CROSS PLATFORM NETWORKING
(WINDOWS – LINUX)

P-869

PROJECT WORK DONE AT

KUMARAGURU COLLEGE OF TECHNOLOGY

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR THE AWARD OF
BACHELORS DEGREE IN COMPUTER SCIENCE ENGINEERING
OF BHARATHIAR UNIVERSITY,COIMBATORE

SUBMITTED BY

DINUP.G.MATHEW (9927K0121)
JAYAPRAKASH.R (9927K0125)
VENKATRAMAN.B (9927K0173)

GUIDED BY

Ms.V.VANITHA M.E., LECTURER

Department of Computer Science and Engineering

Department of Computer Science and Engineering
KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE – 641 006
2002-2003

Department of Computer Science and Engineering
KUMARAGURU COLLEGE OF TECHNOLOGY
(AFFILIATED TO BHARATHIAR UNIVERSITY)
COIMBATORE – 641 006

CERTIFICATE

This is to certify that the project report entitled

CROSS PLATFORM NETWORKING
(WINDOWS – LINUX)

is the bonafide work of

DINUP.G.MATHEW	(9927K0121)
JAYAPRAKASH.R	(9927K0125)
VENKATRAMAN.B	(9927K0173)

in partial fulfillment of the requirement for the award of Bachelors Degree in Computer Science and Engineering of Bharathiar University, Coimbatore

[illegible] 13/3/03

Professor and Head
(Dr.S.Thangasamy)

Project Guide
(Ms.V.Vanitha)

Submitted for University Examination held on 18. ~~16~~.03.2003

[illegible]

Internal Examiner

[illegible]

External Examiner

ACKNOWLEDGMENT

We oblige our Principal **Dr. K. K. Padmanabhan, B.Sc.(Engg), M.Tech., Ph.D** for providing the necessary facilities, which enabled us to complete the project.

We wish to make a special acknowledgment and thanks to **Dr. S. Thangaswamy Ph.D.**, Head of the Department of Computer Science and Engineering for the benefit of his wide engineering experience, knowledge of his work and assessment to implement this project.

We would like to thank our guide **Ms. V. Vanitha, M.E.**, Lecturer, Department of Computer Science and Engineering without whose motivation and guidance we would not have been able to embark on a project of this magnitude.

We would also like to thank **Mr. M. Vinod, Mr. S. Karthikeyan and Mr. S. Nandha Kumar** for their immense help and valuable inputs for the successful completion of our project.

Synopsis

SYNOPSIS

When it comes to Operating Systems, Linux & Windows are widely acclaimed as the best. Linux is popular due to its unmatched stability, efficient shell scripting & the fact that it came free. Windows gained importance because of its wonderful GUI capabilities & user interactiveness.

Windows catered to naive users, as it's GUI capabilities ensured easy understanding & working of Windows OS. On the other hand, Linux required the operator to remember the commands of operation. Hence, a windows based operator, who is new to Linux, found it difficult to comprehend the features of the Linux operating system.

Hence, we developed a utility to overcome the above difficulty faced by a Windows based operator. We built a GUI based client software in Windows for an operator to interactively work with a Linux server by executing the necessary shell commands and C programs.

Thus we have Windows based clients & a Linux based server. The tools we used are Visual C++ for the client & C for server side programming.

Table of Contents

TABLE OF CONTENTS

Introduction

1. INTRODUCTION

1.1 Linux Operating System

Linux is a UNIX operating system clone which runs on a variety of platforms, especially personal computers with Intel 80386 or better processors. It supports a wide range of software's from Tex to the X Windows System, to the GNU C/C++ compiler, to TCP/IP. It's a versatile, bonfire implementation of UNIX.

Linux is a complete multitasking,multi-user operating system. As all other versions of UNIX, this means that many users can log in and run programs on the same machine simultaneously. Linux supports various file systems for storing data, like the ext2 file system, which was specifically developed for Linux

Linux may be used for a variety of applications including networking, software development and as an end-user platform. It is again considered to be an excellent, low-cost alternative when you draw a comparison with other expensive operating systems. The Linux kernel is developed to use the protected-mode features of Intel 80386 and better processors.

1.2 Objective

Windows' GUI capabilities ensured easy understanding & working which catered to the naive users. However, the Linux operating system makes it mandatory for the operator to remember the commands of operation. Hence, a Windows based operator, who was new to Linux, found it difficult to comprehend the features of Linux. This gap could not be bridged by existing systems.

Thus the core objective of our project was to develop a graphical user interface for a Windows client to access the Linux server for executing the necessary shell commands. The output of such commands would be displayed in the client window in a simple format. This interface includes directory features and implements the options available for the corresponding shell command. Simple C programs, which include necessary inputs, can also be executed through the GUI without mapping to the TC server.

1.3 Scope

To make a utility for a beginner in Linux to help him fathom the basic Linux shell commands and execute simple C programs containing necessary inputs.

Literature Survey

2. LITERATURE SURVEY

There have been many GUI utilities available which mimic the idea of intra-platform networking. By intra-platform networking we mean that the client and server application will have to be on the same machine. A case in point is a very similar GUI utility available between a Linux client and Linux server.

However, these utilities could not be implemented on other platforms especially Windows, the most popular among them. Since we cannot have a Windows client and Linux server running on the same machine on the same time, we decided to incorporate a network based utility.

The concept for this project has been constructed from the Kaptain project. The Kaptain project is meant for a Linux client-server system and has been implemented using XML. So it could not be executed on a Windows platform.

Proposed Line of Attack

3. PROPOSED LINE OF ATTACK

3.1 Product Specification

This utility includes a graphical user interface for a Windows client to access the Linux server by making use of the shell commands. The output of such commands is displayed in the client window in a simple format. This interface includes directory functions and the execution of C programs which have the necessary options and user-defined inputs.

3.2 Processing Environment

The minimum requirements are based on the available resources at our discretion:

	Hardware	Software
Client	Basic Windows PC with AMD Athlon 1.5 GHz processor, 128 MB RAM and 40 GB hard disk	Visual C++ 6.0
Server	Red Hat Linux	C

3.3 Solution Strategy

The problem was approached in a systematic manner pertaining to the software development cycle. The software development cycle that we propose to implement is the "The Waterfall Model". The description of this model is as follows:

Task Region 1:

➢ Terminology	:	Requirements
➢ Work Product	:	Gathering of the necessary requirements for the product.

Task Region 2:

➢ Terminology	:	Analysis
➢ Work Product	:	Analysis of the product definition, the functions and features that the product has to perform, the processing environment ,software tools and languages are to be determined & understood.

Task Region 3:

➢ Terminology	:	Design Phase
➢ Work Product	:	This phase decides the logic and concepts that

must be implemented in the product. The GUI design is decided in this phase.

Task Region 4:

➢ Terminology	:	Product Implementation
➢ Work Product	:	The coding of the product is performed.

Task Region 5:

➢ Terminology	:	Product Testing
➢ Work Product	:	The product will be tested under various parameters and should be found satisfactory.

3.3.1 Modules

The entire utility is divided into three lucid modules:

- Client Module
- Server Module
- Socket Module

Client Module

The client module is used to allow the operator to make command entries. The client module comprises a GUI with a host of user friendly features. These features ensure that a novice user can comprehend the requirements of the product.

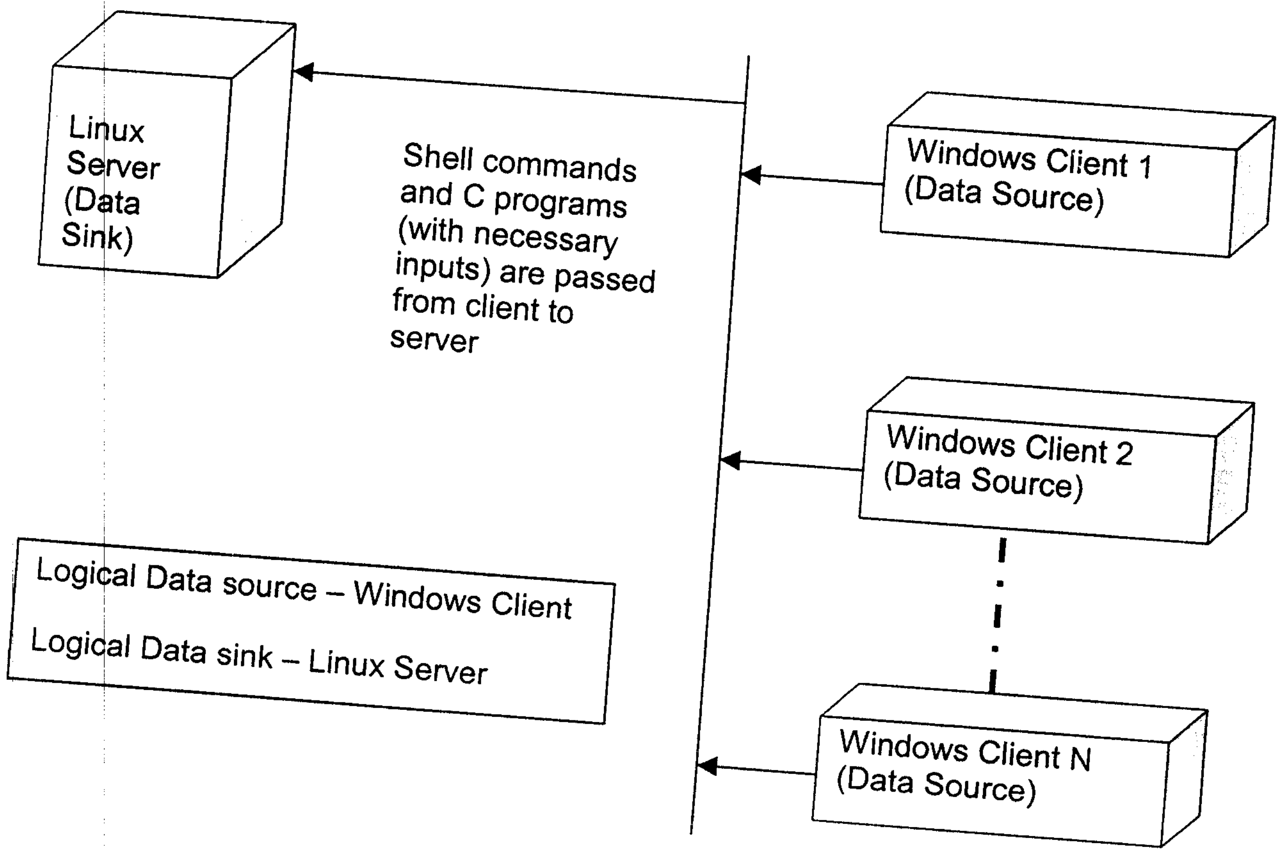
Server Module

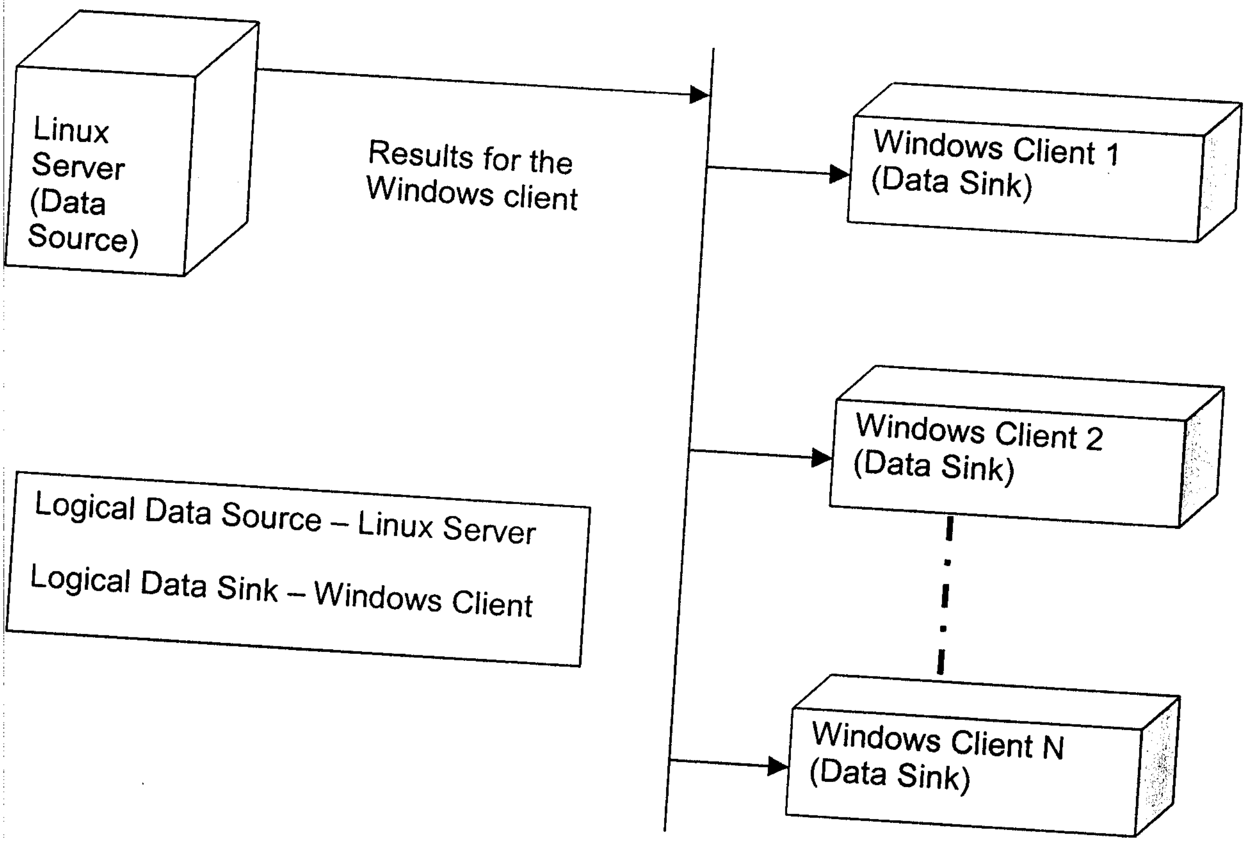
The server module receives the commands from the client side. These are executed on the server and the corresponding response is generated.

Socket Module

This module is employed to establish a connectivity between the client and server. TCP/IP based sockets are used to establish connectivity between the Windows client and the Linux server.

3.3.2 Diagrammatic Representation


Linux Server (Data Sink)
Shell commands and C programs (with necessary inputs) are passed from client to server
Windows Client 1 (Data Source)
Windows Client 2 (Data Source)
Windows Client N (Data Source)
Logical Data source – Windows Client
Logical Data sink – Linux Server


Linux Server (Data Source)
Results for the Windows client
Windows Client 1 (Data Sink)
Windows Client 2 (Data Sink)
Windows Client N (Data Sink)
Logical Data Source – Linux Server
Logical Data Sink – Windows Client

3.3.3 Phases

We propose to attack the problem in a three phased manner.

In the **first phase,** we propose to establish socket connectivity between a Linux client and Linux server, and between a Windows client and Windows server. This phase is done to understand the complexities of socket programming.

In the **second phase,** we propose to establish socket connectivity across platforms – Linux server and Windows client. Here we will be sending a simple packet from the client to the server and get an apposite response from the server based on the data sent.

In the **third phase,** we propose to add the necessary commands and options in the client GUI.

Details of Proposed Methodology

4. DETAILS OF PROPOSED METHODOLOGY

4.1 Module Specification

Shell commands / C programs entry

Scope

This module deals with the entry of shell commands / C programs from the client GUI.

Inputs

The shell command / C program is chosen from the list box of the GUI. Appropriate options are highlighted. The user has to enter the requisite inputs.

Process

The inputs from the user are encapsulated in a socket and sent across to the server.

Outputs

The shell command that is generated is shown as a static text in the GUI.

Shell Command / C program Execution

Scope

This module deals with the execution of shell commands / C programs on the server side.

Inputs

The shell commands / C programs to be executed is obtained from the client side through the socket.

Process

The shell command / C program is executed on the server side.

Outputs

If the command / C program generated from the client is correct, then the corresponding output is displayed on the client GUI. If any error is generated, that will also be displayed in the client GUI.

Connectivity

Scope

This module deals with connecting a Windows client and a Linux server.

Inputs

The shell commands / C programs from the GUI form the input on the client side and the result from the server forms the input on the server side.

Process

Socket connectivity is established using TCP/IP sockets.

Outputs

The shell commands / C programs received by the server side form the output for the client and the result received by the client form the output for the server.

4.2 Product Features

Prototype

A client-server utility that implements two or three simple shell commands.

Current version

The existing system does not incorporate execution of C programs while we have implemented the same, provided the programs have necessary inputs.

The GUI comprises a list box, which contains the **shell commands**, a second list box for directory options and an output box. Based on the command selected, the appropriate options appear as check boxes, edit boxes etc and the user will have to choose the necessary options to get the corresponding output.

The shell command that is executed is displayed as static text in the GUI. This enables the user to comprehend the syntax of the shell command. Any error will be displayed in the output box. All the options for the directory commands are made invisible at this point of time.

For the directory options, we have **make directory** and **remove directory** options. When these are selected, a text box is displayed where in the user will have to type in the directory to be added/deleted. Appropriate error are also displayed. All the options for the shell commands are made invisible at this point of time.

In addition to the above, the GUI comprises a special option for **executing C programs**. Once this particular options gets highlighted, a checkbox is invoked with a text box. The filename is entered in the text box along with the extension(where the extension here is .c). The path is pre-defined to be the desktop. The inputs, if any, are to be entered into a separate text file and saved as input.txt in the desktop. If the C program requires any input, the check box ought to be checked. By default this is unchecked. If there is any sort of compilation error, it will be displayed in the output box. If the compilation is successful, the program is executed on the Linux server and the corresponding output is displayed in the output box.

4.3 Execution

The particular shell command with its options are sent via sockets to the server. At the server side, the first and the second character of the socket data are checked if they are 'e' and 'x' respectively. If they are not, the buffer will contain the shell command to be executed. Using the **system** command, the command is executed.

If the first and second character are 'e' and 'x' respectively, then the remaining socket data consists of a 'C' program and the corresponding input file (if any).The C program is extracted into a file called **abc.c**. It is then compiled. The input is written into a file called **inp**. The program is then executed and the inputs are redirected from the **inp** file.

The output and error (if any) are redirected to two separate files namely **out** and **err** respectively. If the execution has taken place successfully, the **err** file will be empty and the **out** file will contain the output, else the vice-versa. Whichever file that contains data among **err** and **out** is written as socket data and sent across to socket client.

Product Testing

5. PRODUCT TESTING

Machine Configuration

Processor	:	AMD Athlon 1.5 GHz
Hard Disk	:	40GB
RAM	:	128MB
Operating System	:	Windows XP

Type of test - Functional Testing

Requirements Being Tested

An invalid file name should be indicated to the user

Exact test Stimuli

An invalid file name is entered

Expected Outcome

An error message indicating that the filename does not exist is displayed.

Type of test - Functional Testing

Requirements Being Tested

An existing directory name should not be added again

Exact test Stimuli

An existing directory name is entered and the add directory operation is performed

Expected Outcome

An error message indicating that the directory already exists is displayed.

Type of test - Functional Testing

Requirements Being Tested

A non - existing directory cannot be deleted

Exact test Stimuli

A non - existing directory name is entered and the remove directory operation is performed

Expected Outcome

An error message indicating that the directory does not exist is displayed.

Type of test - Functional Testing

Requirements Being Tested

The output cannot be obtained if the C program has errors

Exact test Stimuli

A C program with errors is entered and needs to be executed

Expected Outcome

The errors in the C program will be displayed to the user

Future Enhancements

6. FUTURE ENHANCEMENTS

The product that we have developed is capable of working within the root directory. Furthermore directory changes are not acceptable. The C programs cannot accept command line arguments. Also if any user defined header files are added in the program, it cannot be sent across to the server.

The future enhancements would be to overcome the above mentioned shortcomings and to develop the product to the extent that a user need never login to the Linux server to perform server side operations.

Conclusion

7. CONCLUSION

The project was developed for a Windows client and Linux server over a network. It was put to test and was found to satisfy the requirements specified. Given the time constraints, not all of the shell commands were incorporated. However, the necessary commands and C programs with necessary inputs, have been implemented.

During this course of this project, the nuances of inter platform networking were learnt. The project was challenging and demanding, but the experience we gained from having done it is worth all the hardships we faced.

Bibliography

8. <u>BIBLIOGRAPHY</u>

1. Brian.W.Kernighan and Rob Pike , " **The Unix Programming Environment** ",
   Prentice Hall of INDIA, New Delhi, 2001

2. W. Richard Stevens , " **Unix Networking Programming Volume I** ",
   Pearson Education, 1999

3. Douglus.E.Commer , " **Inter Networking with TCP/IP Volume I** " ,
   Prentice Hall of INDIA, New Delhi, 1997

Appendix

APPENDIX – A

Sample Coding

CheckDlg.cpp (Windows Client Program)

```
// checkDlg.cpp : implementation file
//

#include "stdafx.h"
#include "check.h"
#include "checkDlg.h"
#include <iostream>     // For cout, cerr
#include <afxsock.h>    // For CSocket

using namespace std;

const RECVBUFSIZE = 10000; // Size of receive buffer

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

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
        virtual void DoDataExchange(CDataExchange* pDX);        // DDX/DDV
    support
        //}}AFX_VIRTUAL

    // Implementation
    protected:
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
    };

    CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
    {
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
    }

    void CAboutDlg::DoDataExchange(CDataExchange* pDX)
    {
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
    }

    BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
            // No message handlers
        //}}AFX_MSG_MAP
    END_MESSAGE_MAP()

    /////////////////////////////////////////////////////////////////////////////
    // CCheckDlg dialog

    CCheckDlg::CCheckDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CCheckDlg::IDD, pParent)
    {
        //{{AFX_DATA_INIT(CCheckDlg)
        m_strMsg = _T("");
        m_strMsg1 = _T("");
        m_strDir = _T("");
        //}}AFX_DATA_INIT
        // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
        m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
    }

void CCheckDlg::DoDataExchange(CDataExchange* pDX)
```

```
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CCheckDlg)
        DDX_Control(pDX, IDC_LIST3, m_ctlDir);
        DDX_Control(pDX, IDC_CHECK4, m_optRev);
        DDX_Control(pDX, IDC_CHECK3, m_optChar);
        DDX_Control(pDX, IDC_CHECK2, m_optWord);
        DDX_Control(pDX, IDC_CHECK1, m_optLine);
        DDX_Control(pDX, IDC_LIST2, m_ctlListSel);
        DDX_Control(pDX, IDC_LIST1, m_ctlList);
        DDX_Text(pDX, IDC_EDIT1, m_strMsg);
        DDX_Text(pDX, IDC_EDIT2, m_strMsg1);
        DDX_Text(pDX, IDC_EDIT3, m_strDir);
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CCheckDlg, CDialog)
        //{{AFX_MSG_MAP(CCheckDlg)
        ON_WM_SYSCOMMAND()
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_EN_CHANGE(IDC_EDIT1, OnChangeEdit1)
        ON_LBN_SELCHANGE(IDC_LIST2, OnSelchangeList2)
        ON_LBN_SELCHANGE(IDC_LIST3, OnSelchangeList3)
        ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
        ON_LBN_SETFOCUS(IDC_LIST3, OnSetfocusList3)
        ON_LBN_SETFOCUS(IDC_LIST2, OnSetfocusList2)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCheckDlg message handlers

BOOL CCheckDlg::OnInitDialog()
{
        CDialog::OnInitDialog();

        // Add "About..." menu item to system menu.

        // IDM_ABOUTBOX must be in the system command range.
        ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
        ASSERT(IDM_ABOUTBOX < 0xF000);

        CMenu* pSysMenu = GetSystemMenu(FALSE);
        if (pSysMenu != NULL)
        {
                CString strAboutMenu;
                strAboutMenu.LoadString(IDS_ABOUTBOX);
                if (!strAboutMenu.IsEmpty())
                {
                        pSysMenu->AppendMenu(MF_SEPARATOR);
                        pSysMenu->AppendMenu(MF_STRING,  IDM_ABOUTBOX,
    strAboutMenu);
                }
        }

        // Set the icon for this dialog.  The framework does this automatically
        //  when the application's main window is not a dialog
        SetIcon(m_hIcon, TRUE);                         // Set big icon
        SetIcon(m_hIcon, FALSE);            // Set small icon

        // TODO: Add extra initialization here
        m_ctlDir.AddString("Add directory(mkdir)");
        m_ctlDir.AddString("Remove directory(rmdir)");



        m_ctlListSel.AddString("Search for string within file(grep)");
        m_ctlListSel.AddString("Word count of a file(wc)");
        m_ctlListSel.AddString("List all users logged in(who)");
        m_ctlListSel.AddString("Display current user(whoami)");
        m_ctlListSel.AddString("Display contents of a file(cat)");
        m_ctlListSel.AddString("Display the last n lines of a file(tail)");
        m_ctlListSel.AddString("List names of all files(ls)");
        m_ctlListSel.AddString("Copy a file(cp)");
        m_ctlListSel.AddString("Rename a file(mv)");
        m_ctlListSel.AddString("To echo whatever you have typed(echo)");
        m_ctlListSel.AddString("Date(date)");
        m_ctlListSel.AddString("To know your terminal(tty)");
        m_ctlListSel.AddString("Displaying current settings of terminal(stty)");
        m_ctlListSel.AddString("Exec");
        return TRUE;  // return TRUE  unless you set the focus to a control
}

void CCheckDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
        if ((nID & 0xFFF0) == IDM_ABOUTBOX)
        {
                CAboutDlg dlgAbout;
                dlgAbout.DoModal();
        }
        else
```

```
          {
                    CDialog::OnSysCommand(nID, lParam);
          }
  }

  // If you add a minimize button to your dialog, you will need the code below
  //  to draw the icon.  For MFC applications using the document/view model,
  //  this is automatically done for you by the framework.

  void CCheckDlg::OnPaint()
  {
          if (IsIconic())
          {
                    CPaintDC dc(this); // device context for painting

                    SendMessage(WM_ICONERASEBKGND,                        (WPARAM)
  dc.GetSafeHdc(), 0);

                    // Center icon in client rectangle
                    int cxIcon = GetSystemMetrics(SM_CXICON);
                    int cyIcon = GetSystemMetrics(SM_CYICON);
                    CRect rect;
                    GetClientRect(&rect);
                    int x = (rect.Width() - cxIcon + 1) / 2;
                    int y = (rect.Height() - cyIcon + 1) / 2;

                    // Draw the icon
                    dc.DrawIcon(x, y, m_hIcon);
          }
          else
          {
                    CDialog::OnPaint();
          }
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CCheckDlg::OnQueryDragIcon()
{
          return (HCURSOR) m_hIcon;
}

void CCheckDlg::OnOK()
{
```

```
AfxSocketInit(NULL);
LPTSTR lpsz = new TCHAR[1000];
CString l;
CFile m_file,m_file1;

char filebuf[RECVBUFSIZE],inbuf[50];
for(int g=0;g<RECVBUFSIZE;g++)
        filebuf[g]=0;
for(g=0;g<50;g++)
        inbuf[g]=0;


char* servIP = "90.0.0.103";
switch(m_ctlListSel.GetCurSel())
{
case 0:
        _tcscpy(lpsz,"grep ");
        if(m_optLine.GetCheck())
                _tcscat(lpsz," -v ");

        GetDlgItemText(IDC_EDIT2,l);
        _tcscat(lpsz,l);
        _tcscat(lpsz," ");
        _tcscat(lpsz,m_strMsg);
        SetDlgItemText(IDC_COMMAND,lpsz);
        _tcscat(lpsz,">out 2>err");
        break;
case 1:
        _tcscpy(lpsz,"wc ");
        _tcscat(lpsz,m_strMsg);
        if(m_optLine.GetCheck())
                _tcscat(lpsz," -l");
        if(m_optWord.GetCheck())
                _tcscat(lpsz," -w");
        if(m_optChar.GetCheck())
                _tcscat(lpsz," -c");
        SetDlgItemText(IDC_COMMAND,lpsz);
        _tcscat(lpsz,">out 2>err");
        break;

case 2:
        _tcscpy(lpsz,"who");
        if(m_optLine.GetCheck())
                _tcscat(lpsz," -Hu");
        SetDlgItemText(IDC_COMMAND,lpsz);
        _tcscat(lpsz,">out 2>err");
```

```
            break;
    case 3:
            SetDlgItemText(IDC_COMMAND,"whoami");
            _tcscpy(lpsz,"whoami>out 2>err");
            break;
    case 4:
            _tcscpy(lpsz,"cat ");
            _tcscat(lpsz,m_strMsg);
            SetDlgItemText(IDC_COMMAND,lpsz);
            _tcscat(lpsz,">out 2>err");
            break;

   case 5:
                 _tcscpy(lpsz,"tail ");
                 GetDlgItemText(IDC_EDIT2,l);
            _tcscat(lpsz,l);
                 _tcscat(lpsz," ");
                 _tcscat(lpsz,m_strMsg);
                 SetDlgItemText(IDC_COMMAND,lpsz);
                 _tcscat(lpsz,">out 2>err");
                 break;
   case 6:
           _tcscpy(lpsz,"ls ");
                 if(m_optLine.GetCheck())
                   _tcscat(lpsz," -l");
          if(m_optWord.GetCheck())
                   _tcscat(lpsz," -t");
          if(m_optChar.GetCheck())
                   _tcscat(lpsz," -u");
          if(m_optRev.GetCheck())
                  _tcscat(lpsz," -l|grep ^d|cut -c57-80");
          SetDlgItemText(IDC_COMMAND,lpsz);
          _tcscat(lpsz,">out 2>err");
          break;
  case 7:
               _tcscpy(lpsz,"cp ");
               _tcscat(lpsz,m_strMsg);
               _tcscat(lpsz," ");
               GetDlgItemText(IDC_EDIT2,l);
          _tcscat(lpsz,l);
               SetDlgItemText(IDC_COMMAND,lpsz);
               _tcscat(lpsz,">out 2>err");
               break;
 case 8:
        _tcscpy(lpsz,"mv ");
        _tcscat(lpsz,m_strMsg);
```

```
              _tcscat(lpsz," ");
                    GetDlgItemText(IDC_EDIT2,l);
               _tcscat(lpsz,l);
                    SetDlgItemText(IDC_COMMAND,lpsz);
                    _tcscat(lpsz,">out 2>err");
                    break;
      case 9:
             _tcscpy(lpsz,"echo ");
             GetDlgItemText(IDC_EDIT2,l);
             _tcscat(lpsz,l);
             SetDlgItemText(IDC_COMMAND,lpsz);
             _tcscat(lpsz,">out 2>err");
             break;
     case 10:
             _tcscpy(lpsz,"date ");
             if(m_optLine.GetCheck())
                    _tcscat(lpsz,"+\"%d(%h)\"");
             if(m_optWord.GetCheck())
                    _tcscat(lpsz,"+%y");
             if(m_optChar.GetCheck())
                    _tcscat(lpsz,"+\"%H(%M)\"");
             if(m_optRev.GetCheck())
                _tcscat(lpsz,"+%S");
             SetDlgItemText(IDC_COMMAND,lpsz);
             _tcscat(lpsz,">out 2>err");
             break;
    case 11:
                  SetDlgItemText(IDC_COMMAND,"tty");
            _tcscpy(lpsz,"tty>out 2>err");
            break;
    case 12:
            _tcscpy(lpsz,"stty ");
                  if(m_optLine.GetCheck())
                    _tcscat(lpsz," -a");
                  SetDlgItemText(IDC_COMMAND,lpsz);
            _tcscat(lpsz,">out 2>err");
            break;
   case 13:
           _tcscpy(lpsz,"C:\\Documents and settings\\99cse09\\desktop\\");
           _tcscat(lpsz,m_strMsg);
           m_file.Open(lpsz,0,0);
           m_file.Read(filebuf,RECVBUFSIZE);

           _tcscpy(lpsz,"ex");
           _tcscat(lpsz,filebuf);
```

```
            if(m_optLine.GetCheck())
            {
                                                        m_file1.Open("C:\\Documentsand
   settings\\99cse09\\desktop\\input.txt",0,0);
                        m_file1.Read(inbuf,50);
                                _tcscat(lpsz,"@");
                                _tcscat(lpsz,inbuf);
            }

            break;
    }

   char* echoString = lpsz;

   UINT echoServPort = 1030;

   CSocket echoClient;
  echoClient.Create();


  echoClient.Connect((LPCSTR)servIP, echoServPort);

  int echoStringLen = strlen(echoString);

 echoClient.Send(echoString, echoStringLen, 0);


 char echoBuffer[RECVBUFSIZE],outbuf[RECVBUFSIZE];

  for(int i=0;i<RECVBUFSIZE;i++)
  {
          echoBuffer[i]=0;
          outbuf[i]=0;
  }

 echoClient.Receive(echoBuffer, RECVBUFSIZE - 1, 0);


 m_ctlList.ResetContent();

 i=0;
 int j=0;
 while(i<strlen(echoBuffer))
 {
         if(echoBuffer[i]!='\n')
                 outbuf[j]=echoBuffer[i];
            else
            {
                    m_ctlList.AddString(outbuf);
                    for(int k=0;k<RECVBUFSIZE;k++)
                            outbuf[k]=0;
                    j=-1;
            }
            i++;
            j++;


    }

   echoClient.Close();




  }

  void CCheckDlg::OnChangeEdit1()
  {

          UpdateData(TRUE);
  }

  void CCheckDlg::OnSelchangeList2()
  {
          SetDlgItemText(IDC_COMMAND,"");
          switch(m_ctlListSel.GetCurSel())
          {
          case 0:
                  GetDlgItem(IDC_EDIT1)->ShowWindow(TRUE);
                  GetDlgItem(IDC_EDIT2)->ShowWindow(TRUE);
                  GetDlgItem(IDC_CHECK1)->ShowWindow(TRUE);
                  GetDlgItem(IDC_CHECK2)->ShowWindow(FALSE);
                  GetDlgItem(IDC_CHECK3)->ShowWindow(FALSE);
                  GetDlgItem(IDC_CHECK4)->ShowWindow(FALSE);
                  GetDlgItem(IDC_GROUP)->ShowWindow(TRUE);
                  GetDlgItem(IDC_FILENAME)->ShowWindow(TRUE);
                  GetDlgItem(IDC_STRING)->ShowWindow(TRUE);
                  SetDlgItemText(IDC_CHECK1,"Display    contents    not    having
string");

                  SetDlgItemText(IDC_GROUP,"Options for Grep");
                  SetDlgItemText(IDC_EDIT1,"");
            SetDlgItemText(IDC_EDIT2,"");
            m_ctlList.ResetContent();
            m_optLine.SetCheck(0);
            m_optChar.SetCheck(0);
            m_optWord.SetCheck(0);
            break;

        case 1:
            GetDlgItem(IDC_EDIT1)->ShowWindow(TRUE);
            GetDlgItem(IDC_EDIT2)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK1)->ShowWindow(TRUE);
            GetDlgItem(IDC_CHECK2)->ShowWindow(TRUE);
            GetDlgItem(IDC_CHECK3)->ShowWindow(TRUE);
            GetDlgItem(IDC_CHECK4)->ShowWindow(FALSE);
            GetDlgItem(IDC_GROUP)->ShowWindow(TRUE);
            GetDlgItem(IDC_FILENAME)->ShowWindow(TRUE);
            GetDlgItem(IDC_STRING)->ShowWindow(FALSE);
            SetDlgItemText(IDC_CHECK1,"Lines");
            SetDlgItemText(IDC_CHECK2,"Words");
            SetDlgItemText(IDC_CHECK3,"Characters");
            SetDlgItemText(IDC_GROUP,"Options for Word Count");
            m_ctlList.ResetContent();
            SetDlgItemText(IDC_EDIT1,"");
            m_optLine.SetCheck(0);
            break;
        case 2:
            GetDlgItem(IDC_EDIT1)->ShowWindow(FALSE);
            GetDlgItem(IDC_EDIT2)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK1)->ShowWindow(TRUE);
            GetDlgItem(IDC_CHECK2)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK3)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK4)->ShowWindow(FALSE);
            GetDlgItem(IDC_GROUP)->ShowWindow(TRUE);
            GetDlgItem(IDC_FILENAME)->ShowWindow(FALSE);
            GetDlgItem(IDC_STRING)->ShowWindow(FALSE);
            SetDlgItemText(IDC_CHECK1,"Header Option");
            SetDlgItemText(IDC_GROUP,"Options for users who have logged
in");

            m_ctlList.ResetContent();
            m_optLine.SetCheck(0);
            break;

        case 3:
            GetDlgItem(IDC_EDIT1)->ShowWindow(FALSE);
            GetDlgItem(IDC_EDIT2)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK1)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK2)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK3)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK4)->ShowWindow(FALSE);
            GetDlgItem(IDC_GROUP)->ShowWindow(FALSE);
            GetDlgItem(IDC_FILENAME)->ShowWindow(FALSE);
            GetDlgItem(IDC_STRING)->ShowWindow(FALSE);
            m_ctlList.ResetContent();
            break;
    case 4:
            GetDlgItem(IDC_EDIT1)->ShowWindow(TRUE);
            GetDlgItem(IDC_EDIT2)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK1)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK2)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK3)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK4)->ShowWindow(FALSE);
            GetDlgItem(IDC_GROUP)->ShowWindow(TRUE);
            GetDlgItem(IDC_FILENAME)->ShowWindow(TRUE);
            GetDlgItem(IDC_STRING)->ShowWindow(FALSE);
            SetDlgItemText(IDC_GROUP,"Options to display contents");
            SetDlgItemText(IDC_EDIT1,"");
            m_ctlList.ResetContent();
            break;
    case 5:
            GetDlgItem(IDC_EDIT1)->ShowWindow(TRUE);
            GetDlgItem(IDC_EDIT2)->ShowWindow(TRUE);
            GetDlgItem(IDC_CHECK1)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK2)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK3)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK4)->ShowWindow(FALSE);
            GetDlgItem(IDC_GROUP)->ShowWindow(TRUE);
            GetDlgItem(IDC_FILENAME)->ShowWindow(TRUE);
            GetDlgItem(IDC_STRING)->ShowWindow(TRUE);
            SetDlgItemText(IDC_GROUP,"Options for Tail");
            m_ctlList.ResetContent();
            SetDlgItemText(IDC_EDIT1,"");
            SetDlgItemText(IDC_EDIT2,"");
            SetDlgItemText(IDC_STRING,"Number of lines");
            break;
    case 6:
            GetDlgItem(IDC_EDIT1)->ShowWindow(FALSE);
            GetDlgItem(IDC_EDIT2)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK1)->ShowWindow(TRUE);
            GetDlgItem(IDC_CHECK2)->ShowWindow(TRUE);
            GetDlgItem(IDC_CHECK3)->ShowWindow(TRUE);
            GetDlgItem(IDC_CHECK4)->ShowWindow(TRUE);
            GetDlgItem(IDC_GROUP)->ShowWindow(TRUE);
            GetDlgItem(IDC_FILENAME)->ShowWindow(FALSE);
            GetDlgItem(IDC_STRING)->ShowWindow(FALSE);
            SetDlgItemText(IDC_CHECK1,"Long Listing");
            SetDlgItemText(IDC_CHECK2,"Time of Creation");
            SetDlgItemText(IDC_CHECK3,"Most recently used first");
            SetDlgItemText(IDC_CHECK4,"Directories only");
            SetDlgItemText(IDC_GROUP,"Options for Listing files");
            m_ctlList.ResetContent();
            m_optLine.SetCheck(0);
            m_optChar.SetCheck(0);
            m_optWord.SetCheck(0);
            m_optRev.SetCheck(0);
            break;
    case 7:
            GetDlgItem(IDC_EDIT1)->ShowWindow(TRUE);
            GetDlgItem(IDC_EDIT2)->ShowWindow(TRUE);
            GetDlgItem(IDC_CHECK1)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK2)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK3)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK4)->ShowWindow(FALSE);
            GetDlgItem(IDC_GROUP)->ShowWindow(TRUE);
            GetDlgItem(IDC_FILENAME)->ShowWindow(TRUE);
            GetDlgItem(IDC_STRING)->ShowWindow(TRUE);
            SetDlgItemText(IDC_GROUP,"Options for Copy");
            m_ctlList.ResetContent();
            SetDlgItemText(IDC_EDIT1,"");
            SetDlgItemText(IDC_EDIT2,"");
            SetDlgItemText(IDC_STRING,"Copy file name");
            break;
    case 8:
            GetDlgItem(IDC_EDIT1)->ShowWindow(TRUE);
            GetDlgItem(IDC_EDIT2)->ShowWindow(TRUE);
            GetDlgItem(IDC_CHECK1)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK2)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK3)->ShowWindow(FALSE);
            GetDlgItem(IDC_CHECK4)->ShowWindow(FALSE);
            GetDlgItem(IDC_GROUP)->ShowWindow(TRUE);
            GetDlgItem(IDC_FILENAME)->ShowWindow(TRUE);
            GetDlgItem(IDC_STRING)->ShowWindow(TRUE);
            SetDlgItemText(IDC_GROUP,"Options for Rename");
            m_ctlList.ResetContent();
            SetDlgItemText(IDC_EDIT1,"");
            SetDlgItemText(IDC_EDIT2,"");
            SetDlgItemText(IDC_STRING,"Enter new filename");
            break;
    case 9:
```

```
                GetDlgItem(IDC_EDIT1)->ShowWindow(FALSE);
                GetDlgItem(IDC_EDIT2)->ShowWindow(TRUE);
                GetDlgItem(IDC_CHECK1)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK2)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK3)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK4)->ShowWindow(FALSE);
                GetDlgItem(IDC_GROUP)->ShowWindow(TRUE);
                GetDlgItem(IDC_FILENAME)->ShowWindow(FALSE);
                GetDlgItem(IDC_STRING)->ShowWindow(TRUE);
                SetDlgItemText(IDC_EDIT1,"");
                SetDlgItemText(IDC_STRING,"Enter the string");
                SetDlgItemText(IDC_GROUP,"Options to echo");
                m_ctlList.ResetContent();
                break;
        case 10:
                GetDlgItem(IDC_EDIT1)->ShowWindow(FALSE);
                GetDlgItem(IDC_EDIT2)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK1)->ShowWindow(TRUE);
                GetDlgItem(IDC_CHECK2)->ShowWindow(TRUE);
                GetDlgItem(IDC_CHECK3)->ShowWindow(TRUE);
                GetDlgItem(IDC_CHECK4)->ShowWindow(TRUE);
                GetDlgItem(IDC_GROUP)->ShowWindow(TRUE);
                GetDlgItem(IDC_FILENAME)->ShowWindow(FALSE);
                GetDlgItem(IDC_STRING)->ShowWindow(FALSE);
                SetDlgItemText(IDC_CHECK1,"Day and Month");
                SetDlgItemText(IDC_CHECK2,"Last two digits of year");
                SetDlgItemText(IDC_CHECK3,"Hour and Minute");
                SetDlgItemText(IDC_CHECK4,"Second");
                SetDlgItemText(IDC_GROUP,"Options for Date");
                m_ctlList.ResetContent();
                m_optLine.SetCheck(0);
                m_optChar.SetCheck(0);
                m_optWord.SetCheck(0);
                m_optRev.SetCheck(0);
                break;
        case 11:
                GetDlgItem(IDC_EDIT1)->ShowWindow(FALSE);
                GetDlgItem(IDC_EDIT2)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK1)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK2)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK3)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK4)->ShowWindow(FALSE);
                GetDlgItem(IDC_GROUP)->ShowWindow(FALSE);
                GetDlgItem(IDC_FILENAME)->ShowWindow(FALSE);
                GetDlgItem(IDC_STRING)->ShowWindow(FALSE);
                m_ctlList.ResetContent();
                break;
        case 12:
                GetDlgItem(IDC_EDIT1)->ShowWindow(FALSE);
                GetDlgItem(IDC_EDIT2)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK1)->ShowWindow(TRUE);
                GetDlgItem(IDC_CHECK2)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK3)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK4)->ShowWindow(FALSE);
                GetDlgItem(IDC_GROUP)->ShowWindow(TRUE);
                GetDlgItem(IDC_FILENAME)->ShowWindow(FALSE);
                GetDlgItem(IDC_STRING)->ShowWindow(FALSE);
                SetDlgItemText(IDC_CHECK1,"Current settings");
                SetDlgItemText(IDC_GROUP,"Display terminal settings");
                m_ctlList.ResetContent();
                m_optLine.SetCheck(0);
                break;
        case 13:
                GetDlgItem(IDC_EDIT1)->ShowWindow(TRUE);
                GetDlgItem(IDC_EDIT2)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK1)->ShowWindow(TRUE);
                GetDlgItem(IDC_CHECK2)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK3)->ShowWindow(FALSE);
                GetDlgItem(IDC_CHECK4)->ShowWindow(FALSE);
                GetDlgItem(IDC_GROUP)->ShowWindow(TRUE);
                GetDlgItem(IDC_FILENAME)->ShowWindow(TRUE);
                GetDlgItem(IDC_STRING)->ShowWindow(FALSE);
                SetDlgItemText(IDC_GROUP,"Options to exec");
                SetDlgItemText(IDC_EDIT1,"");
                SetDlgItemText(IDC_CHECK1,"Take inputs from file?");
                m_ctlList.ResetContent();
                break;
        }
}

void CCheckDlg::OnSelchangeList3()
{
        // TODO: Add your control notification handler code here

        switch(m_ctlDir.GetCurSel())
        {

        case 0:
                GetDlgItem(IDC_DIR)->ShowWindow(TRUE);
                GetDlgItem(IDC_EDIT3)->ShowWindow(TRUE);
                GetDlgItem(IDC_BUTTON1)->ShowWindow(TRUE);
                SetDlgItemText(IDC_DIR,"Enter Directory Name");
                SetDlgItemText(IDC_BUTTON1,"Make Directory");
                SetDlgItemText(IDC_EDIT3,"");
                break;
        case 1:
                GetDlgItem(IDC_DIR)->ShowWindow(TRUE);
                GetDlgItem(IDC_EDIT3)->ShowWindow(TRUE);
                GetDlgItem(IDC_BUTTON1)->ShowWindow(TRUE);
                SetDlgItemText(IDC_BUTTON1,"Remove Directory");
                SetDlgItemText(IDC_EDIT3,"");
                break;
        }

}

void CCheckDlg::OnButton1()
{
        // TODO: Add your control notification handler code here
  AfxSocketInit(NULL);
  LPTSTR lpsz = new TCHAR[100];
  CString l;



  char* servIP = "90.0.0.103";

/*switch here*/
  switch(m_ctlDir.GetCurSel())
  {
  case 0:
          _tcscpy(lpsz,"mkdir ");
          GetDlgItemText(IDC_EDIT3,l);
          _tcscat(lpsz,l);
          _tcscat(lpsz," 2>out");
          break;
  case 1:
          _tcscpy(lpsz,"rmdir ");
          GetDlgItemText(IDC_EDIT3,l);
          _tcscat(lpsz,l);
          _tcscat(lpsz," 2>out");

          break;
}
char* echoString = lpsz;

UINT echoServPort = 1030;
```

```
    CSocket echoClient;
    echoClient.Create();


    echoClient.Connect((LPCSTR)servIP, echoServPort);

    int echoStringLen = strlen(echoString);

    echoClient.Send(echoString, echoStringLen, 0);


    char echoBuffer[RECVBUFSIZE],outbuf[RECVBUFSIZE];

    for(int i=0;i<RECVBUFSIZE;i++)
    {
            echoBuffer[i]=0;
            outbuf[i]=0;
    }

    echoClient.Receive(echoBuffer, RECVBUFSIZE - 1, 0);


    m_ctlList.ResetContent();

    i=0;
    int j=0;
    while(i<strlen(echoBuffer))
    {
            if(echoBuffer[i]!='\n')
                    outbuf[j]=echoBuffer[i];
            else
            {
                    m_ctlList.AddString(outbuf);
                    for(int k=0;k<RECVBUFSIZE;k++)
                            outbuf[k]=0;
                    j=-1;
            }
            i++;
            j++;


    }
   echoClient.Close();
  }

  void CCheckDlg::OnSetfocusList3()
```

```
{
        // TODO: Add your control notification handler code here
        GetDlgItem(IDC_EDIT1)->ShowWindow(FALSE);
        GetDlgItem(IDC_EDIT2)->ShowWindow(FALSE);
        GetDlgItem(IDC_CHECK1)->ShowWindow(FALSE);
        GetDlgItem(IDC_CHECK2)->ShowWindow(FALSE);
        GetDlgItem(IDC_CHECK3)->ShowWindow(FALSE);
        GetDlgItem(IDC_CHECK4)->ShowWindow(FALSE);
        GetDlgItem(IDC_GROUP)->ShowWindow(FALSE);
        GetDlgItem(IDC_FILENAME)->ShowWindow(FALSE);
        GetDlgItem(IDC_STRING)->ShowWindow(FALSE);
        GetDlgItem(IDC_COMMAND)->ShowWindow(FALSE);
        m_ctlList.ResetContent();

}

void CCheckDlg::OnSetfocusList2()
{
        // TODO: Add your control notification handler code here
        GetDlgItem(IDC_DIR)->ShowWindow(FALSE);
        GetDlgItem(IDC_EDIT3)->ShowWindow(FALSE);
        GetDlgItem(IDC_BUTTON1)->ShowWindow(FALSE);
        m_ctlList.ResetContent();
}
```

Tcpser3.c (Linux Server Program)

```
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<stdlib.h>
#include<netinet/in.h>
#include<netdb.h>
#include<errno.h>

FILE *fp,*fperr,*fpfile,*fpinp;

main()
{
int sockMain,sockClient,length,child;
struct sockaddr_in servAddr;

if ( (sockMain=socket(AF_INET,SOCK_STREAM,0)) < 0)
{
perror("Server cannot open main socket.");
exit(1);
}

bzero( (char *) &servAddr,sizeof(servAddr));
servAddr.sin_family=AF_INET;
servAddr.sin_addr.s_addr=INADDR_ANY;
servAddr.sin_port=htons(1030);

if(bind(sockMain,(struct sockaddr *)&servAddr,sizeof(servAddr))==-1)
{
perror("Servers bind failed.");
exit(1);
}

length=sizeof(servAddr);
if(getsockname(sockMain,(struct sockaddr *)&servAddr,&length))
{
perror("getsockname call failed.");
exit(1);
}
printf("SERVER:Port Number is %d\n",ntohs(servAddr.sin_port));

listen(sockMain,5);

for(;;)
```

```
{
if((sockClient=accept(sockMain,0,0))<0)
{
perror("Bad client Socket.");
exit(1);
}

if((child=fork())<0)
{
perror("Failed to create a child.");
exit(1);
}
else if (child==0)
{
close(sockMain);
childWork(sockClient);
close(sockClient);
exit(0);
}

close(sockClient);
}
}

#define BUFLEN 10000
int childWork(sockClient)
int sockClient;
{
char buf[BUFLEN],outbuf[BUFLEN],ch;
int msgLength,cnt,i;

bzero(buf,BUFLEN);
if((msgLength=recv(sockClient,buf,BUFLEN,0))<0)
{
perror("Bad receive by chld.");
exit(1);
}
printf("SERVER:The unix command sent from client is:%s\n",buf);
if(buf[0]=='e'&&buf[1]=='x')
{
fpfile=fopen("abc.c","w");
i=2;
while(i<msgLength && buf[i]!='@')
{
fputc(buf[i],fpfile);
i++;
}
fputc('\n',fpfile);
fclose(fpfile);
i++;
fpinp=fopen("inp","w");
while(i<msgLength)
{
fputc(buf[i],fpinp);
i++;
}
fclose(fpinp);
system("rm -f err");
system("cc -o abc abc.c 2>err");
fperr=fopen("err","r");
if((ch=fgetc(fperr))==-1)
{
        system("./abc <inp >out");
}
else
{
        fp=fopen("out","w");
        fclose(fp);
}
fclose(fperr);
}
else
{
 system(buf);
}
fp=fopen("out","r");

for(i=0;i<BUFLEN;i++)
outbuf[i]=0;
cnt=0;
while(!feof(fp))
{
ch=getc(fp);
outbuf[cnt]=ch;
cnt++;
}
fclose(fp);
if(cnt==1)
{
fperr=fopen("err","r");

for(i=0;i<BUFLEN;i++)
```

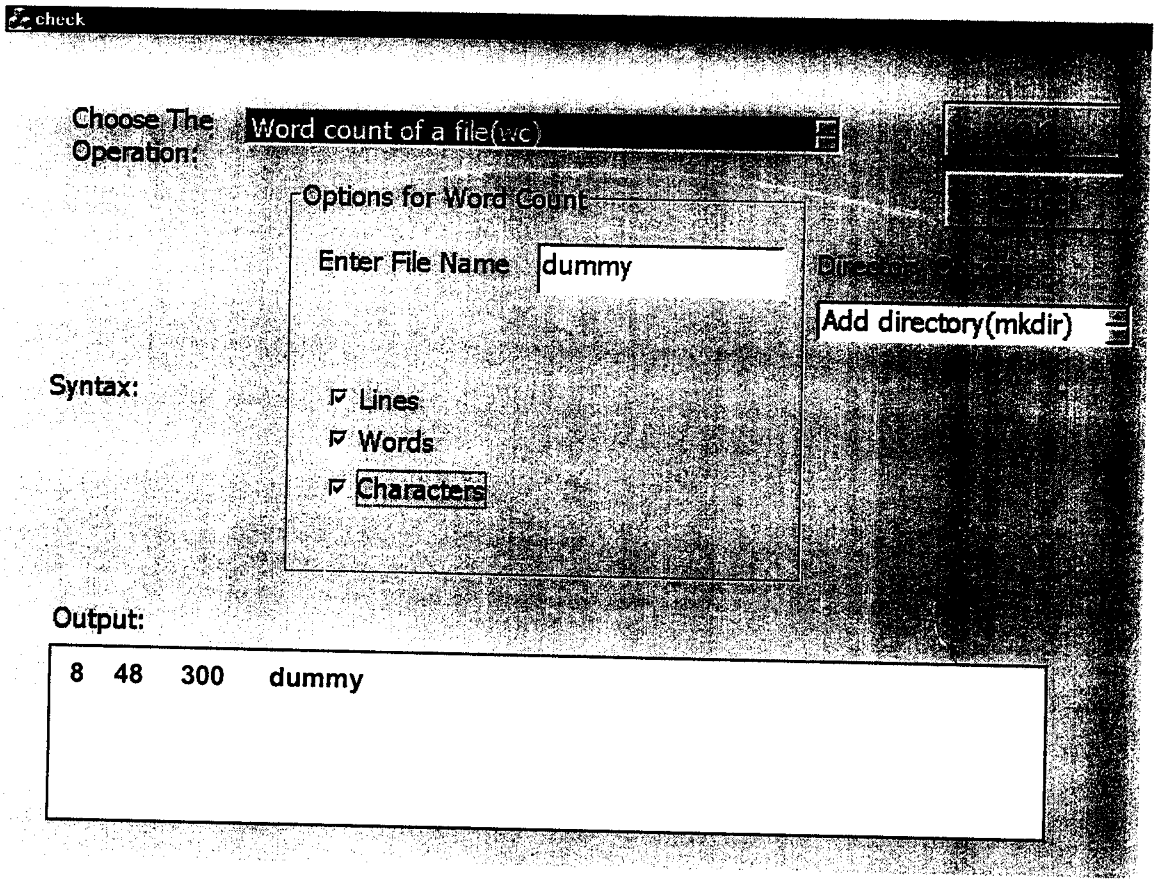
```
outbuf[i]=0;
cnt=0;
while(!feof(fperr))
{
ch=getc(fperr);
outbuf[cnt]=ch;
cnt++;
}
fclose(fperr);
}


printf("%s",outbuf);
send(sockClient,outbuf,cnt-1,0);
}
```

APPENDIX – B

Sample Screens

Screen Shot 1: Executing shell command(wc)



Word count of a file(wc)

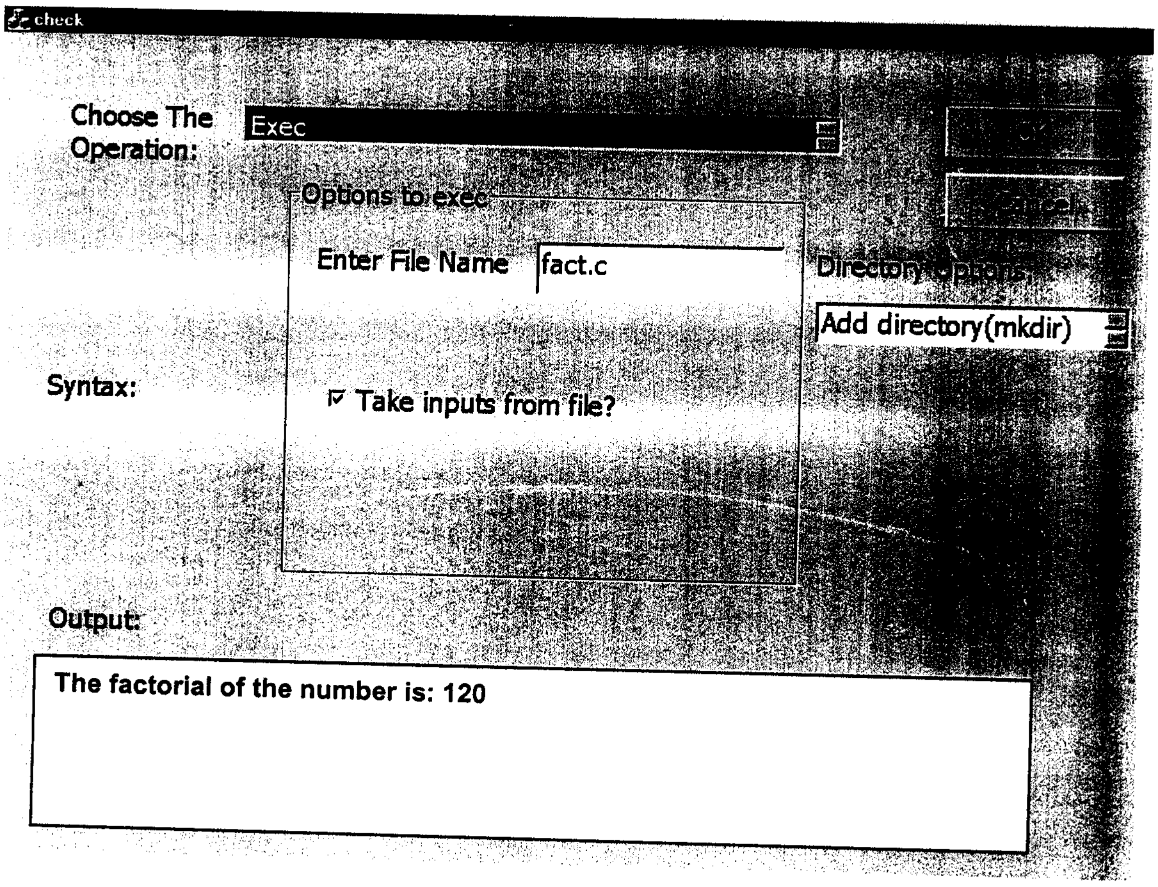
Enter File Name: dummy

Add directory(mkdir)

☑ Lines
☑ Words
☑ Characters

```
8   48   300    dummy
```

Screen Shot 2:Executing C program



Choose The Operation: Exec

Options to exec

Enter File Name fact.c

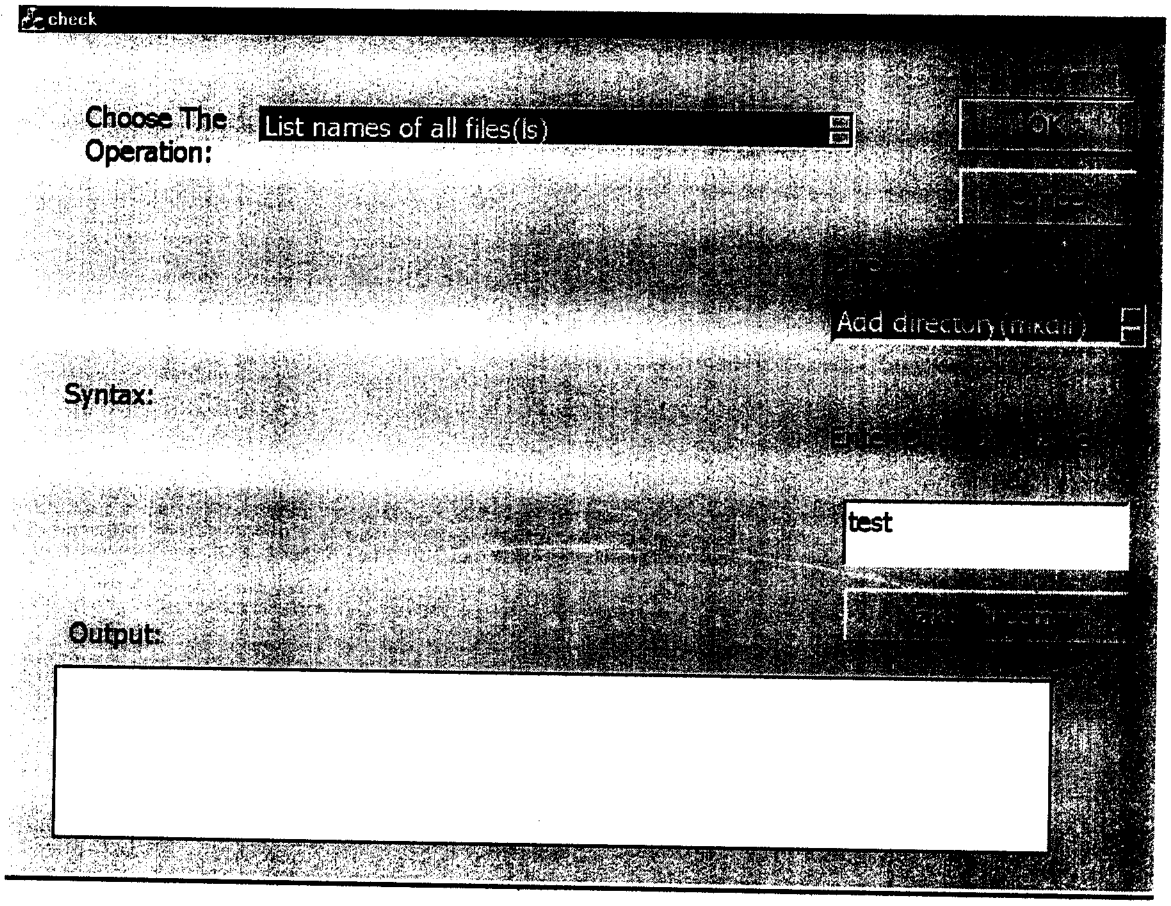
☑ Take inputs from file?

Directory Options: Add directory(mkdir)

Output:

```
The factorial of the number is: 120
```

Screen Shot 3:Executing directory options(mkdir)



test