSTEMS AND SOFTWARE is a company committed to provide support to small, medium and large corporations in the development and management of software essential to their needs over the entire life cycle of a project or system. All corporations, regardless of size, need to process enormous amounts of data in support of the day-to-day operation of the company and the dependence on a corporate information system and up-grade the existing ones. In seeking efficient and cost-effective approaches to manage change, many companies have found outsourcing to be particularly attractive.

SRM Systems and Software is here to provide expert services and support for "change management" in software systems allowing your organization to focus on its core business. SRM Systems and Software offers the expertise of experienced individual software consultants, as well as an offshore facility with a state-of-the-art information technology infrastructure and a well-trained and committed staff, all at extremely competitive prices. We at SRM provide our clients the potential for significant savings without a compromise in quality or schedule. SRM Systems and Software guarantees that the software services will be delivered to the customer on time, within budge, incomplete conformance means that at SRM, we are indeed
" Determined to Make a Difference".

MISSION STATEMENT

The stated mission of the SRM System and Software is to offer value addition to the customer's Business through IT Solutions of high quality and appropriate Technology on time and on budget.

CORPORATE BACKGROUND

- Reputation built over 3 decades
- Global vision

- Asset base of over US $100 million
- Many interests but one objective - Commitment to Excellence

SRM Systems and Software is a unit of the renowned SRM Group, which in the past30 years has established itself in Southern India in the field of Engineering education and Research. Over the years, the SRM Group, with an asset base of more than US $ 50 million, has expanded into the fields of Health Care, Hospitality, Manufacturing, Financial Services and Construction.

SRM Systems and Software was established with a specific business focus on Software Development and consultancy. As a member of the Software Technology Park of India, SRM Systems and Software benefits through business and customs duty incentives from the Government of India and consequently is committed to export 100% of its products and services.

The overseas office of SRM Systems and Software in Boston provides an effective link to customers in the United States and other parts of the world. Efforts are under way to establish similar offices in Japan, UK, Europe and Australia. Connected by broadband data links, the Headquarter in Chennai and the overseas offices will be positioned to provide customers global information technology market by an unwavering commitment to quality.

SRM Systems and Software - A Customer Centric Company

OBJECTIVES

- World Class Products
- Commitment to Quality
- Impeccable Customer Service
- Excellent Technical Support

BUSINESS ETHICS

- Customer is God
- Work is Worship
- Employee is Strength
- Humanity is the Base

STRATEGY

- Our International strategy is to penetrate and service the market by On-site, Off shore & Turnkey projects based on our expertise and related software solutions
- Our Domestic Strategy in India is to increase market share, expand Client base and focus on large IT contracts.

UNIFIED STRENGTH

- Three decades of SRM's Track Record
- Strong Team Work
- Excellent Technical Competence
- Structured Project Approach
- Customer Centric and Focus on Customers' Customers
- Japanese Language Competence

SERVICES OFFERED

SRM Systems and Software through its Strategic Business Units offers the following services.

CUSTOMIZED SOFTWARE DEVELOPMENT

SRM can provide complete business turnkey solutions to small, medium and large size companies spanning every phase of the software life cycle: System Analysis, Design, Implementation, Testing, Installation and Maintenance. The SRM staff has an accumulated experience of more than 300 man-years in varied application areas. SRM offers software services in the following technology areas:

- Web Based Applications and e-commerce
- Client-Server (two-three and n-tier Technology)
- Group Ware and Workflow
- Multimedia and Computer Graphics
- Computer Aided Design and Computer Aided Manufacturing

SRM guarantees each customer that any project executed by SRM will be developed as per the specifications, delivered on time, and without cost overrun. SRM strictly adheres to the latest Software Engineering standards in the development of customized software. The aim of SRM is to win the allegiance of each customer so that the relationship does not end with the completion of the first contract but becomes an ongoing and mutually beneficial association.

3.HARDWARE AND SOFTWARE REQUIREMENTS

3.1. HARDWARE SPECIFICATION

System	Pentium @600mhz
Cache	128 MB
RAM	128 MB
Hard Disk	20 GB
NIC	LAN card

3.2. SOFTWARE SPECIFICATION

Operating system :Windows workstation

Software Required :Visual C++

4. PROPOSED APPROACH TO A PRODUCT

Software used to develop the system is Visual C++,

4.1. VISUAL C++:

Visual C++ has various features for which it is selected. It has very good compiling tools. Some of the features of Visual C++ are

1 Supports network communication programs

2 Supports ActiveX, ODBC, OLE

3 Easy to handle graphics and animation

4 Easy to write threading applications

Other features of Visual C++ are given below.

4.2. THE VISUAL C++ ENVIRONMENT:

An IDE, or Integrated Development Environment, is a program that hosts the compiler, debugger, and application building tools. The central part of the Visual C++ package is Developer Studio, the Integrated Developer Environment (IDE) developer Studio is used to integrate the development tools and the Visual C++ compiler. You can

create a windows program, scan through an impressive amount of online help, and debug the program without leaving Developer Studio.

Visual C++ and Developed Studio makes up a fully integrated environment which makes it easier to create Windows programs. By using tools and wizards provided as a part of Developer Studio, along with the MFC class library you can create a program in just a few minutes.

The programs use thousands of lines of source code that are part of MFC class library, They also take advantage of AppWizard and Class Wizard, two of the Developer Studio tools that manage the project.

4.3.DEVELOPER STUDIO WIZARDS

A Wizard is a tool that helps guide you through a series of steps. In addition to tools that are used for debugging, editing, and creating resources, Developer Studio includes several wizards that are used to simplify developing your windows programs. The most commonly used ones are

1. App Wizard (also referred to in some screens as MFC AppWizard) is used to create the basic outline of a windows program. Three types of programs are supported by AppWizard: Single Document and Multiple Document applications based on the Document/View architecture and dialog box-based programs, in which a dialogue box serves as the application's main window.

2. Class Wizard is used to define the classes in a program created with AppWizard. Using Class Wizard, you can add classes to your project. You can also add functions that control how messages received by each class are handled. Class Wizard also helps manage controls that are contained in dialog boxes by enabling you to associate an MFC object or class member variable with each control.

4.4. MFC LIBRARY:

A library is a collection of source code or compiled code that you can reuse in your programs. Libraries are available from compile vendors such as Microsoft, as well as from third parties.

Visual C++ 6 includes Version 6.0 of MFC, the Microsoft Foundation Classes, a class library that makes programming for Windows much easier. By using the MFC classes when writing your program for Windows, you can take advantage of a large amount of source code that has been written for you, This enables you to concentrate on the important parts of your code rather that worry about the details of Windows Programming.

A recent addition to the C++ standard is the standard C++ Library. This library includes a set of classes that were known as Standard Template Library, which is primarily used for Windows Programming, the standard C++ library is used for general-purpose programming.

4.5. THE VISUAL C++ EDITOR:

Developer Studio includes a sophisticated editor as one of its tools. The editor is integrated with other parts of Developer Studio; files are edited in a Developer Studio; files are edited in a Developer Studio child window. You can use the Developer Studio editor to edit C++ source file that will be compiled into Windows Programs. The editor supplied with Developer Studio is similar to a word processor, but instead of fancy text-formatting features. It has features that make it easy to write source code. You can use almost any editor to write C++ source code, but there are several reasons to consider using the editor integrated with Developer Studio, The editor includes many features that are found in specialized programming editors.

1 Automatic syntax highlighting colors keywords, comments, and other source code in different colors.

2 Automatic "smart" indenting help lines up your code into easy-to-read Columns

3 Emulation for keystrokes used by other editors helps if you are familiar with editors such as Brief and Epsilon.

4 Integrated keyword help enables you to get help on any keyword, MFC class, or Windows function just by pressing F1.

5 Drag-and-drop editing enables you to move text easily by dragging it with the mouse

5. TECHNICAL NOTES

5.1. NETWORK

A Network consists of a group of connected computers sharing resources such as documents and printers. Networks are popular because they can save a great deal of time and money.

5.2. PROTOCOL

Protocol is a set of parameters used by a networked computer to communicate with another networked computer. Most computer communication is any network interaction between two networked computers. This may include: file sharing, application sharing, and printing to a networked printer. The various protocols may be considered as its own unique language. Each computer platform, or operating system, uses its own protocol. In general terms, protocol may be thought of as a conversation between to networked computers. The protocol for both networked computers must be the same protocol, platform and operating system.

5.3. LAN NETWORK

A *LAN* is a high-speed data network that covers a relatively small geographic area. It typically connects workstations, personal computers, printers, servers, and other devices. LANs offer computer users many advantages, including shared access to devices and applications, file exchange between connected users, and communication between users via electronic mail and other applications.

5.4. TCP/IP

TCP/IP is a hybrid of the acronyms of its two most important protocols: Transport Control Protocol and the Internet Protocol. The design of the TCP/IP made it suitable for corporate internetworks and it became more commonly used for networks that weren't necessarily connected to the wider internet. TCP/IP had become a de facto standard for network communications. Every important operating system supports TCP/IP for network communications, and as TCP/IP becomes more important. Many network operating systems are even beginning to drop their reliance on proprietary protocols in favour of standardizing on TCP/IP.

5.5. TCP/IP COMPONENTS

Generally TCP/IP is a suite of protocols and each protocols performs a specific role in the process of communicating data over an Internetwork.The TCP/IP stack consists of four layers :

- Network Access Layer
- Internet Communication Layer
- Host-to-Host Communication layer
- Application Services Layer

NETWORK ACCESS LAYER

The Network Access Layer combines the functionality of the OSI Physical and Data Link Layers.This layer encompasses both the physical communications media and the communications protocols used to transmit frames over those media.

The TCP/IP Network Access Layer can use industry standard protocols such as Ethernet over 10Base-T,but some stack will implement this access in different ways.

INTERNET COMMUNICATION LAYER

The Internet Communication Layer is responsible for providing the capability to communicate between hosts, regardless of underlying Network Access Layer. It includes an addressing and communication protocol that is routable. The Internet Communication Layer uses IP for addressing and transmission of data. In addition , this layer is responsible for providing all the information necessary to the Network Access Layer to send its frameto the local destination. Thus it incorporates the ARP(Address Resolution Protocol) protocol. The Reverse Address Resolution Protocol (RARP).The RARP which is used to provide diskless workstations , also lies in this layer.

This layer includes RIP(Routing Information Protocol), which can interrogate devices on the network to determine how to route packets to a destination. This Layer also includes Internet Control Message Protocol(ICMP) for the hosts to communicate information about problems or failures in the network itself. Finally, the Internet Communication Layer provides multicasting the functionality for sending information to multiple destination hosts once. The Internet Group Management Protocol (IGMP) supports this process.

HOST-TO-HOST COMMUNICATION LAYER

The Host-to-Host Communication Layer handles the services needed to provide reliable communication functionality between hosts, and corresponds roughly to the Transport Layer and part of the Session Layer from OSI model , but it also includes part of the Application and Presentation Layers. This Layer consists of two protocols, the TCP(Transmission Control Protocol) and the UDP(User Datagram Protocol).

TCP provides he capability to establish connection-oriented services between hosts. It includes the features like segmentation of data into packets. Reconstruction of data streams from packets, socket services for providing multiple connection to port s to ports

on remote hosts, Packet verification and error control, packet sequencing and reordering and the flowcontrol.

The UDP was designed to provide a low-network-overhead mechanism for transmitting data over the low layers. UDP was designed to reduce network stack overhead for applications that implement their own versions of the connection-oriented services used by the TCp.In addition to implementing the TCP and UDP protocols this Layer includes the API(Application Programming Interfaces) used to take advantage of them.

APPLICATION SERVICES LAYER

The Application Services Layer is another layer that does not correspond easily to the OSI model. It consists primarily of high-level protocol services designed to take advantage of the low-level protocols such as TCP and UDP. These services take advantage of those lower-level protocols such as TCP and UDP. These services take advantage of those lower-level protocols to provide common Internet services, such as:

Terminal Emulation(Telnet)

File Transfer(FTP,TFTP)

World Wide Web Server (HTTP)

Remote Shell Access(RSH)

5.6. COMMUNICATING WITH TCP/IP

The client application begins by telling the TCP/IP stack to send the updated record to the destinated workstation. Now the data stream is passed to the TCP and it opens and maintains a socket connection. TCP encodes the data stream into a format and then breaks the data stream into TCP packets. These TCP packets are passed to IP, in the Internet Communication Layer. This will encapsulate the TCP packets into IP packets which is ready to send over the network. IP will send each IP packet with appropriate hardware address information to the Network Access Layer and sent over as frames through the physical media.

Now the destinated workstation will simply reverse the process. The Network Access Layer opens each frame and sends to the IP and it will open each IP packet and sends to the TCP. Finally after checking the data stream is passed to the server application.

5.7. SOCKET

A socket is an end point for communications. For network communications to occur through a socket interface the program requires a socket at each end of the network communication. The connection between the socket can be connection oriented or connectionless.

5.8. PORT

A port can refer to a physical connection point for peripheral devices such as serial, parallel, and USB ports. The term port also refers to certain Ethernet connection points, such as those on a hub, switch, or router.

Port numbers most commonly appear in network programming, particularly socket programming. Sometimes, though, port numbers are made visible to the casual user. For example, some Web sites a person visits on the Internet use a URL like the following:

http://www.mairie-metz.fr:8080/

In this example, the number 8080 refers to the port number used by the Web browser to connect to the Web server. Normally, a Web site uses port number 80.

In IP networking, port numbers can theoretically range from 0 to 65535. Most popular network applications, though, use port numbers at the low end of the range (such as 80 for HTTP). The port number is included as a field within the header of each IP packet.

5.9. IP ADDRESSING

In order for a network to be useful hosts must have a way of contacting each other. The TCP/IP protocol utilizes multiple layers of addressing to make this possible. It can be broken into four different layers. The physical or hardware layer lies at the base of this protocol "stack". This layer depends upon the actual physical implementation of the network (such as ethernet or token-ring) and is ultimately responsible for the delivery of network information packets. The IP (Internet Protocol) layer sits above the physical layer in the stack and is independent of the network hardware. Therefore it may encompass many different physical types of networks. The TCP (Transmission Control Protocol) comes next and is independent of both the physical network and the IP layers (although it is most commonly used with IP). The application layer, where a user typically interacts, tops off the stack. The underlying protocols below the application are mostly transparent to the end user. Some applications include FTP (File Transfer Protocol) and Telnet.Each layer in the protocol stack has its own addressing scheme. The hardware address is used to identify hosts connected to the network. Different physical networks have different forms of addresses that are not compatible with one another. The IP address is also used to identify hosts connected to the network, however it does not rely upon the underlying hardware so it can identify hosts on different physical networks. The TCP port number identifies a particular application on a machine. A packet of information leaves the application layer and enters the TCP layer where a TCP header is added containing the destination and source port number. Then it enters the IP layer which adds an IP header containing a destination and source IP address. Finally the hardware layer adds its header and the appropriate address information. When the packet reaches its destination each layer removes and decodes its header information and then sends the remainder of the packet on until it reaches the destination application

6. DETAILS OF THE DESIGN

6.1. SYSTEM DESIGN

System Design is an approach that deals with the information to be fed and the output obtained. This important phase is composed of several steps. It provides the understanding and procedural aspects necessary for implementing the system recommended in the feasibility study.

Emphasis is on translating the performance requirements into design specifications. Design goes through logical and physical stages of development.

System Design is the system solution to a problem that has the same components and interrelationships among the components as the original problem. System Design

- Schedules Design activities
- Works with the user to determine the various data inputs to the system
- Plans how data will flow through the system
- Designs required Outputs
- Program Specification

6.2. INPUT DESIGN

Input Design is the most important part of the overall system design , which requires very careful attention. The collection of input data is the most expensive part of the system. Input Design is concentrated on estimating what the inputs are and how often they are arranged on the input screen, how frequently the datas are collected etc.The input screens for the File Sender and File Find are very Simple and user-friendly.

6.3. OUTPUT DESIGN

Output Design generally refers to the results generated by the system. For many end-users, output is the main reason for developing the system and the basis on which the usefulness of the application is evaluated. The objective of a system finds its shape in terms of the output. The analysis of the objective of a system leads to determination of outputs.

6.4. CODE DESIGN

In this design an object physical characteristics or performance characteristics or operational instructions are specified. This can also show the inter relationship and may sometimes be used to achieve secrecy or confidentiality.

The development methodology is used in the code design. The approach used here is the top-down approach. Here codes are used for finding ,sending receiving and monitoring the files across the network.The speed of the File Transfer depends upon the code written.

6.5. MODULE DESIGN

A module is defined as a collection of program statements with four attributes

INPUT
OUTPUT
FUNCTION
INTERNAL DATA

Modular systems consists of a well-defined , manageable units with well-defined interfaces among the units.

The following are the various Modules that are available in this system:

- File Sender
- File Receiver
- File Find
- File Watch

6.5.1. File Sender

File sender module transfers the files to the destination machine. It involves specification of the file name to be transferred which can also be selected by using the "browse" command button. Then the IP address of the destination machine is specified. The "send" command button is activated only when correct filename and IP address is specified. The status bar gives the exact transfer rate of the file transferred. The Exit command, similar in nature to the stop button found on many Web browsers, allows a user to stop any operation in progress while still maintaining the connection to the destination

workststion. The Status Bar helps to ensure the accurate transfer of data without interruptions.

INPUT SCREEN FOR FILE SENDER:

The input screen for the file transfer consists of specification of the file name which can be browsed using the browse command box.

The user has to input the IP address or the host name of the system in the network to get connected. So for this purpose an user friendly screen is generated listing all the IP addresses of all the machines in the network. There is a status bar which shows the exact rate of file transfer.

6.5.2. File Receiver

File receiver receives the files from the source workstation and stores it in the destined workstation. The File receiver contains the IP address of the destination machine and it also shows the amount of files transferred in terms of KBPS. The receiver can at any time exit the application by making use of the EXIT command. The files are received and stored in the destinstion workstation once the files are transferred using the File Reciever Application.

6.5.3. File Find

File find is used for searching the files through directory trees doing work on each fileFor each directory found it will go into the directory and continue the search.This powerful tool provides search and replace operations across multiple ASCII (text, HTML etc.) files. Actual Search & Replace works under Windows 95/98/Me/2000/NT/XP operation system .Find files by specifying the path, name ,size and text contents.Actual Search & Replace displays abstracts from the searched files.Find files within the previous search results.Once you find the documents, you can replace, insert, and delete text in them.Files can be filtered by their size and last modification date. File find is used to save the report about the found files and detailed information about the occurrences found in them.

6.5.4 File Watch

FileWatch is a utility that can monitor a given file for changes. Monitoring can detect file size changes or simply file writes, both with minimal impact on system resources (no polling is performed). The primary use of this utility is for monitoring changes in the log file of a personal firewall program and being able to spawn a separate application when changes are detected, but the application is applied to any number of other uses. his class helps you to monitor files like in the DevStudio. If the file is modified outside, a message box pops up and lets you choose between ignoring modifications or reloading the data, discarding changes.

If the file is modified, a message is send either to the specified view or to the first view of a specified document. The message handler will call the reload function of the document class. The class should be thread save.

6.6. FUNCTIONAL PROCEDURE

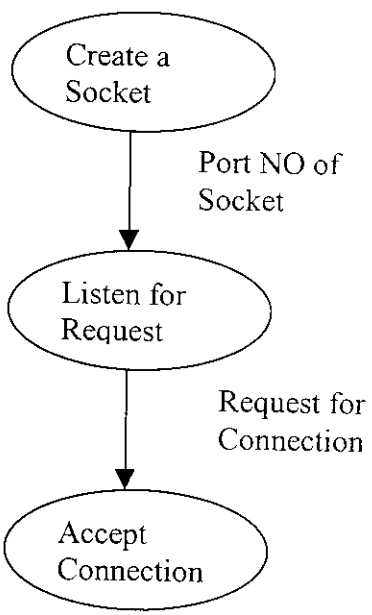
6.6.1. CLIENT COMMUNICATION

Application level	Network PC level
socket( );	socket is called by the program. This will set up a data structure for the operating system to use, to read and write, and control the socket
*bind( );	bind is used to set the address and port number associated with a given socket

connect( );	With TCP this connect call will make the OS preform what is known as "the TCP three part handshake". This consists of the Client computer (the computer that called connect) sending a SYN packet, to the Server computer. If all goes well the server computer will respond with an SYN,ACK packet, and then finally the Client computer sends a final ACK packet. If all this works well connect( ) returns a posetive integer, and you will have a connected socket
send( );, and or recv( );, sendto( );, recvfrom( );	Bind() function is used to bind a ipaddress and port, recv is for reading, and send is for sending.When using TCP it is more common for applications to use send, and recv, allthough sendto, recvfrom, and read and write do work.

close( );, shutdown( )	These functions will close the socket and deallocate any resources from the OS.

PROCESS DIAGRAM



6.6.2 SERVER COMMUNICATION

socket( );

This will set up a data structure for the operating system to use, to read and write, and control the socket

*bind( );

In server applications it you will almost always see a call to bind( ). Bind is used to relate a local port number to a socket, therefore if you wanted to create a server that ran

on a specific port, such as telnet, FTP, and HTTP, you will need to use bind in your sever application.

*listen( );

The call to listen( ) is only needed in TCP applications, this will allow you to specify how many clients can connect to any given TCP port at one time.

*accept( );

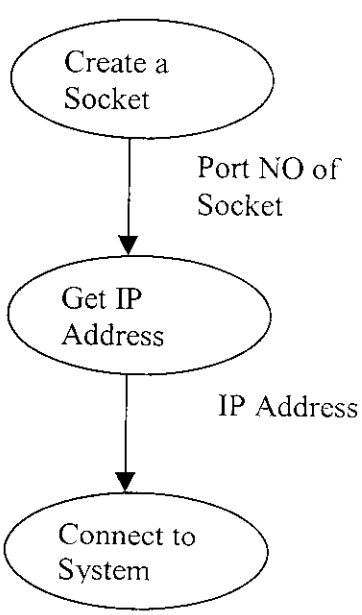
The call to accept( ) again , is only needed in TCP applications, this will allow you to perform the server side of the three way handshake when a client connects. If accept is successful it will return a new socket to you for reading and writing.

send( );, and
or recv( )##;, sendto( );, recvfrom( );
bind () function is used to bind a ipaddress and port, you will most likely want to just use recv for reading, and send for sending.

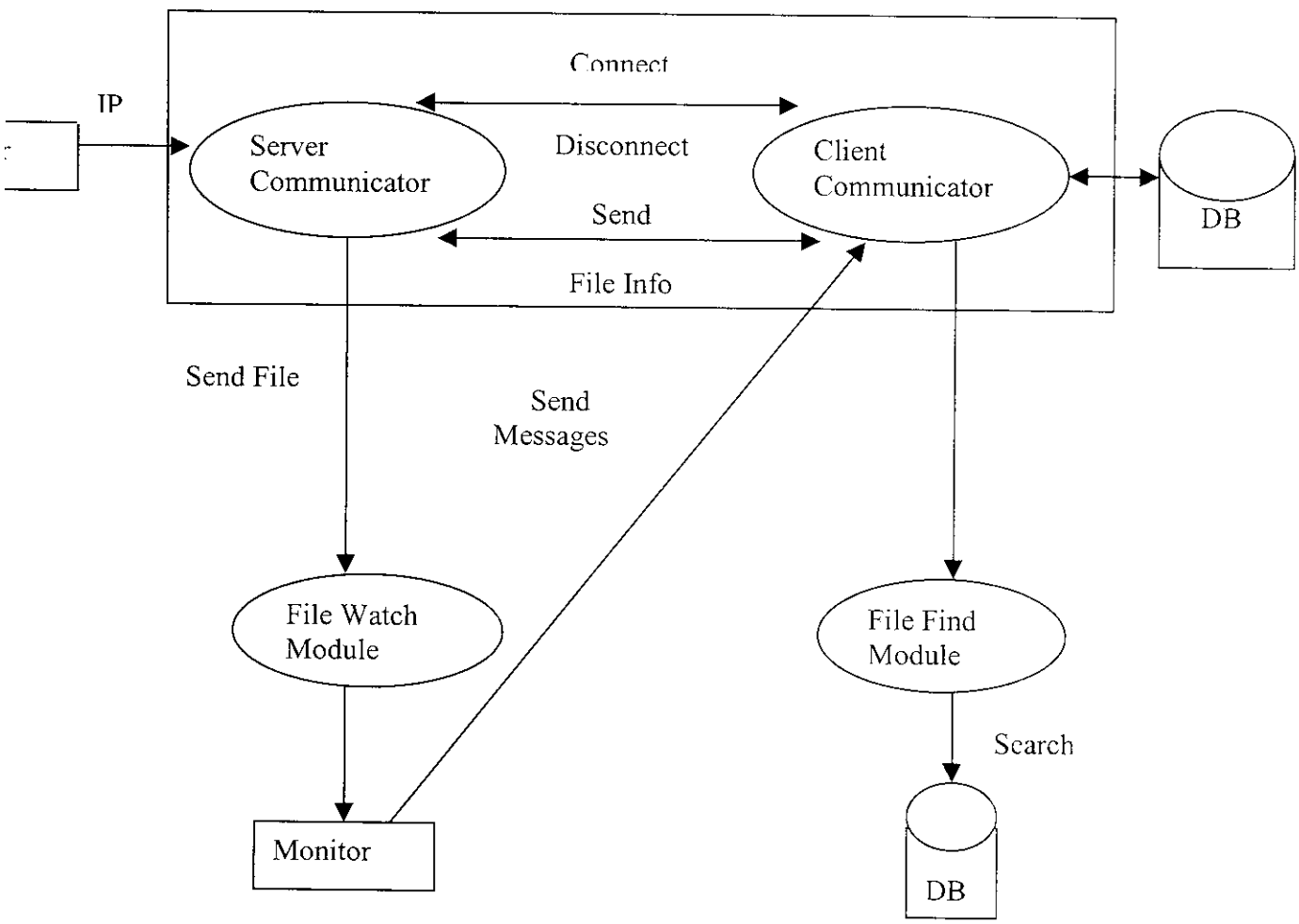
close( );, shutdown( );

These functions will close the socket and deallocate any resources from the OS.

PROCESS DIAGRAM



6.7.OVERALL DIAGRAM



7. IMPLEMENTATION DETAILS

A crucial phase in the system life cycle is the successful implementation of the new system design. Implementation is the stage of project when the theoretical design is turned into a working system. Implementation involves creating computer- compatible files, training the operating staff, and installing hardware, terminals and telecommunications network (where necessary) before the system is up and running. A crucial factor in conversion is not disrupting the functioning of the organization.

In system implementation, user training is crucial for minimizing resistance to change and giving the new system a chance to prove its worth. The training aids include user manuals, help screens, data dictionary, job aids etc..

There are three types of implementation:

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

8. TESTING

8.1. SYSTEM TESTING

Testing is an activity to verify that correct system is being built and is performed with intent of finding faults in the system. Testing is an activity, however not restricted to being performed after the development phase is complete. But this is to be carried out in parallel with all stages of system development, starting with requirements specification. Testing results, once gathered and evaluated, provide a qualitative induction of software quality and reliability and serve as a basis for design modification if required. A project is set to the incomplete without project testing.

System testing is process of checking whether the development system is working according to the original objectives and requirements. The system should be tested experimentally with the test data so as to ensure that the system works according to the required specification. When the system is found working, test it with actual data and check performance.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and attendant "cost" associated with a software failure is motivation forces for a well planned, through testing.

8.2. TESTING OBJECTIVES:

The testing objectives are summarized in the following three steps. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as yet undiscovered. A successful test is one that uncovers as-yet-undiscovered error.

8.3. TESTING PRINCIPLES:

All tests should be traceable to customer requirements. Tests should be planned long before testing begins, that is, the test planning can begin as soon as the requirement model is complete. Testing should be "in the small" and progress towards testing "in large". The focus of testing will shift progressively from progressively from programs, to individual modules and finally to the entire project. Exhaustive testing is not possible. To be more effective, testing should be one, which has highest probability of finding errors.

The following are attributes of good test:

- A good test has a high probability of finding an error
- A good test is not redundant
- A good test should be " best of breed"
- A good test should be neither too simple nor too complex

8.4. LEVELS OF TESTING:

The details of the software functionality tests are given below. The testing procedure that has been used is as follows

- Unit Testing
- Integration testing
- Validation testing
- Output testing

8.4.1. UNIT TESTING:

Unit testing is carried out to verify and uncover errors within the boundary of the smallest unit or a module. In this testing step, each module was found to be working satisfactory as per the expected output of the module. In the package development, each module is tested separately after it has been completed and checked with valid date. Unit testing exercises specific paths in the modules control structure to ensure complete coverage and maximum error detection.

8.4.2. INTEGRATION TESTING:

Integration testing address the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high-order test are conducted. The main objective in this testing process is to take unit tested modules and build a program structure that has been dictated by design.

The following are the types of Integrated Testing:

TOP-DOWN INTEGRATION:

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module. The module subordinates to the main program module are incorporated to the structure in either a depth first or breath-first manner.

BOTTOM-UP INTEGRATION:

This method designs the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

- The low level modules are combined into clusters that perform a specific software sub-function.

- A driver (i.e.) the control program for testing is returned to coordinate test case input and output.

- The cluster is tested.
- Drivers are removed and clusters are combined moving upwards in the program structure.

8.4.3. VALIDATION TESTING

At the end of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and correction testing begins.

VALIDATION TEST CRITERIA:

Software testing and validation is achieved through a series of black box tests that demonstrate conformity with the requirements are achieved, documentation is correct and other requirements are met.

8.4.4. OUTPUT TESTING:

Output testing is a series of different test whose primary purpose is to fully exercise the computer based. Although each test has a different purpose all the work should verify that all system elements have been properly integrated and perform allocated functions.

Output testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operations commence. The input screens, output documents were checked and required modifications made to suit the program specification. Then using the rest data prepared, the whole system was tested and found to be a successful one.

9. CONCLUSION AND FUTURE OUTLOOK

9.1. CONCLUSION

The File Transfer Protocol application developed was successful in operation to perform the sharing of file over the LAN network.It was able to achieve its intended goals.The system was developed with quality consiousness and more importantly it pertained to the standards of the company .The testing was completely successful for all the requirements and specifications. This system is developed with the specifications and abiding by the existing rules and regulations of the company.

Since the requirements of any organizations and their standards are changing day by day the system has been designed in such a way that its scope and boundaries could be expanded in future with little modifications.The main aim behind the implementation of this project is to facilitate easiest way for transferring the files using VC++.

9.2. FUTURE ENHANCEMENT

There are many features that can be added to the system.Due to the insufficient time it could not be incorporated in this system.

The future enhancements that cn be provided are

- Network Security can be provided with Encryption and Decryption Techniques
- Transferring voice over the network can be implemented.
- LAN network can be implemented through internets.

10. REFERENCES

1.David J. Kruglinski, George Shepherd, Scot Wingo,
"Programming Microsoft Visual C++",
Microsoft Press; Fifth Edition, 1992

2.Richard C. Leinecker and Archer Tom,
"Visual C++ 6 Programming Bible",
IDG Books India (P) Ltd; Fifth Edition, 1999

3. Roger S. Pressman
"Software Engineering and Application",
McGraw Hill; Fourth Edition, 1993

4.John Paul Mueller
"Visual C++ 6 from the Ground up",
Tata McGraw Hill, 1998

5.Andrew Tenanbaum
"Computer Networks"
Tata McGraw Hill, 1996

5.MSDN Library

Web-sites:

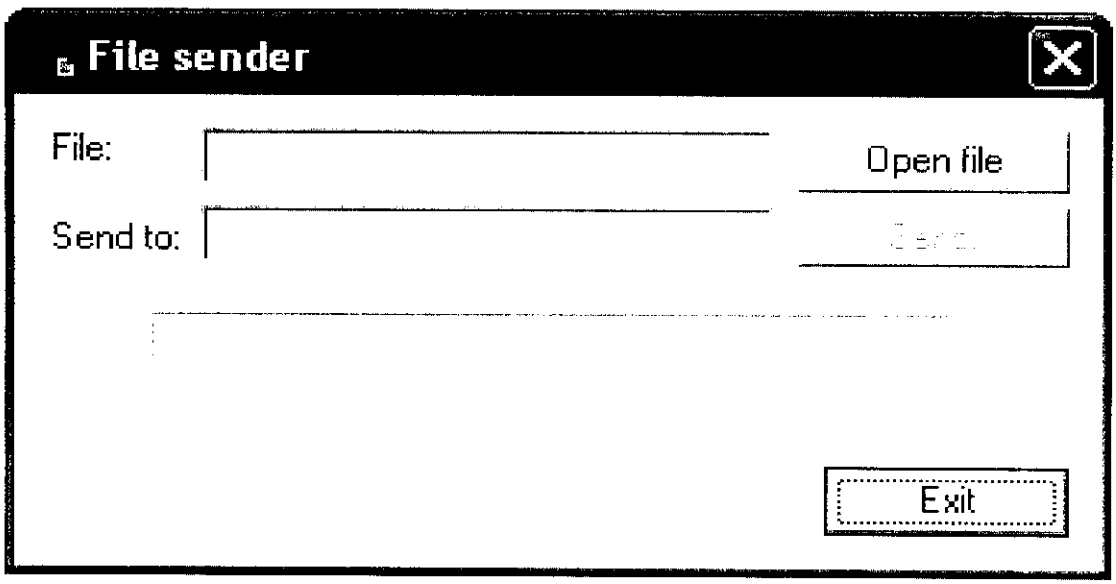
www.microsoft.com

www.netscape.com

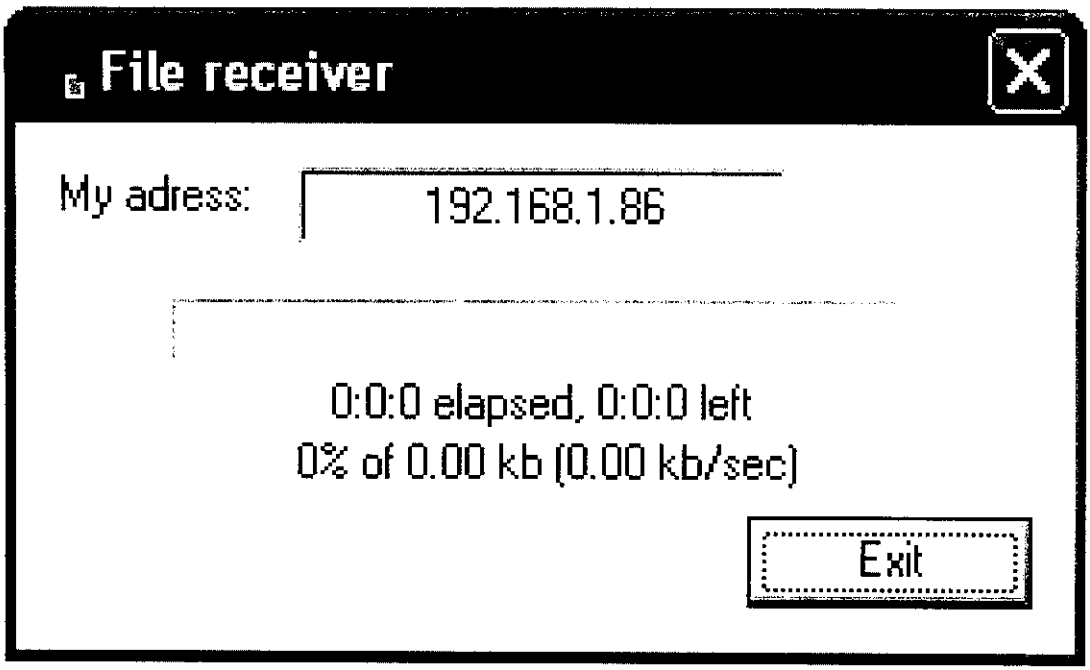
www.MFCnet.com

11. APPENDIX

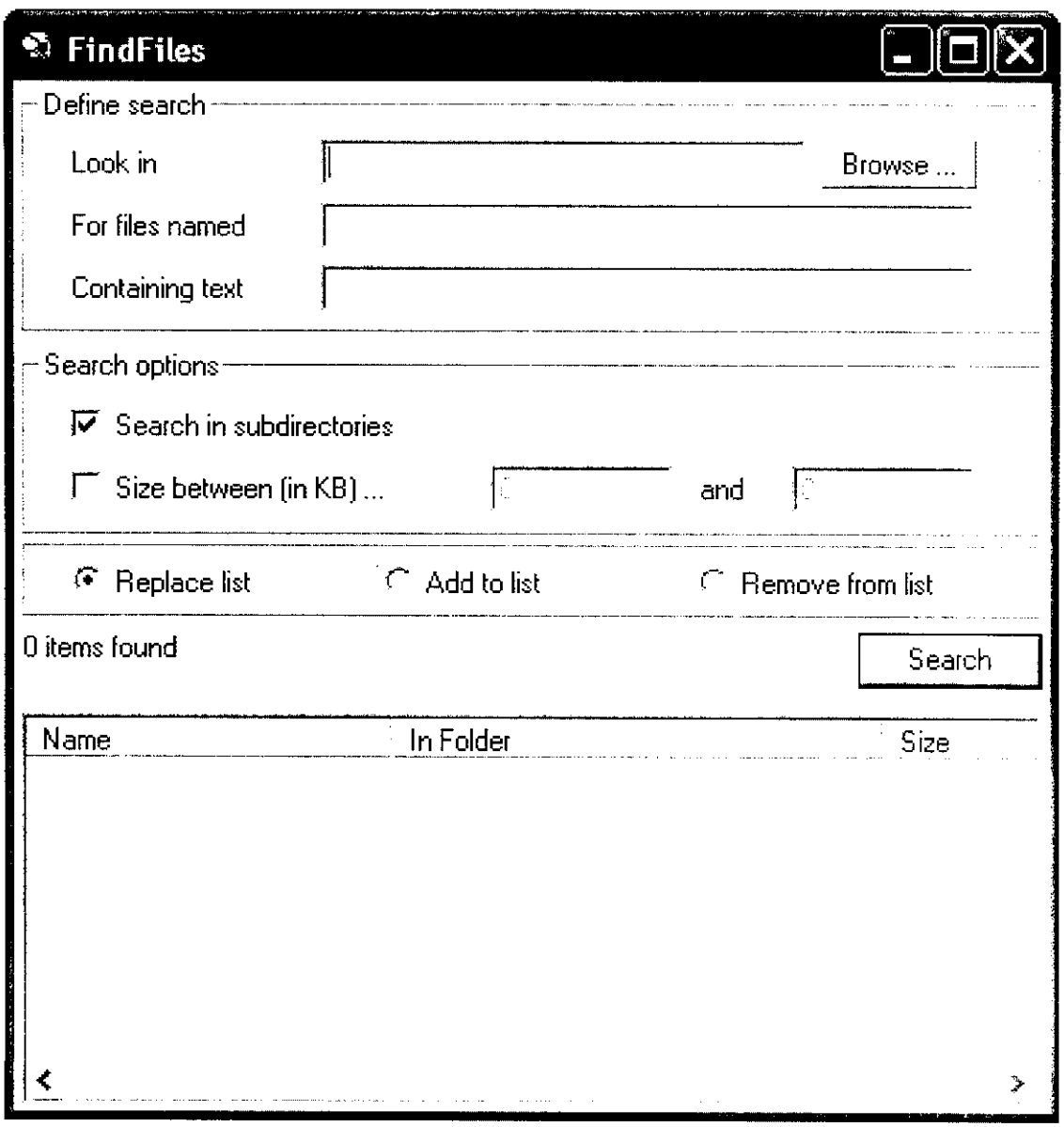
FILE SENDER



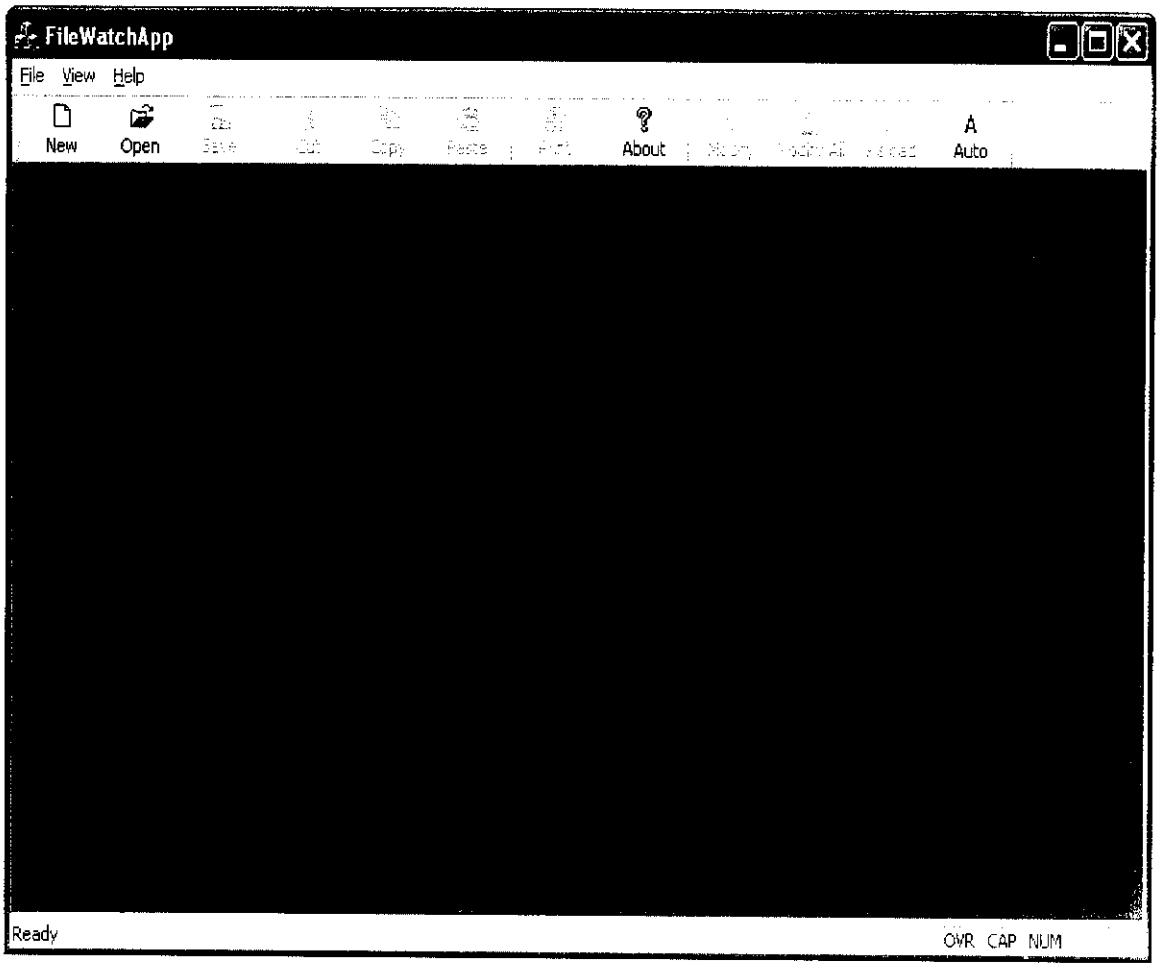
FILE RECEIVER



FILE FIND


FindFiles
Define search
Look in
Browse ...
For files named
Containing text
Search options
Search in subdirectories
Size between (in KB) ...
and
Replace list
Add to list
Remove from list
0 items found
Search
Name
In Folder
Size

FILE WATCH


FileWatchApp
File View Help
New
Open
About
Auto
Ready
OVR CAP NUM

11.2. SAMPLE CODE

FILE SENDER

```
#include "stdafx.h"
#include "file_sender.h"
#include "file_senderDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CFile_senderApp

BEGIN_MESSAGE_MAP(CFile_senderApp, CWinApp)
        //{{AFX_MSG_MAP(CFile_senderApp)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG
        ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFile_senderApp construction

CFile_senderApp::CFile_senderApp()
```

```
{
	// TODO: add construction code here,
	// Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CFile_senderApp object

CFile_senderApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CFile_senderApp initialization

BOOL CFile_senderApp::InitInstance()
{
	if (!AfxSocketInit())
	{
		AfxMessageBox(IDP_SOCKETS_INIT_FAILED);
		return FALSE;
	}

	AfxEnableControlContainer();

	// Standard initialization
	// If you are not using these features and wish to reduce the size
	//  of your final executable, you should remove from the following
	//  the specific initialization routines you do not need.

#ifdef _AFXDLL
	Enable3dControls();			// Call this when using MFC in a shared
DLL
```

```
#else
	Enable3dControlsStatic();	// Call this when linking to MFC statically
#endif

	CFile_senderDlg dlg;
	m_pMainWnd = &dlg;
	int nResponse = dlg.DoModal();
	if (nResponse == IDOK)
	{
		// TODO: Place code here to handle when the dialog is
		//  dismissed with OK
	}
	else if (nResponse == IDCANCEL)
	{
		// TODO: Place code here to handle when the dialog is
		//  dismissed with Cancel
	}
	// Since the dialog has been closed, return FALSE so that we exit the
	//  application, rather than start the application's message pump.
	return FALSE;
}
```

FILE RECEIVER

```
#include "stdafx.h"
#include "file_receiver.h"
#include "file_receiverDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CFile_receiverApp

BEGIN_MESSAGE_MAP(CFile_receiverApp, CWinApp)
        //{{AFX_MSG_MAP(CFile_receiverApp)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG
        ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFile_receiverApp construction

CFile_receiverApp::CFile_receiverApp()
{
        // TODO: add construction code here,
        // Place all significant initialization in InitInstance
}
```

```
/////////////////////////////////////////////////////////////////////////////
// The one and only CFile_receiverApp object

CFile_receiverApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CFile_receiverApp initialization

BOOL CFile_receiverApp::InitInstance()
{
	if (!AfxSocketInit())
	{
		AfxMessageBox(IDP_SOCKETS_INIT_FAILED);
		return FALSE;
	}

	AfxEnableControlContainer();

	// Standard initialization
	// If you are not using these features and wish to reduce the size
	//  of your final executable, you should remove from the following
	//  the specific initialization routines you do not need.

#ifdef _AFXDLL
	Enable3dControls();			// Call this when using MFC in a shared
DLL
#else
	Enable3dControlsStatic();	// Call this when linking to MFC statically
#endif
```

```
	CFile_receiverDlg dlg;
	m_pMainWnd = &dlg;
	int nResponse = dlg.DoModal();
	if (nResponse == IDOK)
	{
		// TODO: Place code here to handle when the dialog is
		//  dismissed with OK
	}
	else if (nResponse == IDCANCEL)
	{
		// TODO: Place code here to handle when the dialog is
		//  dismissed with Cancel
	}

	// Since the dialog has been closed, return FALSE so that we exit the
	//  application, rather than start the application's message pump.
	return FALSE;
}
```

FILE FIND

```
#include "stdafx.h"
#include "FileFinder.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CFileFinder::CFileFinder()
{
        _bStopSearch = false;
        _pFileFinderProc = NULL;
        _pCustomParam = NULL;
}

CFileFinder::~CFileFinder()
{

}

int CFileFinder::FindFiles(LPCTSTR szBaseFolder, LPCTSTR szFileMask, BOOL
bSubFolders)
{
        CFindOpts opts;
```

```
	opts.sBaseFolder = szBaseFolder;
	opts.sFileMask = szFileMask;
	opts.bSubfolders = bSubFolders;

	// Get all files, but no directories
	opts.FindAllFiles();

	Find(opts);

	return GetFileCount();
}

int CFileFinder::Find(CFileFinder::CFindOpts &opts)
{
	CFileFind	finder;
	CString		sFullMask;
	CFindOpts	subOpts;
	BOOL		bFound, bValidFile;
	CTime		timeFile;

	_bStopSearch = false;

	opts.sBaseFolder = CPath::AddBackSlash(opts.sBaseFolder);

	// Find directories
	if (opts.bSubfolders)
	{
		sFullMask = opts.sBaseFolder + CString("*.*");
		bFound = finder.FindFile(sFullMask);
		while ((bFound) && (!_bStopSearch))
```

```
            {
                bFound = finder.FindNextFile();
                if ((finder.IsDirectory()) && (!finder.IsDots()))
                {
                    subOpts = opts;
                    subOpts.sBaseFolder = opts.sBaseFolder +
finder.GetFileName();

                    // Recursive call
                    Find(subOpts);
                }
            }
        }

        finder.Close();
```

FILE WATCH

```
#include "stdafx.h"
#include "FileWatchApp.h"
#include "FileWatch.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif


bool GetLastWriteTime(LPCTSTR lpszFileName, __int64 &ftLastWriteTime)
{
        CFile file;
        if (file.Open(lpszFileName, CFile::shareDenyNone))
        {
                BY_HANDLE_FILE_INFORMATION info;
                if (GetFileInformationByHandle((HANDLE)file.m_hFile, &info))
                {
                        ftLastWriteTime =
(__int64(info.ftLastWriteTime.dwHighDateTime)<<32) +
info.ftLastWriteTime.dwLowDateTime;
                        return true;
                }
        }
        TRACE("GetLastWriteTime failed for %s", lpszFileName);
        return false;
}
```