Online Development

Exercises

ProBiller Online Development Exercises

Table of Contents

Exercise

1

Checking Out a Program Using Visual SourceSafe

Overview

Our version control tool is Visual SourceSafe. In this exercise, we will use Visual SourceSafe to establish a working directory, perform a “get,” check out a program, and check a program back in.

Approximate Time for Completion

One hour

Tools for Completion

- Computer terminal set up with Visual SourceSafe, Version 5.0, and Microsoft Developer Studio
- Visual SourceSafe user name and ID number for each student

Starting the Visual SourceSafe Explorer

To start Visual SourceSafe, take the following steps.

1) Select the Start button on the Windows 95 taskbar.

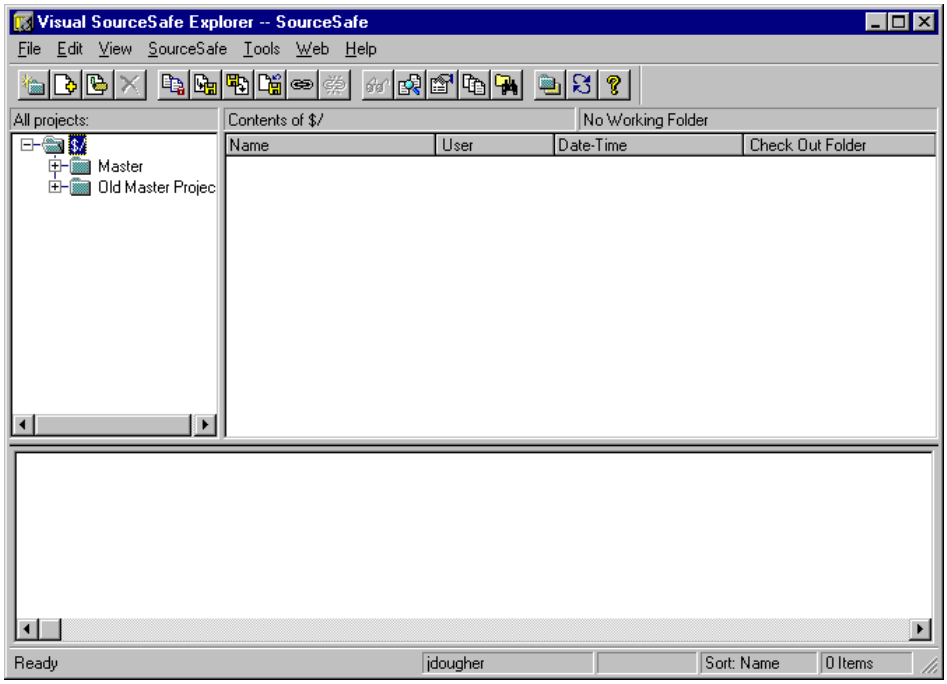
2) Select Programs, Visual SourceSafe.

✓ *The Visual SourceSafe login dialog box appears.*

3) Type your username and password into the Visual SourceSafe Login dialog box.

🕮 *Your instructor will provide you with a username and password.*

4) Select OK.

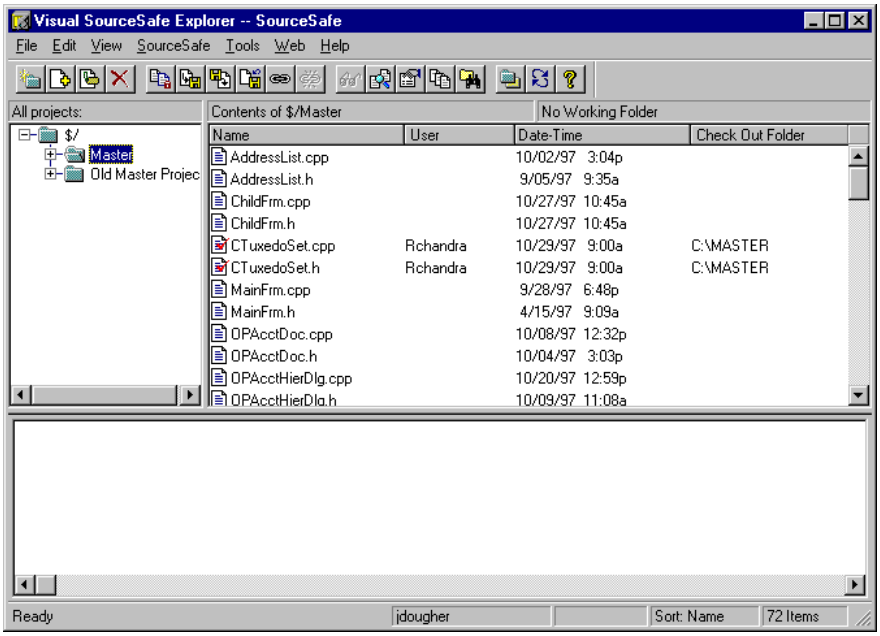


✓ *Visual SourceSafe opens.*

🕮 *The Visual SourceSafe window looks similar to Windows Explorer, with the directory structure of your project displayed in a hierarchical tree view*

mode in the upper left-hand pane, and the contents of the currently selected directory in the upper right-hand pane.

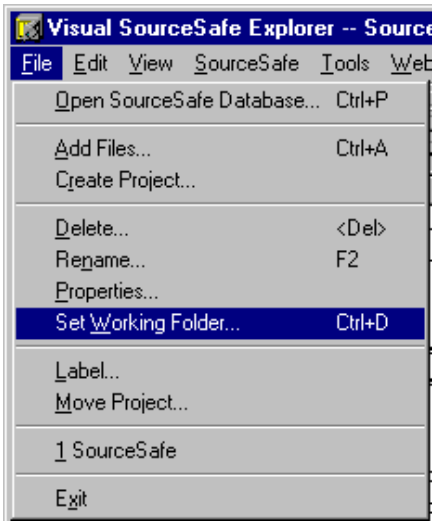
5) Select the Master folder in the left-hand pane of the Visual SourceSafe Explorer window.



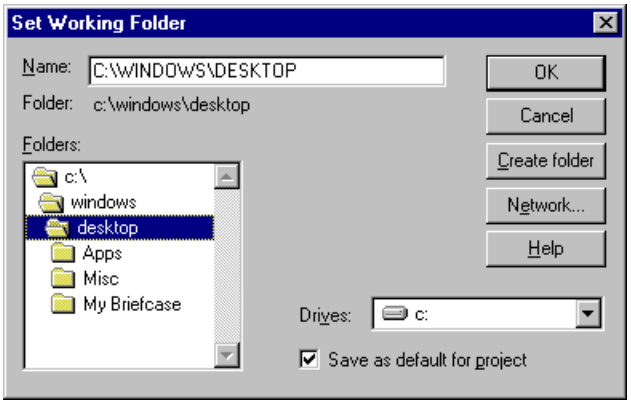
Establishing a Working Directory

Before checking out a program, you must establish a working directory (or folder) on your local hard drive to hold the programs that you check out. To establish a working directory, make sure the Master directory is highlighted in Visual SourceSafe, and then take the following steps.

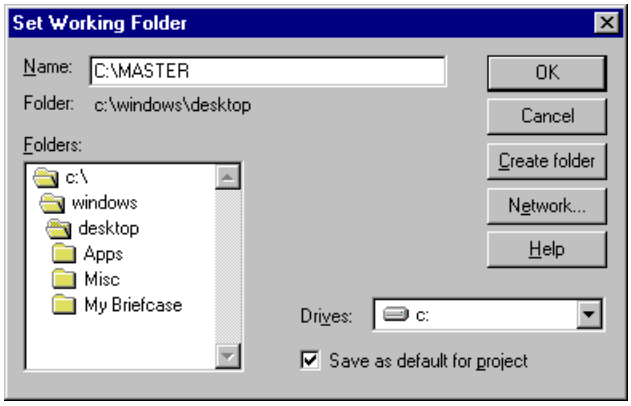
1) Select Set Working Folder from the File menu in Visual SourceSafe.



✓ *The Set Working Folder dialog box opens.*



2) Move to the Name text box, and change C:\WINDOWS\DESKTOP to **C:\MASTER**.



3) Select the Create folder button.

📖 *Make sure the **Save as default for project** checkbox is marked.*

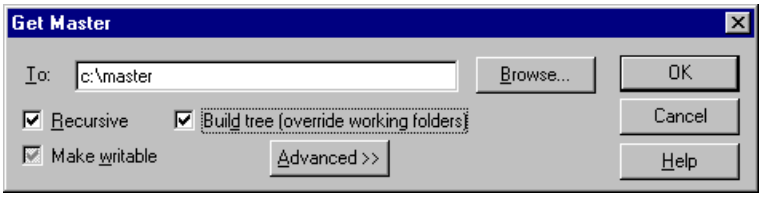
4) Select OK.

✓ *The program's directory structure is replicated on your local hard drive.*

Performing a "Get"

Now that the working directory has been established on your hard drive, you must create a copy of the files that make up the project. To do so, you perform a "get" by taking the following steps.

1) Select Get Latest Version from the SourceSafe menu in Visual SourceSafe.



✓ *The Get Master dialog box opens.*

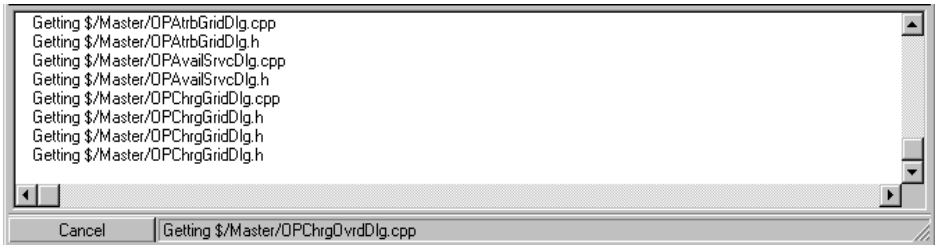
🕮 *By default, the directory you created when establishing a working directory appears in the To text box of the Get Master dialog.*

2) Mark the Recursive check box.

✓ *When you mark the Recursive check box, the Build tree (override working folders) check box appears.*

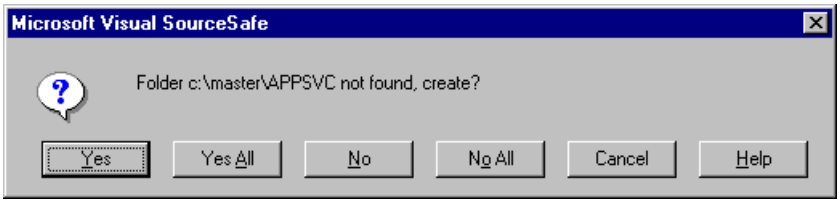
3) Mark the Build tree (override working folders) check box.

4) Select OK to complete the process of getting the program files.



✓ *The lower horizontal pane of the Visual SourceSafe Explorer shows you the status of the Get operation.*

5) If a dialog box appears asking whether you want to create folders that were not found on your local hard drive, select the Yes All button.



✓ *Upon completion, the word "ready" appears in the status bar of the Visual SourceSafe Explorer.*

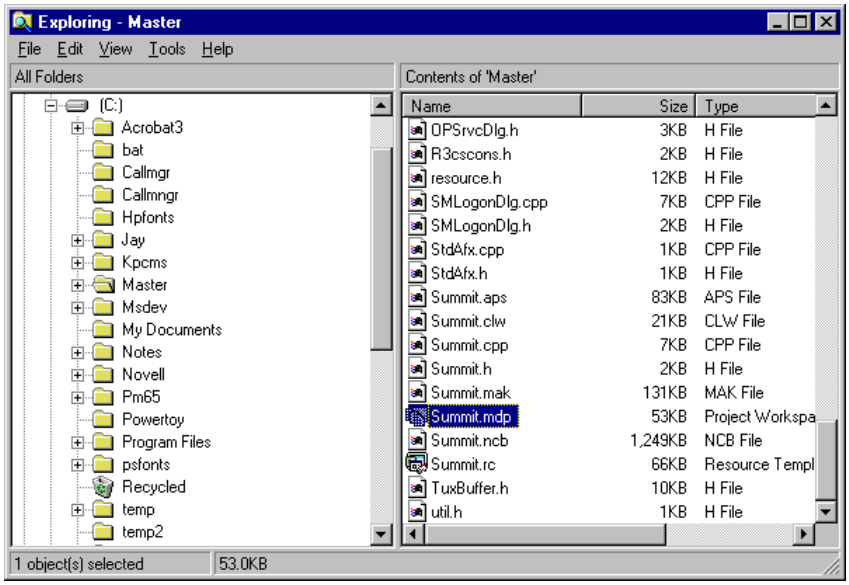
Removing Read-only Attributes

After you perform a "get," you cannot immediately modify any of the program files that make up ProBiller, since the files are marked read-only. Before you edit any of the files, you must remove the read-only attribute from Summit.mdp and Summit.ncb. The easiest way to do this is with Windows Explorer.

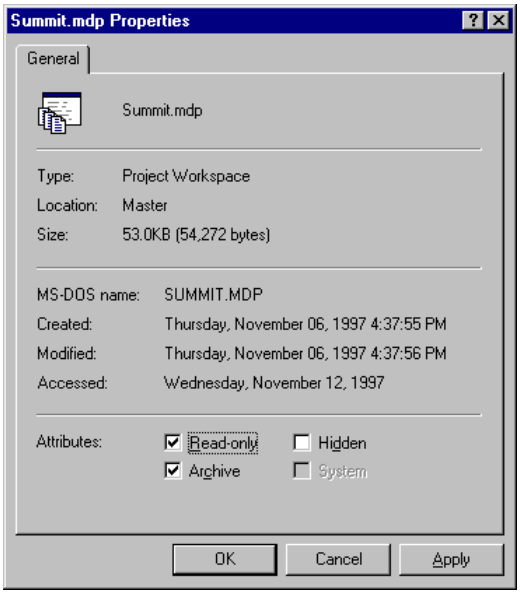
1) Open Windows Explorer by selecting Start, Programs, Windows Explorer.

2) Select the Master folder  in the left-hand pane of Explorer.

3) In the right-hand pane of Explorer, find and select the Summit.mdp file.



4) Select Properties from the File menu of Explorer.



✓ *The Properties dialog box appears.*

5) In the Attributes section of the Properties dialog box, uncheck the Read-only check box.

6) Select OK.

✓ *You have now removed the read-only attribute from the Summmit.mdp file.*

7) In the right-hand pane of Windows Explorer, select the Summit.ncb file.

8) Use the same procedure to remove the read-only attribute from Summitncb.

Alternate Method of Removing Read-only Attributes

You can also remove the read-only attributes of all of a project's files at the same time you perform a "get" by using the Get Latest Version toolbar button . Note that this method does not allow you to selectively remove read-only attributes, but this method may be preferable to the preceding one, depending upon your needs.

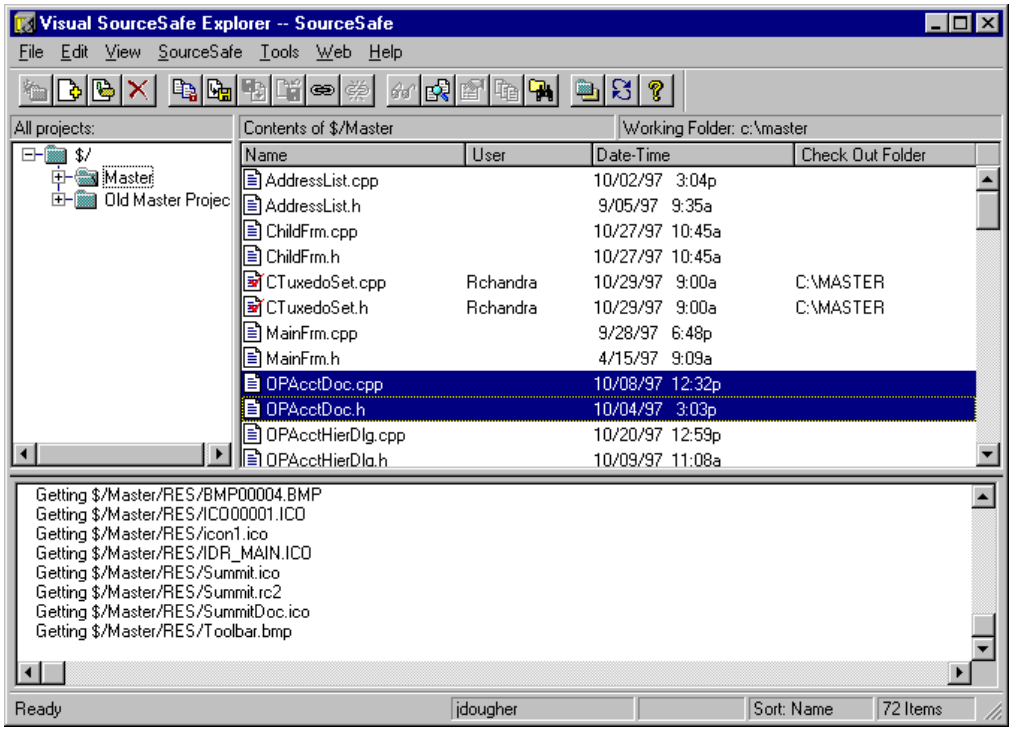
To use this method, take the following steps.

1)

Checking Out the Program

After you have established your working folders, performed a get on the project files, and removed the read-only attributes from Summit.mdp and ProBiller.ncb, you're ready to check out a file using Visual SourceSafe so that you can modify it.

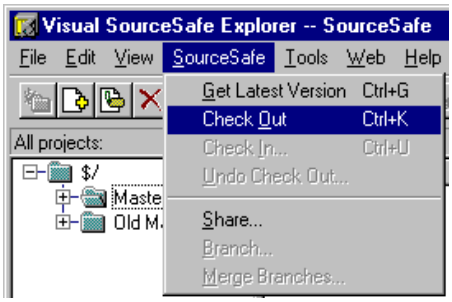
1) From the right-hand pane of the Visual SourceSafe Explorer window, select the program files you wish to check out. For the purposes of this exercise, select the OPAcctDoc.cpp and OPAcctDoc.h files, as illustrated in the figure below.



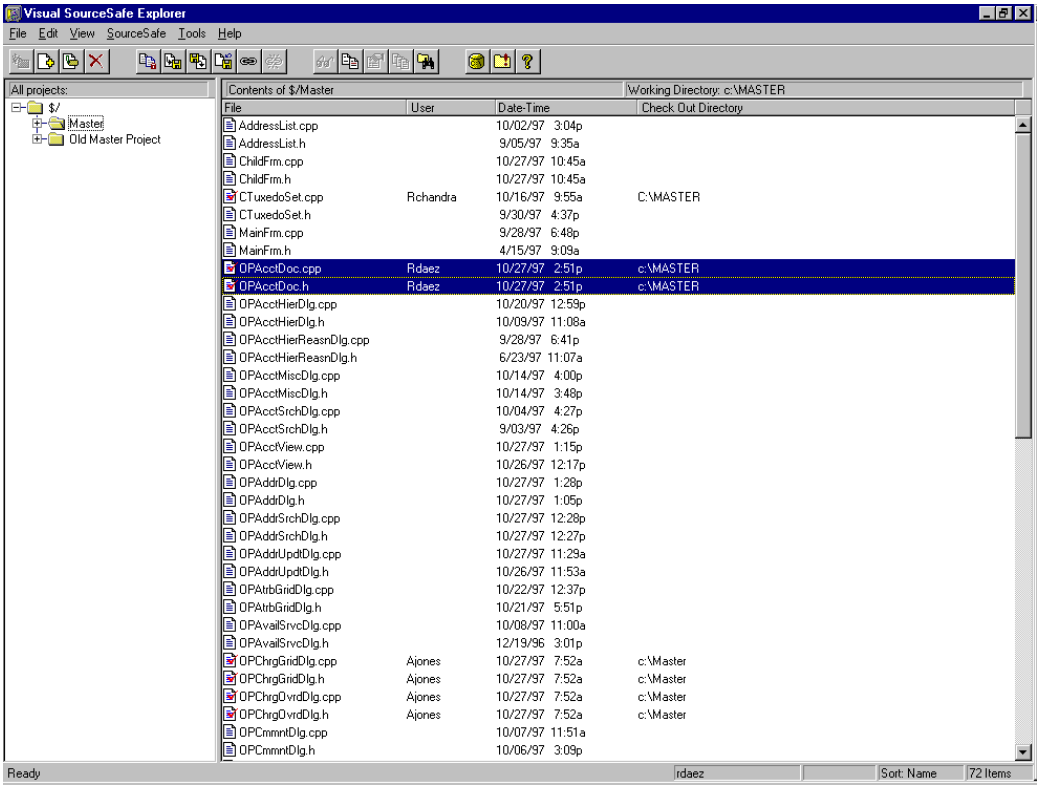
📖 *During normal working conditions, please check out only those project files that you need at the moment.*

📖 *Note that you can select contiguous files by holding down the left mouse button and dragging the mouse pointer over the names of the files you wish to select. To select non-contiguous files, hold down the <CTRL> key as you select files.*

2) Select Check Out from the SourceSafe menu of the Visual SourceSafe Explorer.



✓ *After the Check Out process is complete, your user name and the check out directory will appear next to the filenames you checked out in the right-hand pane of the Visual SourceSafe Explorer window, as illustrated in the figure below.*



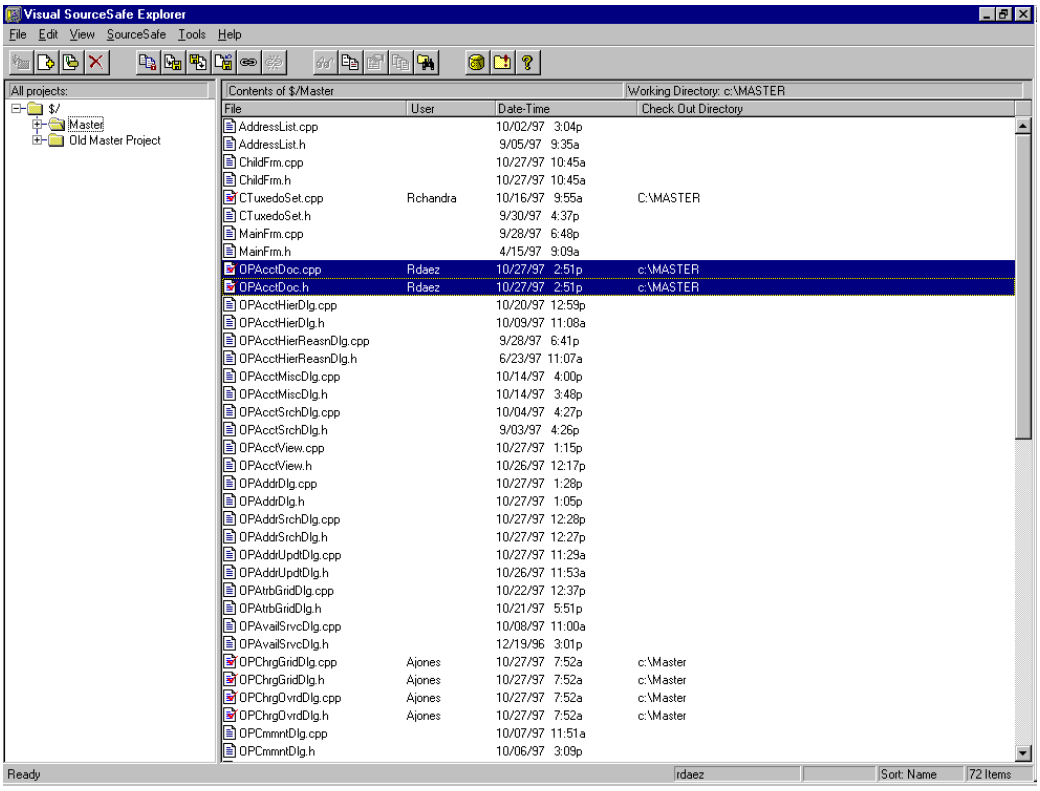
3) You may now edit the programs you have checked out using Microsoft Developer Studio or another editing tool.

Checking In the Program

When you have finished editing a program that you checked out using Visual SourceSafe, you need to check the program back in.

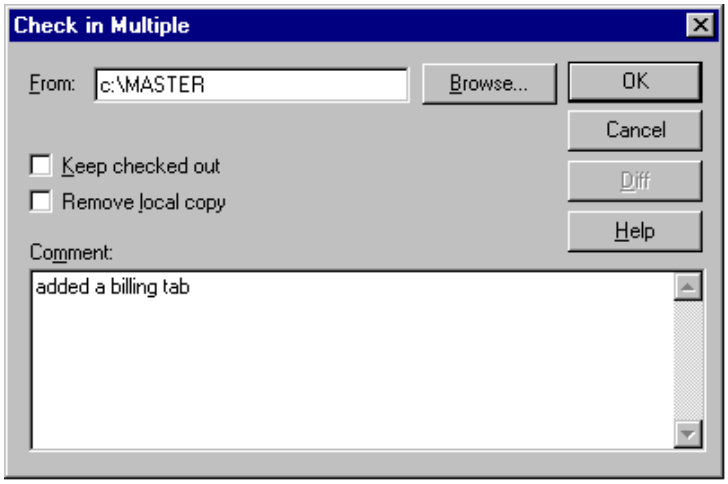
1) Open or switch to the Visual SourceSafe Explorer.

2) In the Explorer's right-hand pane, select the program files you wish to check back in.



3) From the SourceSafe menu, select Check In.

✓ *The Check in Multiple dialog box opens.*



4) In the Comment section of the Check in dialog box, type a description of the changes you made to the program.

5) Select OK.

This completes the demonstration exercise for checking out a program using Visual SourceSafe.

Exercise
2

Editing and Compiling Presentation Layer Classes and Application Services

Overview

This exercise covers the following topics:

- Opening the ProBiller workspace in Microsoft Developer Studio
- Editing presentation layer classes and application services
- Programming etiquette
- Function headers
- Line comments
- Saving, compiling, building the source code

Approximate Time for Completion

Two hours

Tools for Completion

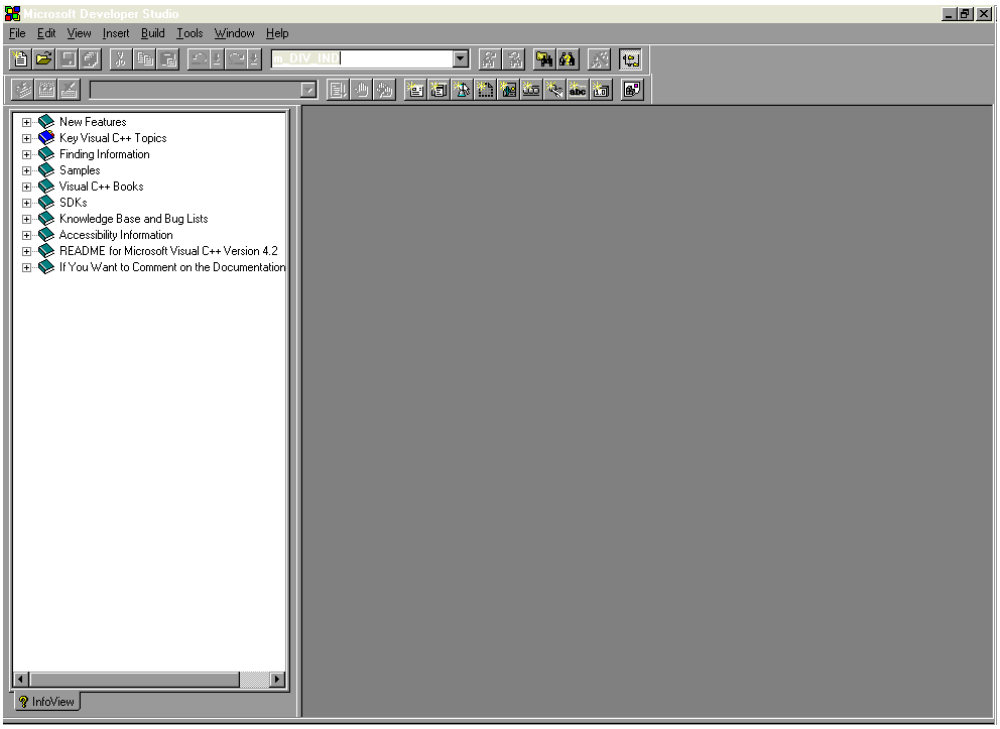
- Computer set up with Microsoft Developer Studio and SourceSafe
- User name and ID number
- This handout

Editing Presentation Layer Classes and Application Services

Microsoft Developer Studio is used to edit presentation layer classes and application services.

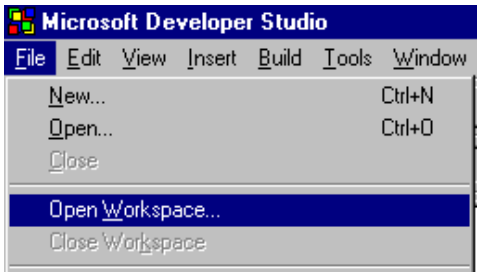
In this exercise, we will open the ProBiller project file in Microsoft Developer Studio and practice editing the presentation layer file OPAcctDoc.cpp.

1) From the Windows Start menu, select Programs, Microsoft Visual C++, Microsoft Developer Studio.

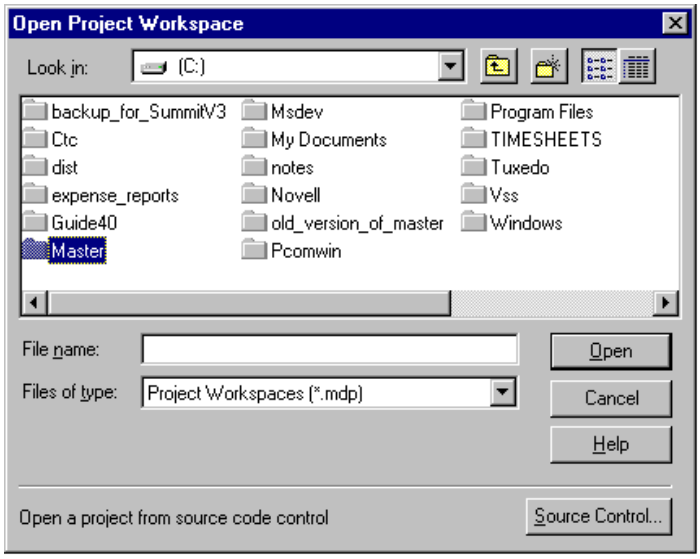


✓ *Microsoft Developer Studio opens.*

2) From the File menu, select Open Workspace.

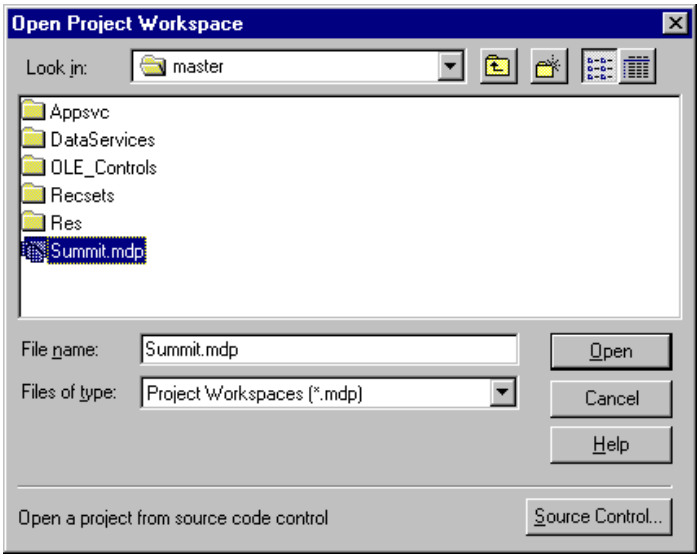


✓ *The Open Project Workspace dialog box opens.*



3) From the Open Project Workspace dialog, double-click on the Master directory, displayed in the figure above.

🕮 *You created the Master directory in Exercise 1 when you replicated the ProBiller directory structure with Visual SourceSafe.*



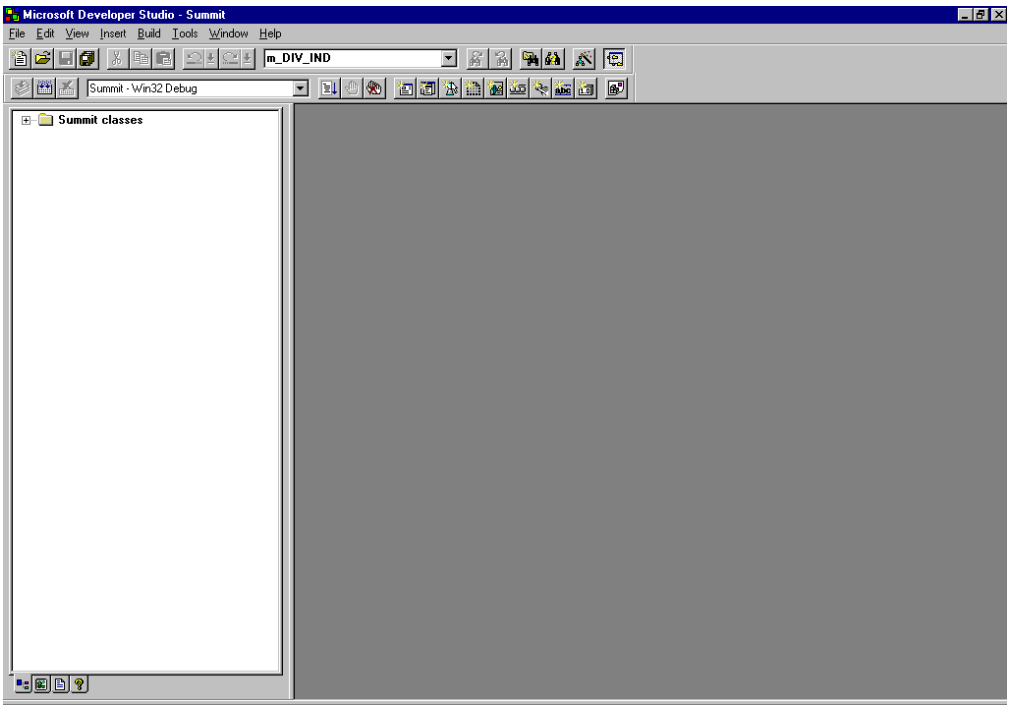
✓ *The Open Project Workspace should now display the contents of the Master directory.*

4) Select the ProBiller*.mdp* file from the folder and file listing.

🕮 *Files that end in* ***.mdp*** *are Project Workspace files. The "mdp" stands for Microsoft Developer Project.*

5) Select the Open command button.

📖 *A dialog box that reads, "Cannot access the ClassView information file. ClassView information will not be available." may appear. Simply select OK to remove this dialog box.*



✓ *The ProBiller classes folder opens in the left-hand pane of Microsoft Developer Studio.*

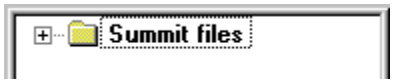
Selecting a View

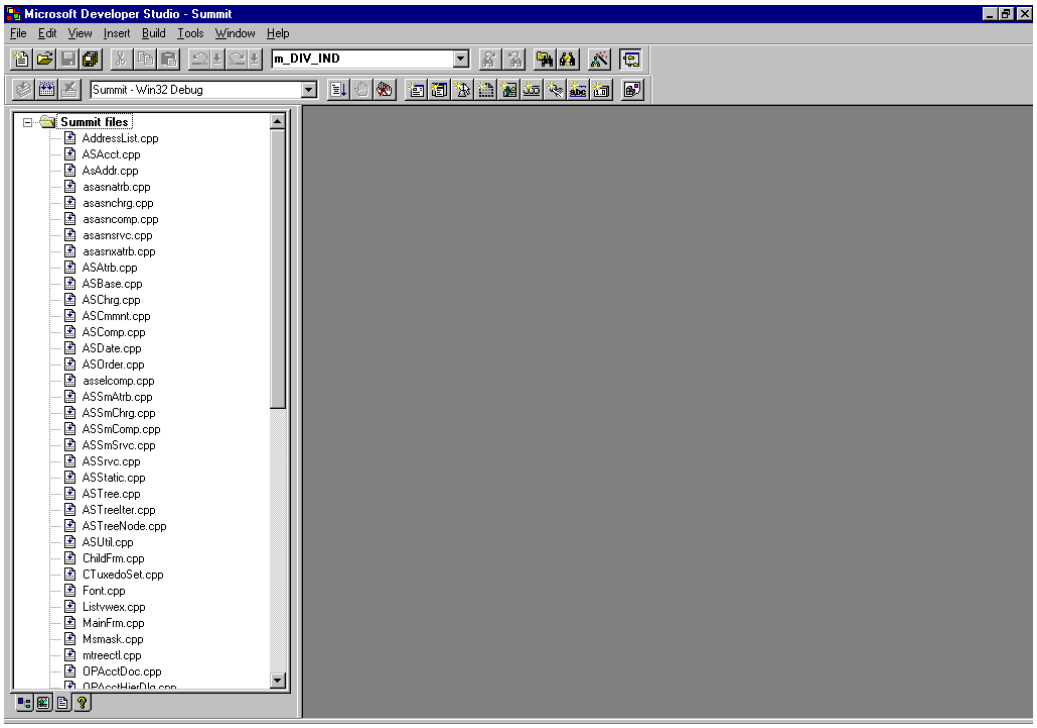
Note that there are four tabs attached to the bottom of the left-hand pane of Developer Studio. If you allow the mouse cursor to hover over each tab, a pop-up balloon will provide you with their names. From left to right, the tabs represent the Class View, Resource View, and File View of the project currently open. The fourth tab represents Help.

6) Select the third tab from the left, the File View tab.

📖 *The File View allows us to see all of the ProBiller program files. The default Class View will not.*

7) Expand the **ProBiller files** tree in the left-hand pane by clicking once on the plus sign.



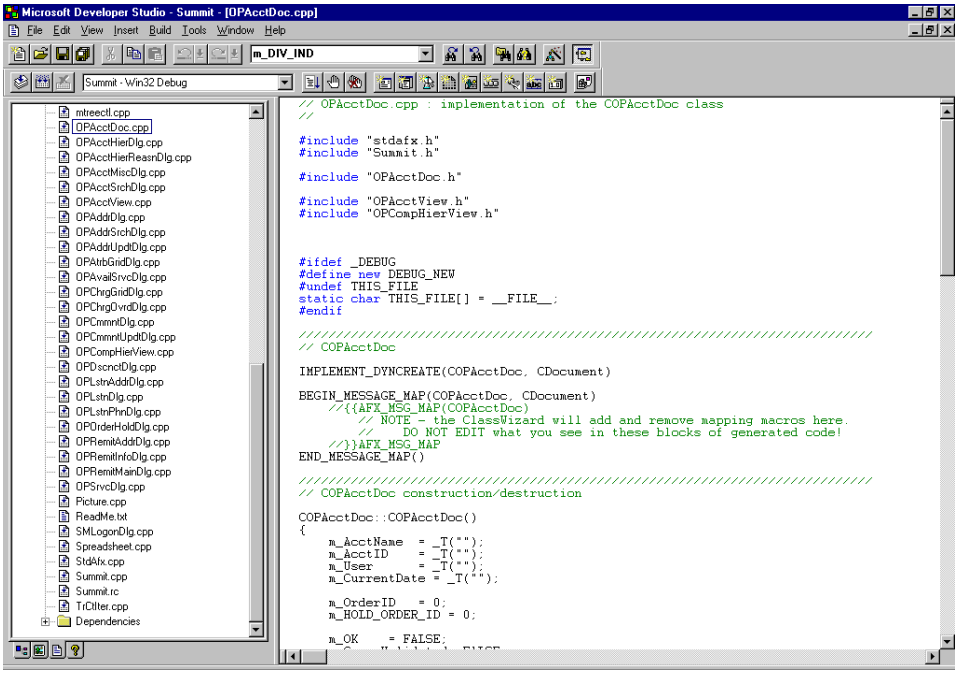


✓ *The tree expands, revealing the ProBiller presentation layer classes and application services.*

Editing the Program

For the purposes of this exercise, we will edit the presentation layer class that we checked out in Exercise 1 — OPAcctDoc.cpp.

8) Scroll down in the ProBiller file listing until you see OPAcctDoc.doc. Double-click on the filename.



✓ *The file opens in the right-hand pane of Microsoft Developer Studio.*

🕮 *Opening and editing application services is performed the same way. Application services begin with the letters* AS*.*

9) Edit the program.

Programming Etiquette

When editing code, ensure that you adhere to ProBiller online programming standards. Although these are covered elsewhere in the course, here is a reminder of the programming "etiquette" you should keep in mind:

- Include a header for every new function
- Strive for readability: use correct indentation, spaces in-between lines, and variable conventions
- Use comments liberally within functions

Adding Comments

There are two types of comments you should add to programs as you edit them: headers and individual line comments.

Headers

A header should contain the following information:

- Name of function
- Purpose of function: a textual description
- Inputs
- Outputs

The following figure illustrates a function header:

```
////////////////////////////////////////////////////////////////////////////
//
// Name: SetRemitAcctPtr
//
// Purpose: This method sets, for each account in a hierarchy, the pointer to its
//          remittance account in memory.  This method is called after an account
//          hierarchy has been loaded.
//
//          The remittance pointer was not set when the account hierarchy
//          was loaded because it's possible for an account to be loaded before its
//          remittance account.
//
// Inputs:  None.
//
// Output:  None.
//
////////////////////////////////////////////////////////////////////////////
```

You can make the job of correctly formatting a function header easier by copying an existing function header from another part of the program you're working on and pasting it where you need it. At that point, you can delete the existing header text and replace it with your own.

Line Comments

You should use line comments liberally to document individual sections of code.

The highlighted line in the following figure is an example a line comment.

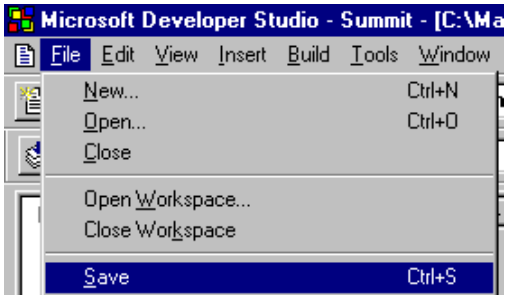
```
void ASAcct::SetRemitAcctPtr()
{
    ASTreeNode_Iter iter(this);

    ASAcct* acct;

    while((acct = (ASAcct*)iter.GetNext()) != NULL)
    {
        // find its remittance account
        acct->m_remittance = acct->GetRemitAcctPtr();
    }
}
```

Finishing Up

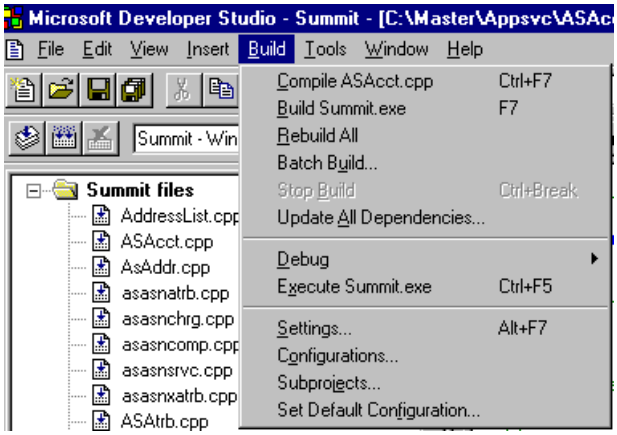
When you have finished editing the program, you can do one of three things:

- Save the program
- Compile the program
- Build the program

10) To save the program, simply select Save from the File menu of Developer Studio, or press <Ctrl>+<S>.



11) To compile the program, select Compile from the Build menu of Developer Studio.



📖 *Compiling automatically saves the program.*

12) To build the program, select Build from the Build menu of Developer Studio.

📖 *Build first saves all of the programs that have been modified, compiles them, and finally builds the executable.*

📖 *Note that there are toolbar buttons and keyboard shortcuts available for the compile and build functions.*

This completes the exercise.

Exercise

3

Adding a New Function

Overview

This exercise steps you through the process of adding a new function to a presentation layer class or application service. It draws upon skills learned in the first two exercises and asks you to perform the following activities:

- Check out the program using SourceSafe
- Use the Developer Studio ClassWizard
- Insert a function header
- Add a new function
- Compile and build the program
- Use SourceSafe to check the program back in

Approximate Time for Completion

One hour

Tools for Completion

- Computer set up with Microsoft Developer Studio and SourceSafe
- User name and ID number
- This handout

Adding a Function to OPAcctView

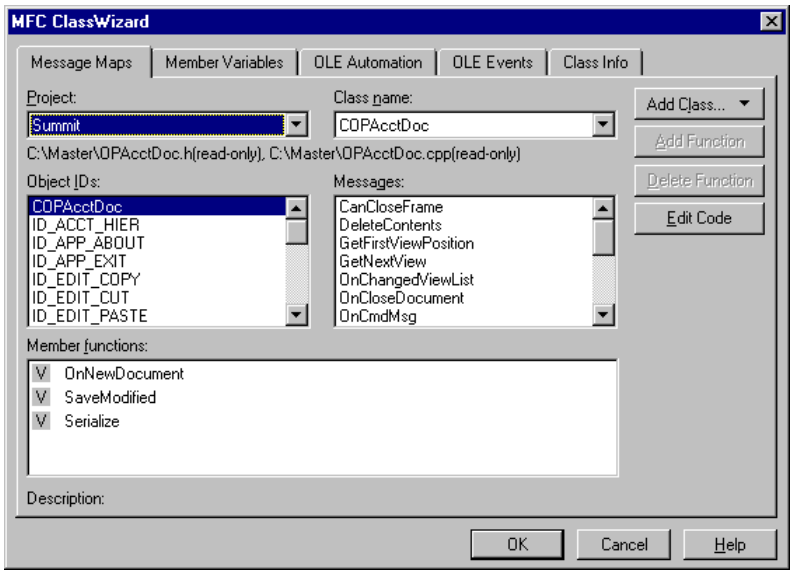
Before beginning this exercise, use SourceSafe to check out the presentation layer class OPAcctView. If necessary, refer to Exercise 1.

Using the Class Wizard

The easiest way to add a function is to use the “Class Wizard” in Microsoft Developer Studio. You can access the Class Wizard in one of three ways:

- By selecting Class Wizard from the View menu
- By pressing <Ctrl>+<W>
- By clicking the Class Wizard button on the Standard toolbar

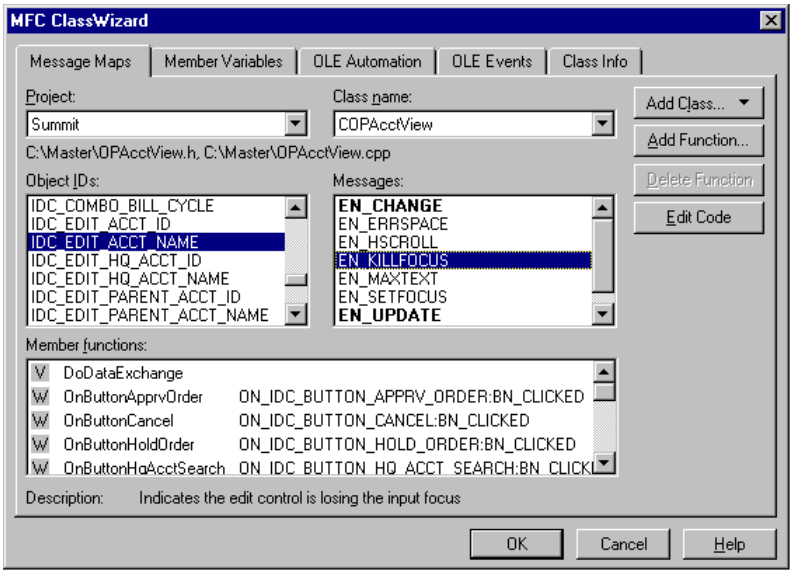
1) Start the Class Wizard.



✓ *The Class Wizard dialog box opens, with the focus on the Message Maps tab.*

2) Search for the class to which you would like to add a function by using the **Class name** drop-down list. In this exercise, we’ll select COPAcctView.

✓ *For each object listed in the **Class name** drop-down list, a corresponding list of available events or messages appears in the Object IDs list box and the Messages list box, as illustrated in the following figure.*

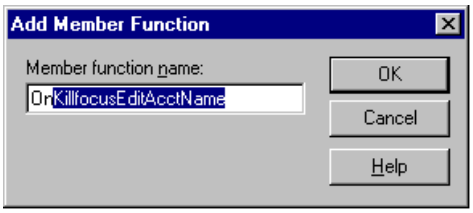


📖 *Messages that are already used in the program are listed in **bold** type in the Messages list box. The list of available messages changes when you select different Object IDs. When you select a particular message, such as EN KILLFOCUS above, you will note that an overview of the function appears on the Description line toward the bottom of the dialog box.*

3) Select IDC_EDIT_ACCT_NAME in the Object IDs list box.

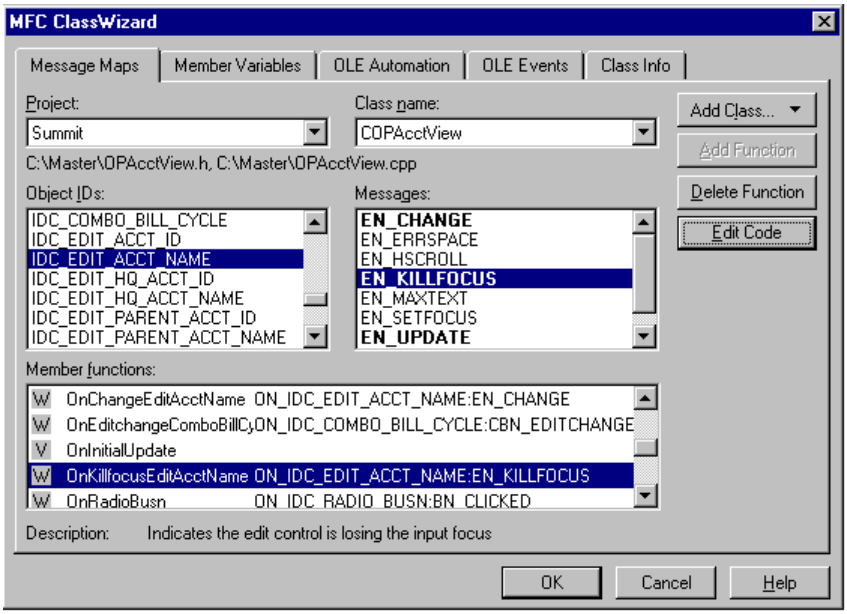
4) Select EN KILLFOCUS in the Messages list box.

5) Select the Add Function command button.



📖 *The Add Member Function dialog box automatically suggests a member function name.*

6) Select OK to accept the default member function name.

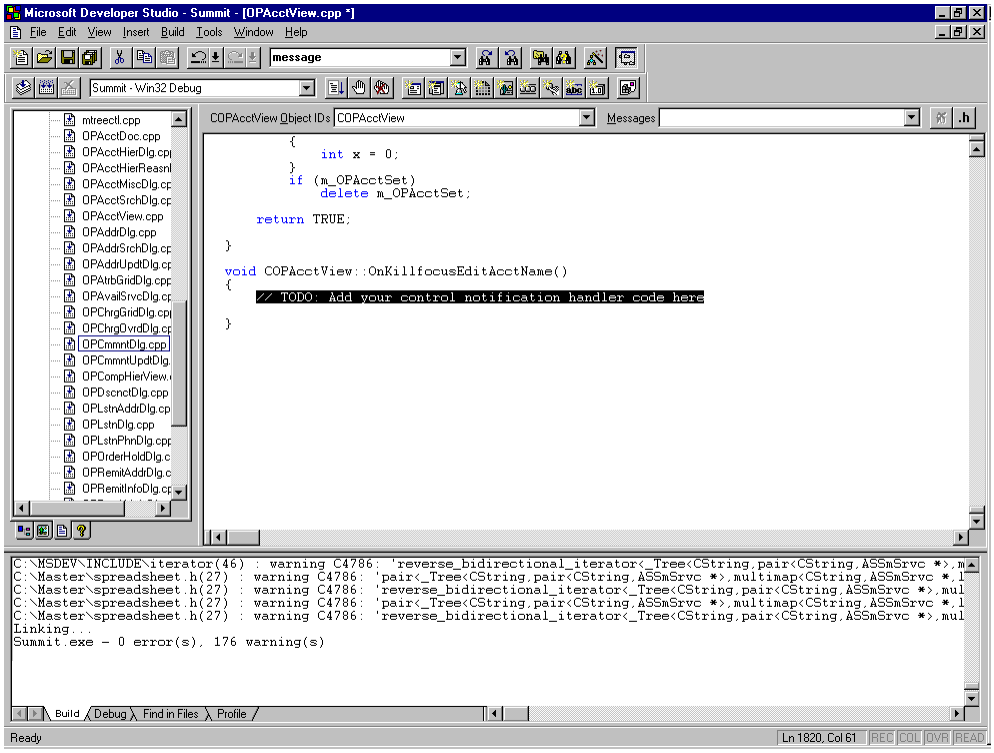


✓ *OnKillfocusEditAcctName is added to the Member Functions list.*

Moving to the Location of the New Function

With the MFC Class Wizard dialog box still open, you can now do one of two things: select the Edit Code command button to move directly to the new function we just created in OPAcctView, or select the OK command button, which would return us to Microsoft Developer Studio. If you choose the latter, you will need to move to OPAcctView in the ProBiller file listing to edit the function.

7) Select the Edit Code command button in the MFC ClassWizard dialog box.



✓ *The ClassWizard opens OPAcctView in the right-hand pane of Developer Studio, writes the function name, and indicates where you need to add your code by inserting "// TODO: Add your command handler code here," illustrated in the figure above.*

Inserting the Header

As we learned in the previous exercise, the easiest way to add a header and ensure formatting consistency is by copying an existing header from another location and pasting it into the program we're editing.

8) Locate a header in OPAcctView.cpp or in another presentation layer class, such as OPAcctDoc.cpp.

9) Highlight the header.

```
void COPAcctView::CloseView()

////////////////////////////////////////////////////////////////////////////
// Description:     This function closes and deletes the dialogs that were opened
//                  under the account view.  It also deletes the account hierarchy
//                  and closes the pointer to the account view's document.
// Inputs:          N/A
// Outputs:         N/A
// Return Value:    N/A
////////////////////////////////////////////////////////////////////////////

{
    // delete account hier
```

10) Copy it to the Windows clipboard by pressing <Ctrl>+<C> or by selecting Copy from the Edit menu.

11) If you copied the header from a presentation layer class other than OPAcctView, move back to OPAcctView by selecting it in the left-hand pane of Developer Studio.

12) Place the insertion point *before* the open brace ({) that precedes the TODO line in OPAcctView.cpp.

13) Press <Enter> to create a blank line.

14) Move the insertion point up to the beginning of the blank line you created.

15) Paste the header at the cursor location by pressing <Ctrl>+<P> or by selecting Paste from the Edit menu.

```
void COPAcctView::OnKillfocusEditAcctName()

////////////////////////////////////////////////////////////////////////////
// Description:     This function closes and deletes the dialogs that were opened
//                  under the account view.  It also deletes the account hierarchy
//                  and closes the pointer to the account view's document.
// Inputs:          N/A
// Outputs:         N/A
// Return Value:    N/A
////////////////////////////////////////////////////////////////////////////

{
```

✓ *The header appears at the cursor location and should look similar to the figure above.*

16) Highlight and delete the existing description, and type a new description in its place, as illustrated in the following figure.

```
void COPAcctView::OnKillfocusEditAcctName()

////////////////////////////////////////////////////////////////////////////
// Description:     This function displays a message when the focus on the Account
//                  Name is lost.
// Inputs:          N/A
// Outputs:         N/A
// Return Value:    N/A
////////////////////////////////////////////////////////////////////////////

{
```

Adding the New Function

17) Type in the new function or paste it in from another location, and add a line comment to explain what the function does.

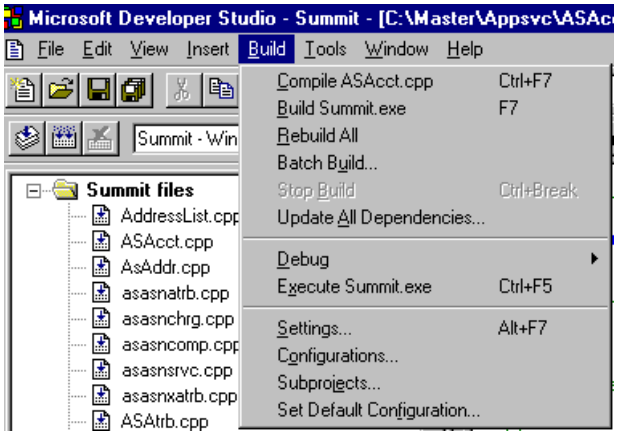
```
void COPAcctView::OnKillfocusEditAcctName()
{
    // display message when leaving account name field
    MessageBox("Leaving Account Name field", "Account",MB_OK);

}
```

📖 *You may wish to use the same method we used when copying and pasting a header for retrieving the proper format for a function.*

Finishing Up

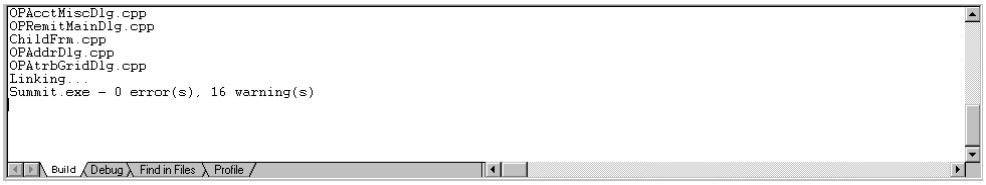
When you have finished adding the function, you should test it by compiling the program and building the application.

18) Select Compile from the Build menu of Developer Studio.



📖 *Compiling automatically saves the program.*

19) To build the program, select Build from the Build menu of Developer Studio.

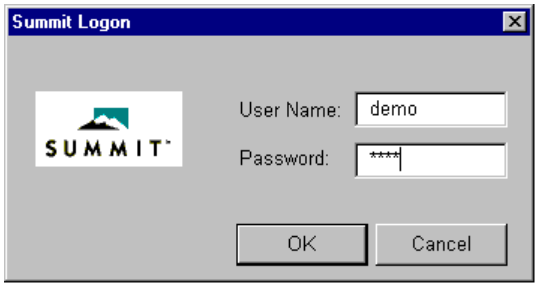


✓ *The results of the build appear in the status window of Developer Studio.*

Executing the Program

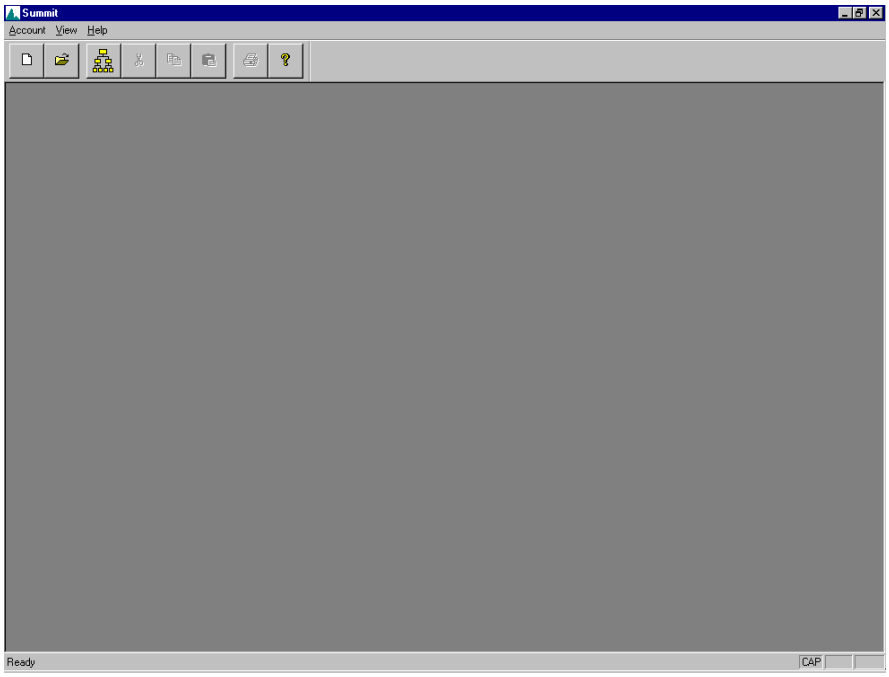
To test the new function within ProBiller, take the following steps.

20) Select Execute ProBiller.exe (Ctrl+F5) from the Build menu of Developer Studio.



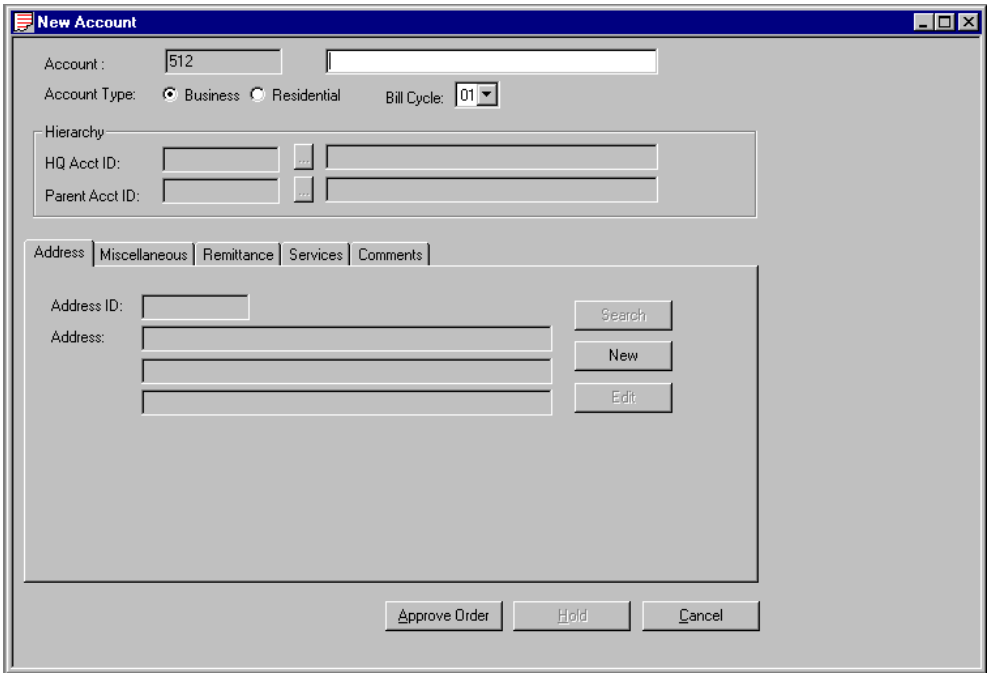
✓ *The ProBiller Logon dialog box appears.*

21) Type your user name and password into the ProBiller Logon dialog box, and then select OK.



✓ *The ProBiller application window appears.*

22) Select the New Account button from the ProBiller button bar or choose New (Ctrl+N) from the Account menu.



✓ *The Account View window opens.*

23) Type a name into the Account Name text box.

24) Press <Tab> or use the mouse to move to another entry point in the Account View window.

✓ *A dialog box appears, showing the results of the new function that we added.*

25) Exit ProBiller by selecting Exit from the Account menu or by pressing <Alt>+<F4>.

Using SourceSafe to Undo the Checkout

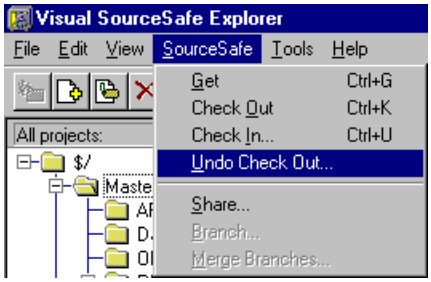
Normally, after adding a function to a program and testing the function to make sure it works, we would then use SourceSafe to check the program back in (see the following section).

Since the function we added was just an exercise, we do not wish to keep it, so we'll use SourceSafe to undo checking out the program.

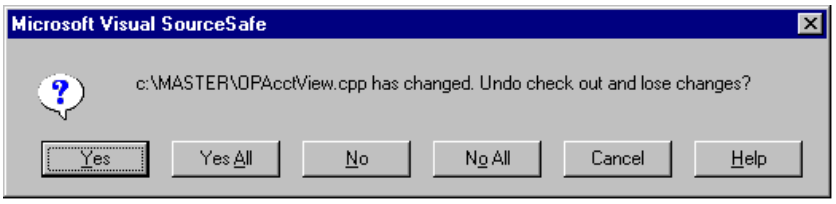
26) Open SourceSafe, or switch to it if it is still open.

27) Highlight OPAcctView.cpp and OPAcctView.h in the file list pane of the SourceSafe Explorer. These were the files we checked out previously.

28) Select Undo Check Out from the SourceSafe menu.



✓ *A dialog box appears informing you that OPAcctView has changed and asking whether you wish to undo the check out and lose all changes.*



29) Select the Yes All command button.

Using SourceSafe to Check the Program In

Under normal circumstances, after you add a function to a program and test the program, you will want to check the program back in by using Visual SourceSafe.

Use the skills you learned in Exercise 1 to do so.

This completes the exercise.

Exercise

4

Editing and Compiling Data Services

Overview

This exercise steps you through the process of editing and compiling Pro-Biller data services. It draws upon skills obtained in previous exercises and provides instruction for performing the following tasks:

- Editing ProBiller data services from within Developer Studio
- Logging on to the UNIX server
- Copying data services to the UNIX server
- Compiling data services on the UNIX server
- Editing data services directly on the UNIX server

Approximate Time for Completion

Three hours

Tools for Completion

- Computer set up with Microsoft Developer Studio and SourceSafe
- User name and ID number for Developer Studio; user name and password for the Sierra UNIX server
- This handout

Editing Data Services

You can edit data services either from Microsoft Developer Studio or using the UNIX-based editor *vi* on the UNIX server. We will first use Developer Studio so that we can build on skills learned in previous exercises.

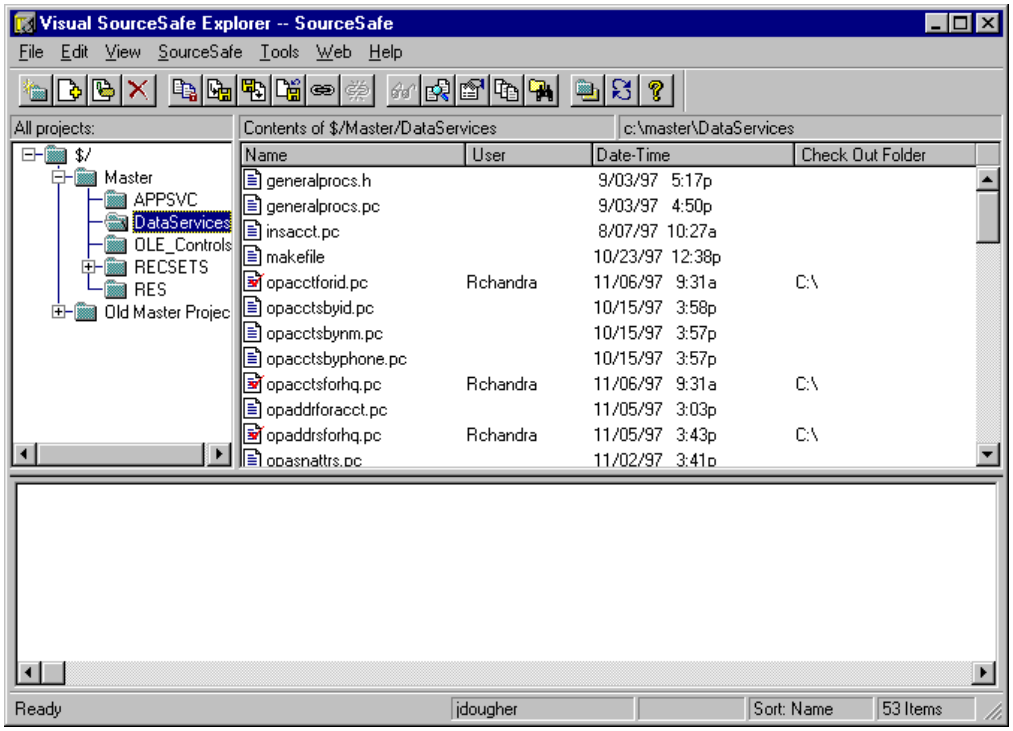
Checking Out the Data Service

1) Open Visual SourceSafe.

📖 *The ProBiller database should appear by default, since it was the last one opened.*

2) Expand the Master directory tree by clicking once on the plus sign ⊞ to the left of the Master folder.

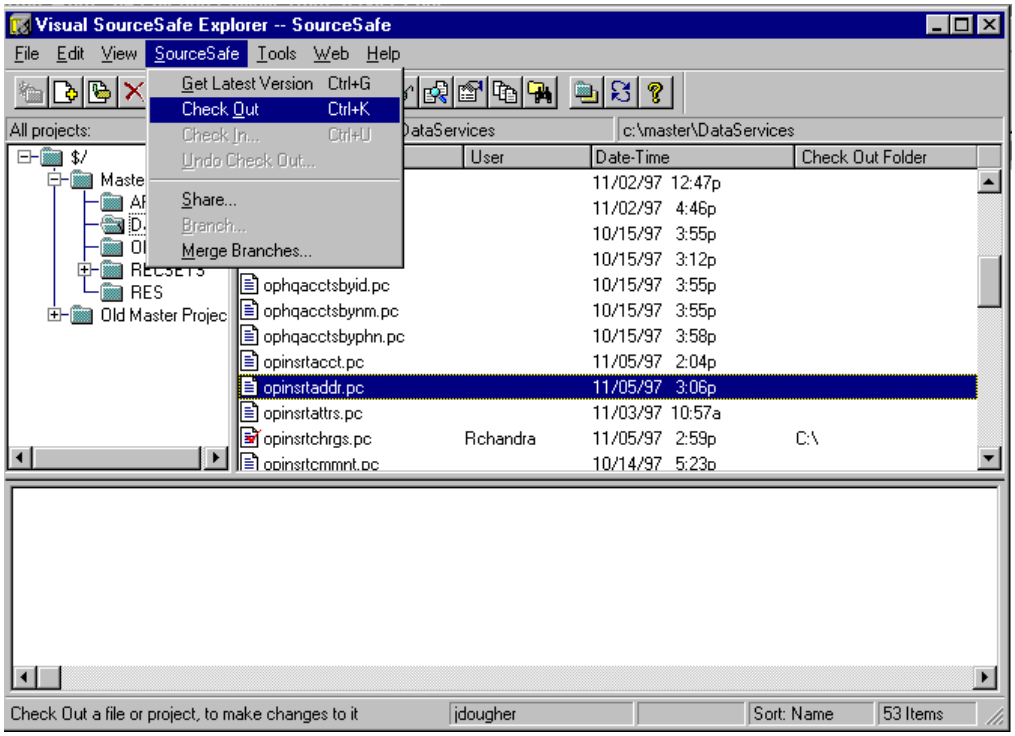
3) Select the data services folder.



✓ *The ProBiller data services appear in the upper right-hand pane of the Visual SourceSafe Explorer.*

4) Select the file you wish to check out. For this exercise, scroll down the list of data services and select **opinsrtaddr.pc**.

5) Check out the data service by selecting Check Out from the SourceSafe menu.

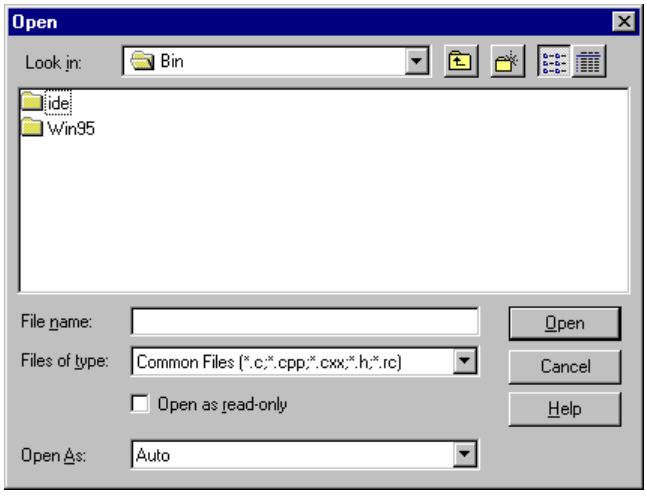


✓ *The file is checked out.*

Opening the File in MS Developer Studio

6) Open or switch to Microsoft Developer Studio.

7) Select Open from the File menu.



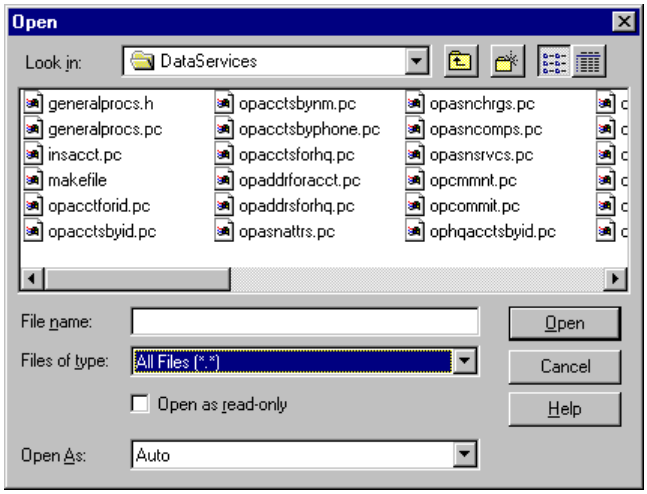
✓ *The Open dialog box appears.*

8) In the Open dialog box, select the Up One Level button  until you reach the root directory of the C drive, in which the Master folder for the ProBiller project is located.

9) Double-click the Master folder Master.

10) Double-click the DataServices folder DataServices.

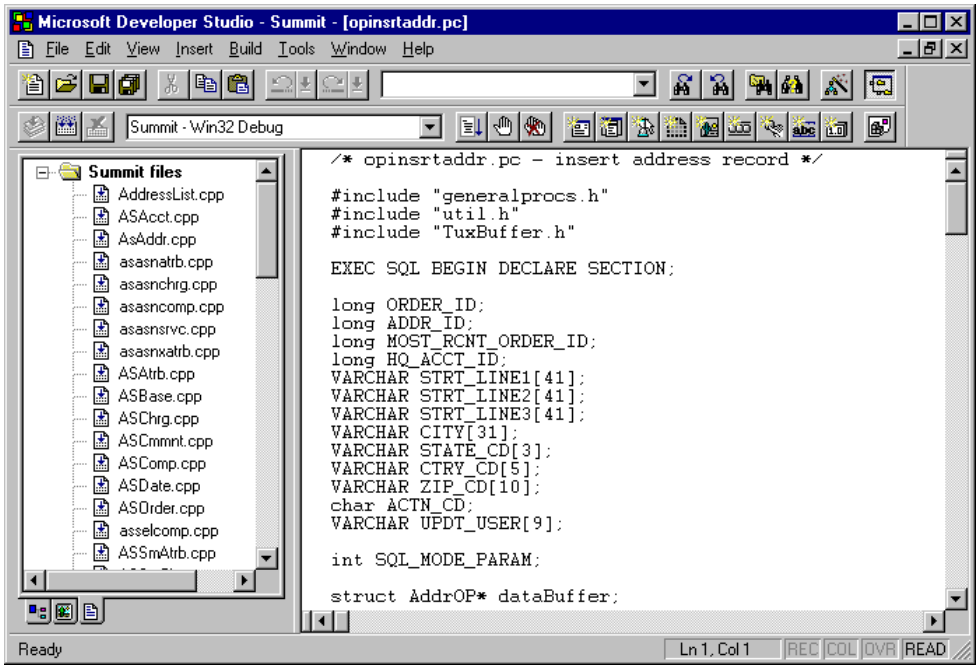
11) In the Files of Type drop-down list box within the Open dialog, select “All Files (*.*).”



Because the data services are not classes that normally show up in Developer Studio, it’s necessary to choose the “All Files (.*)” option under file type to display them.*

12) From the file list, select the data service you wish to edit. For the purposes of this exercise, choose **opinsrtaddr.pc**.

13) Select Open.



✓ *The file opens in Developer Studio.*

14) At this point, you may edit the data service. For this exercise, move to the end of the file before the final brace, and add the following comment:

`userlog("end of service %s",rqst->name);`

Your code should look similar to that in the figure below.

```
        }*/
        EXEC SQL COMMIT;
        if (SQL_MODE_PARAM == INSERT)
            dataBuffer[0].errorCode = INSERT_SUCCESS;
        else
            dataBuffer[0].errorCode = UPDATE_SUCCESS;
        tpreturn(TPSUCCESS, 0, (char *)dataBuffer, bufSize, 0);

        userlog("end of service %s",rqst->name);
    }
```

15) Choose Save from the File menu in Developer Studio.

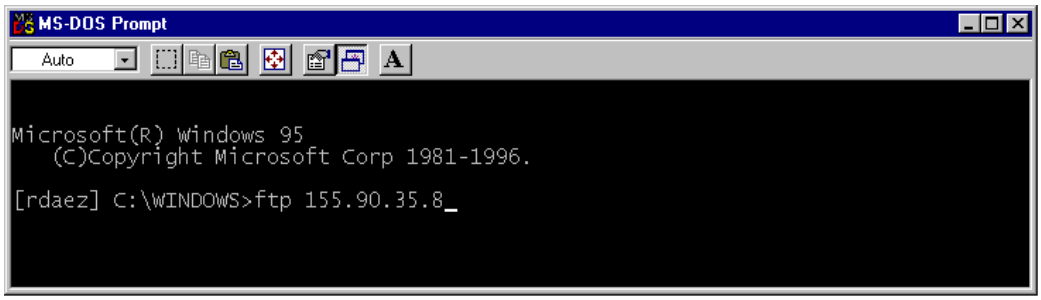
✓ *The data service is saved.*

Copying the File to the UNIX Server

To compile data services, we will use the File Transfer Protocol (ftp) utility to transfer the data service we just edited.

16) Open a DOS window by selecting the Windows Start button and choosing Programs, MS-DOS Prompt.

17) At the DOS prompt, type ftp 155.90.35.8.

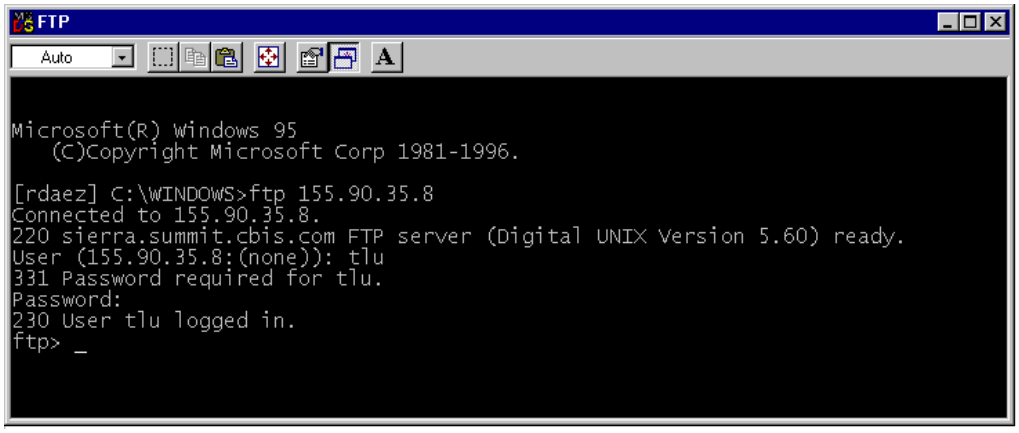


18) Press <Enter>.

✓ *The FTP utility connects you to the Sierra UNIX server and presents you with the user and password prompts.*

19) Type your user name at the user prompt, and press <Enter>.

20) Type your password at the password prompt; press <Enter>.

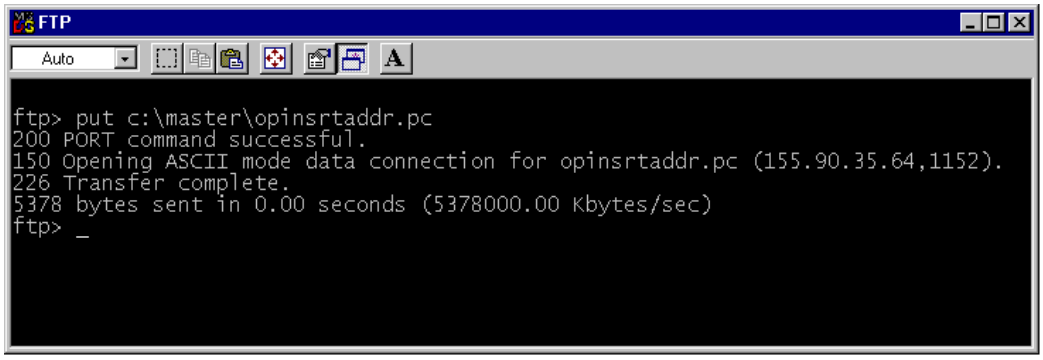


✓ *You are now logged on to the Sierra UNIX machine.*

21) Type **cd tux/summit** to change to the Tuxedo Summit directory.

22) Type **put c:\master\opinsrtaddr.pc** at the prompt to copy the opinsrtaddr.pc file from your local drive to the server.

23) Press <Enter>.



✓ *The file has been transferred to the Tuxedo ProBiller directory on the server.*

24) Type **bye** at the command prompt to exit the FTP session.

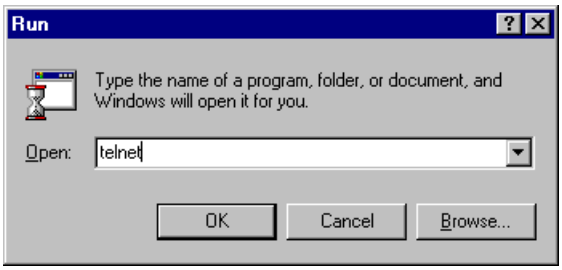
25) Press <Enter>.

Compiling the Program

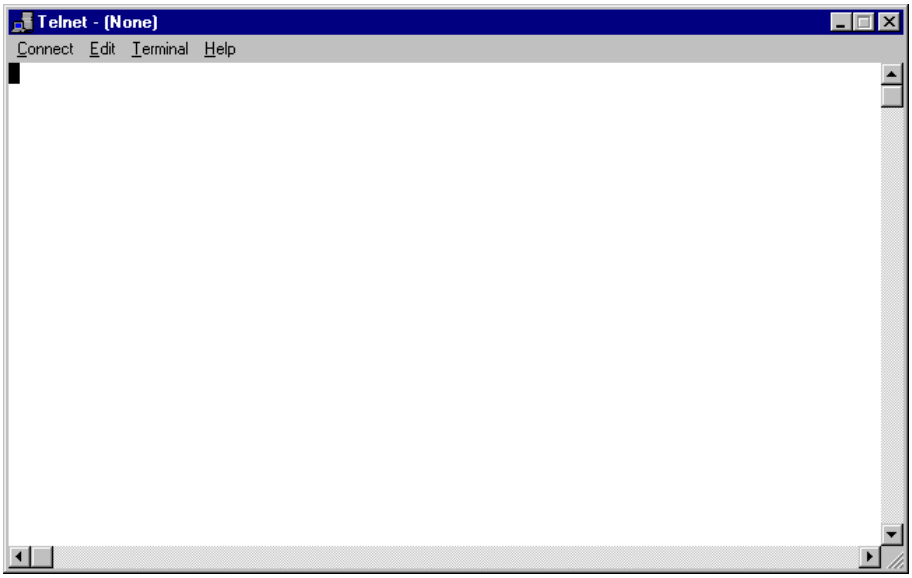
Now that we have transferred the file to the UNIX machine using the File Transfer Protocol utility, we'll log on to the server using Telnet and compile the data service.

26) Open the Windows Start menu on the taskbar, and select Run.

27) Type **telnet** into the Open text box in the Run dialog.

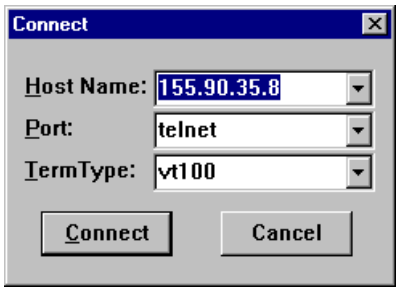


28) Select OK.



✓ *The Telnet terminal window opens.*

29) Select Remote System from the Connect menu.



30) Type **155.90.35.8** into the Host Name text box, if necessary. Make sure that **telnet** is listed in the Port text box and **vt100** is set as the Term Type.

31) Select Connect.

32) At the **login:** prompt, type your user name.

33) Press <Enter>.

34) At the **password:** prompt, type your password.

35) Press <Enter>.

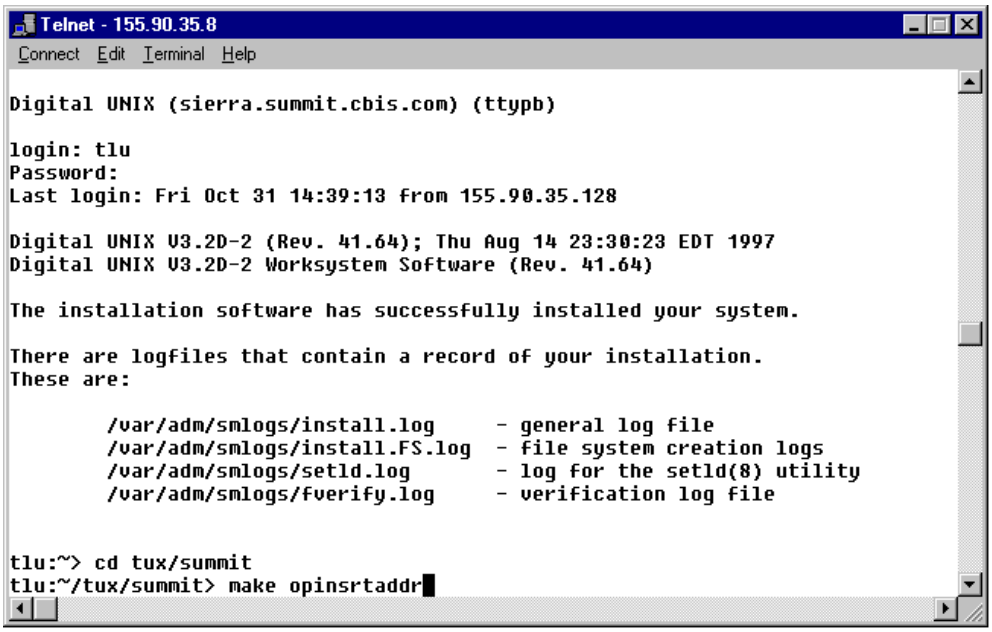
✓ *You are now logged on to the UNIX server.*

36) At the prompt, type **cd tux/summit** to change to the ProBiller directory.

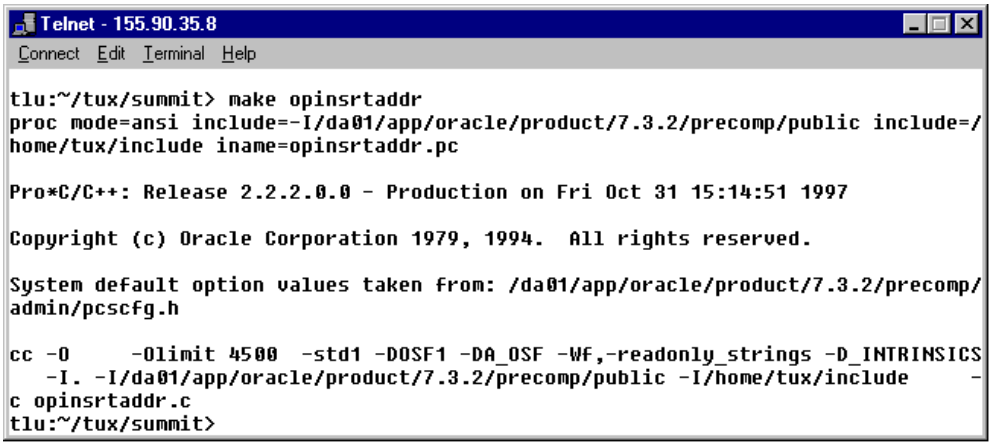
37) Press <Enter>.

38) To compile the program, issue the command **make <dataservice>**.

To compile the program you edited in this exercise, type **make opinsrtaddr**, as illustrated in the figure below.



39) Press <Enter>.



✓ *The program is compiled. Any errors in the compilation would show up at this stage. In the figure above, there are no errors.*

Activating the Edited Data Service

You have now edited, saved, and compiled a data service that already existed (rather than one you created). After you compile such a data service, you must activate it by following a five-step process:

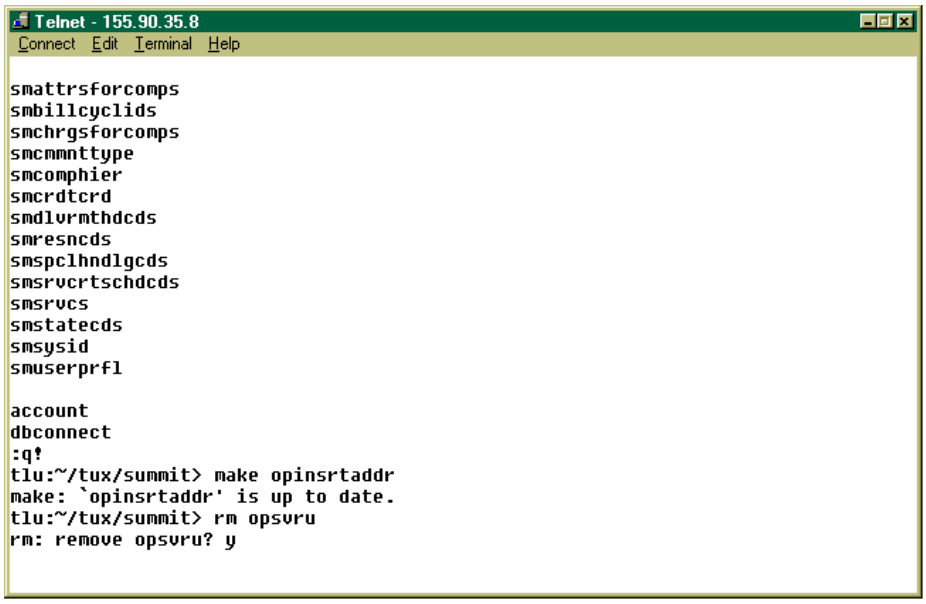
- Remove the Tuxedo server on which the data service resided
- Doing a “make” on the server you just removed
- Shut down the server
- Load the new configuration into the **ubbsummit** file
- Reboot the server

Removing the Tuxedo Server

1) Remove the server by typing **rm <servername>** at the UNIX prompt. In the example below, the server’s name is **opsvru**.

2) Press <ENTER>.

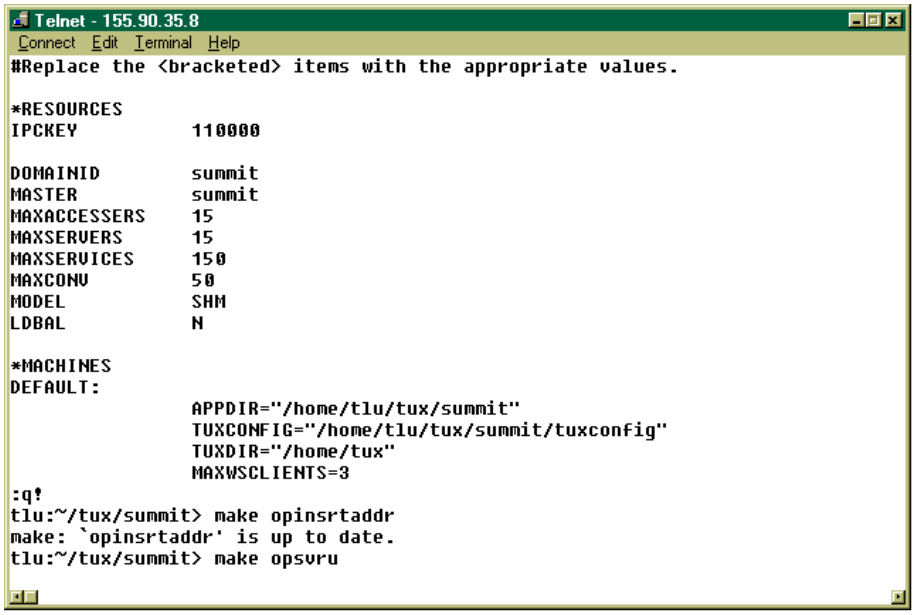
3) At the **rm: remove opsvru?** prompt, answer **y**, as in the figure below.



4) Select <Enter>.

Performing a Make on the Server

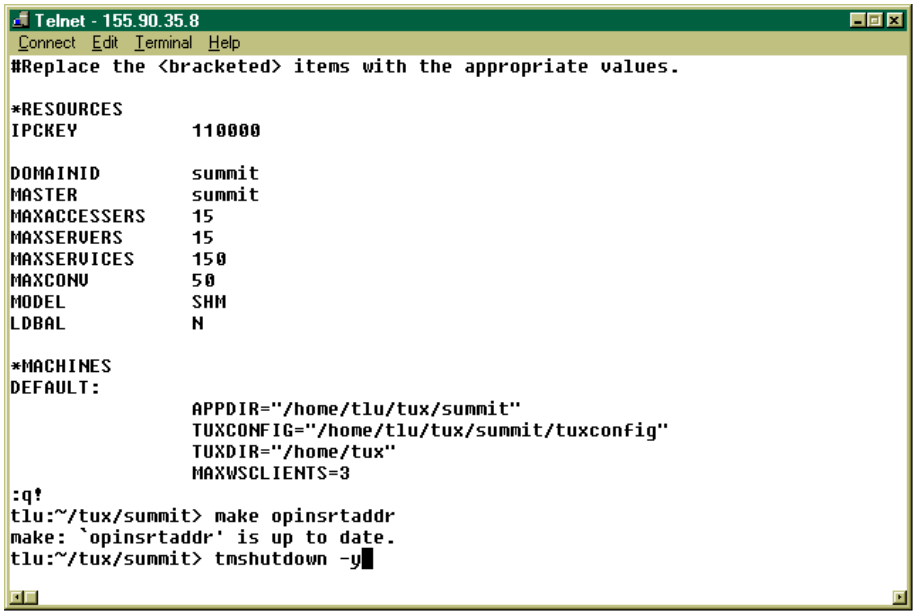
5) Now type **make <servername>** to do a make on the server. In the example below, the server's name is **opsvru**.



6) Press <ENTER>.

Shutting Down the Tuxedo Bulletin Board

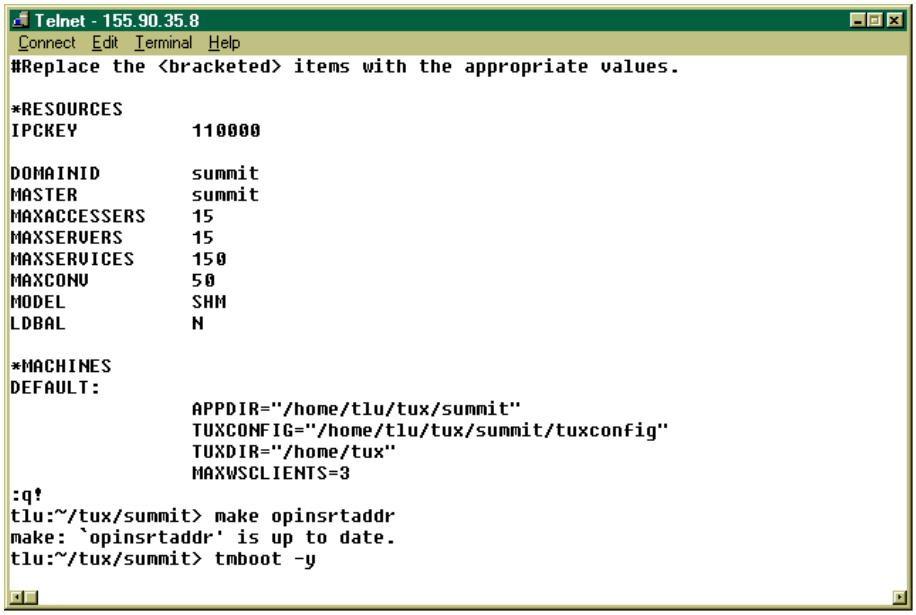
7) Shut down the Tuxedo bulletin board by typing **tmshutdown –y** at the command prompt.



8) Press <Enter>.

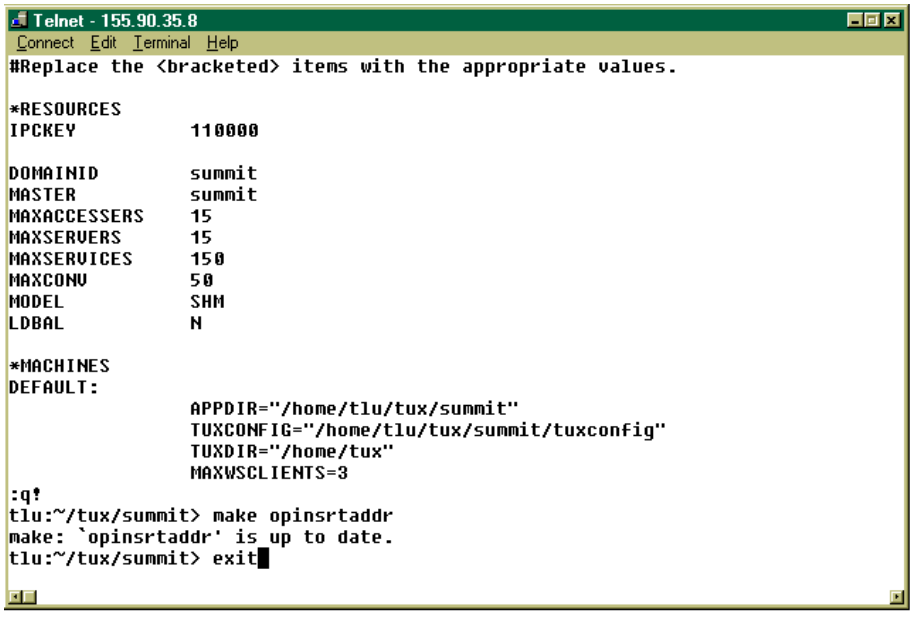
Rebooting the Tuxedo Server

9) Activate (or boot) the bulletin board by typing **tmboot –y**.



10) Press <ENTER>.

11) Exit the system by typing **exit**.



Editing Data Services from UNIX

You can also use the *vi* editor on UNIX to edit the data services directly on the UNIX server. To do so, take the following steps.

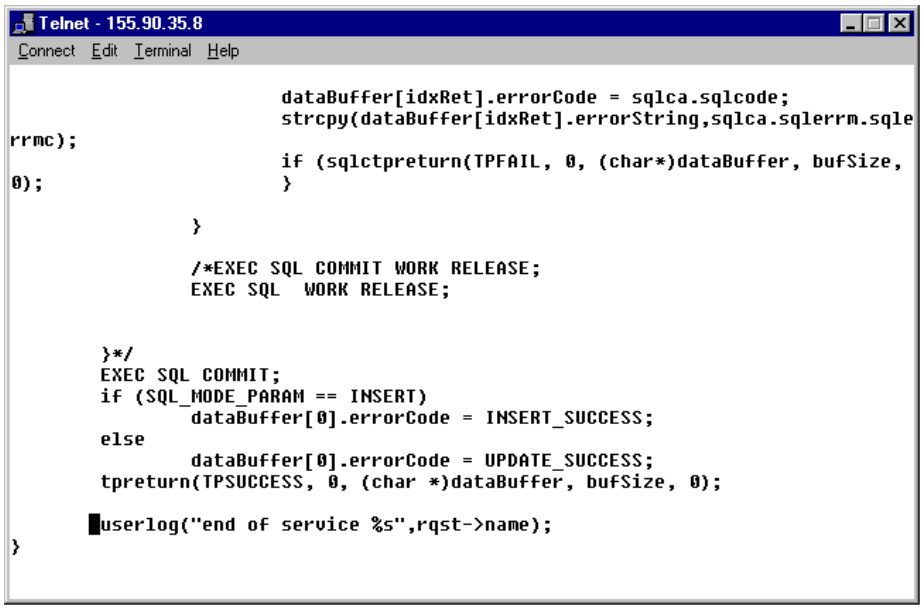
1) Use the Telnet utility to gain access to the UNIX Sierra machine.

📖 *If necessary, follow the instructions above under* ***Compiling the Program*** *to log on to UNIX.*

2) Issue the **cd tux/summit** command to change to the ProBiller directory.

3) Type **vi opinsrtaddr.pc** to edit the same file we edited earlier. The normal command line syntax is *vi* <*filename*>, where <filename> is the file you wish to edit.

4) Press <Enter>.



✓ *The* opinsrtaddr *file opens in the* vi *editor.*

5) Hit the <Esc> key, and then type <Shift>+<G> to move to the bottom of the file, where we added the line **userlog("end of service %s",rqst->name);**

📖 *For a complete list of* vi *commands, see the handout on* vi *included in the Student Handbook.*

6) Move to the beginning of the line that we added earlier, and type **dd** to delete the line.

📖 ***dd*** *is the* vi *command used to delete a line.*

7) Type **:wq** to write the changes and quit the *vi* editor.

✓ *You are returned to the UNIX command prompt.*

8) Type **exit** at the UNIX prompt to exit the system.

Exercise

5

Adding a New Data Service

Overview

This is an optional exercise for advanced users.

In the previous exercise, we edited an existing data service and then went through the process of compiling it and reloading the server on which it resides. This exercise will step you through the process of adding an entirely new data service to ProBiller. It will draw heavily upon the skills you learned in the previous exercise.

Approximate Time for Completion

One to two hours

Tools for Completion

- Computer set up with Microsoft Developer Studio and SourceSafe
- User name and ID number for Developer Studio; user name and password for the Sierra UNIX server
- This handout

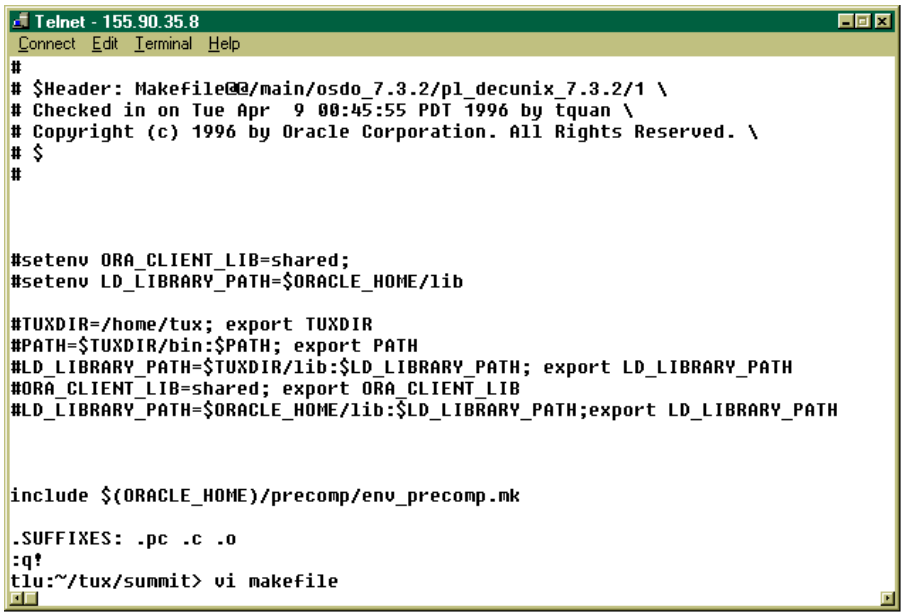
Adding a New Data Service

When you create a new data service, you must edit the makefile to add the service compile options to the makefile. In the example below, we will add the new service using the UNIX-based *vi* editor from the **tux/summit** subdirectory. To get to that directory, follow the steps outlined in the previous exercise.

Take the following steps to add the new service.

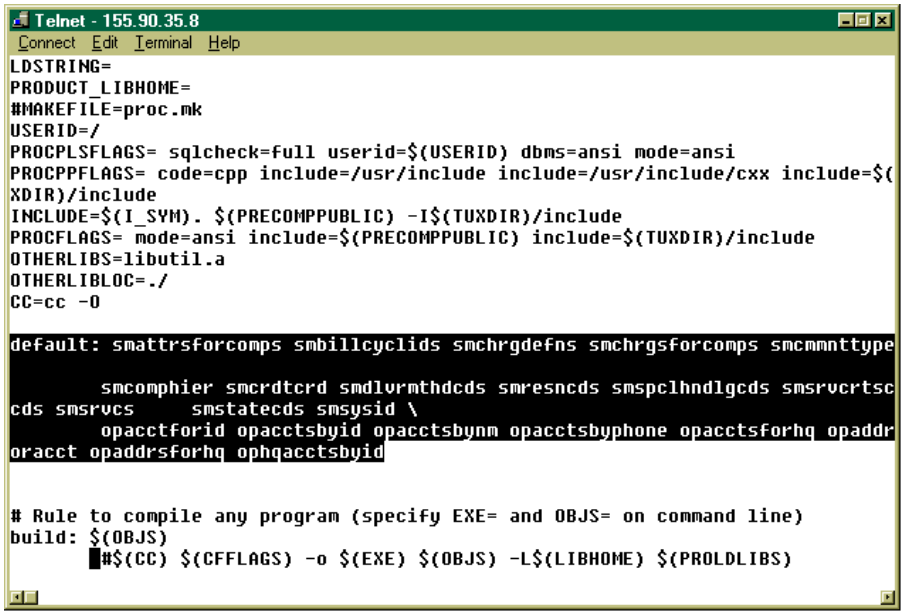
1) Log on to the UNIX server and change to the /tux/summit directory using the method you learned in the previous exercise.

2) Type **vi makefile** at the UNIX prompt.



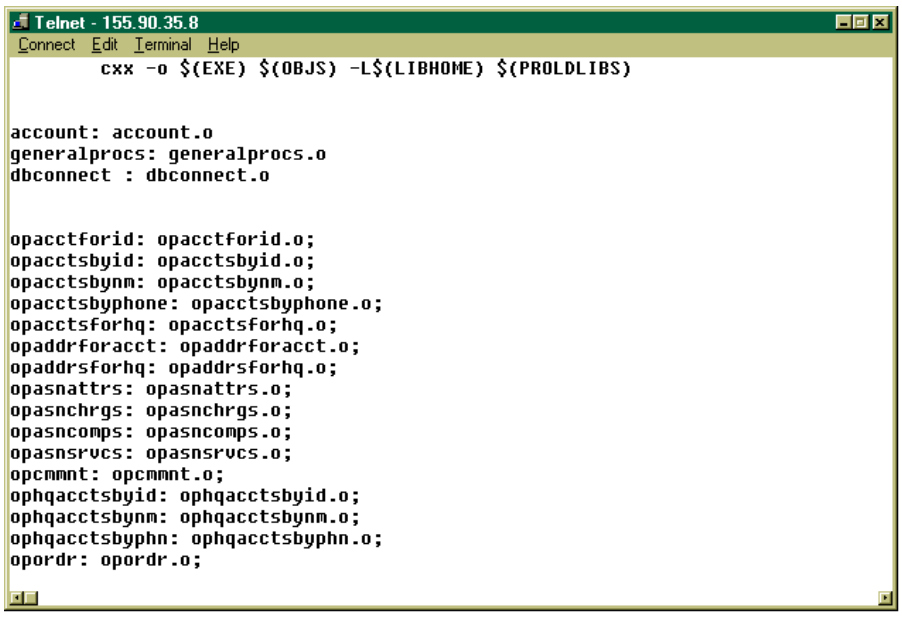
3) Press <ENTER>

4) Locate the **default:** section in the makefile, as illustrated in the following figure.



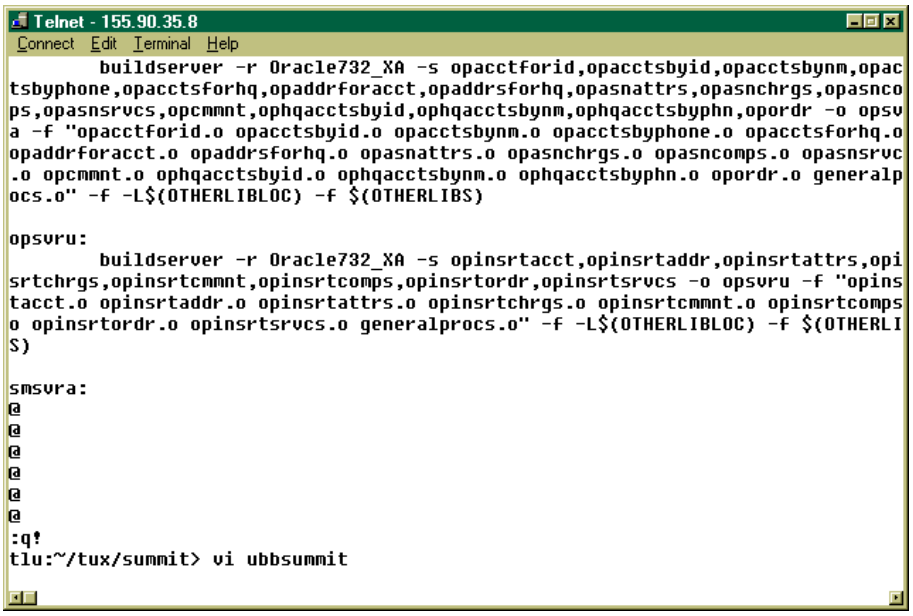
5) Add the name of the new service to the end of the **default:** section.

6) Add an entry for generating the output file for the new service using the name you provided above.



7) Type **:wq** to save changes and exit the file.

8) At the prompt, type **vi ubbsummit** to edit the **ubbsummit** file.



📖 *The **ubbsummit** file is the Tuxedo bulletin board file, which advertises all of the servers with their available services.*

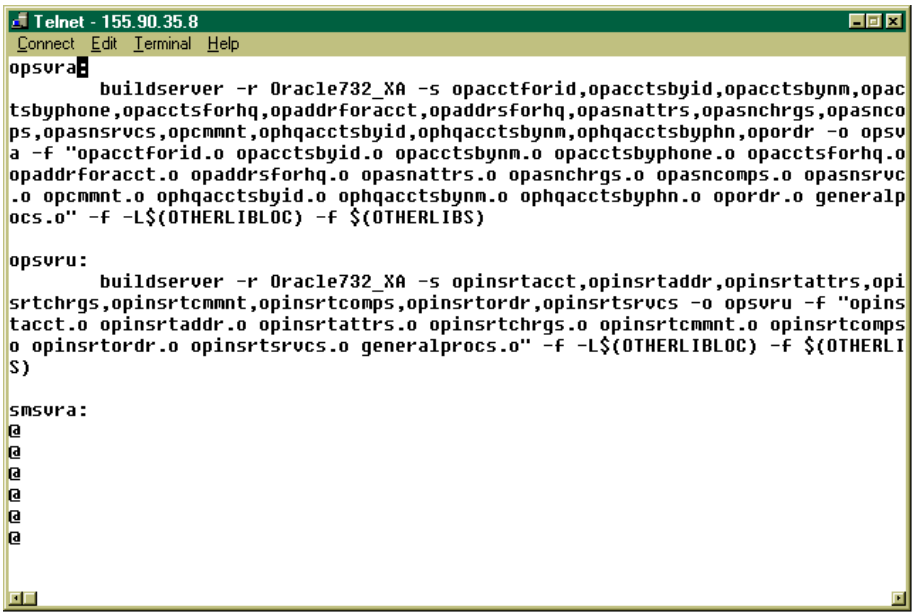
9) Press <ENTER>.

10) Scroll down to the *Services section and add the service name to the appropriate group.

11) Scroll to the server sections, as illustrated below. Locate the **–s** flag, and type the service name after that flag within the appropriate server section. Separate the service name from other services with a comma, as illustrated in the figure below.

12) Next, locate the first **–f** flag within the appropriate server section, and type the name of the object file after that flag (following the quotation mark after the **–f** flag). Separate the object file name from other object file names with a space, as illustrated in the figure below.

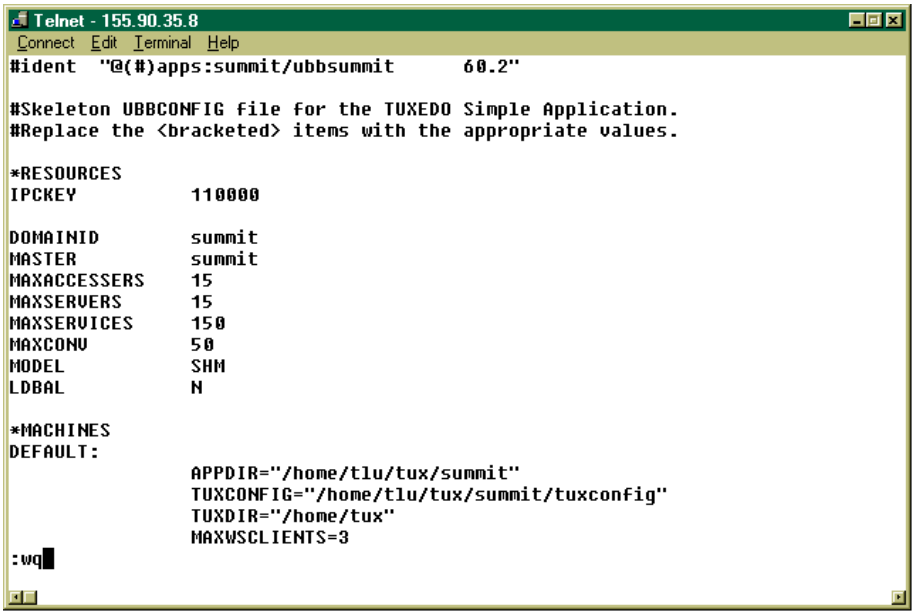
📖 *The object file name consists of the service name followed by **.o**, as illustrated below.*



13) Add your service to one of the three servers—*smsvra*, *opsvra*, or *opsvru*—listed in the figure above.

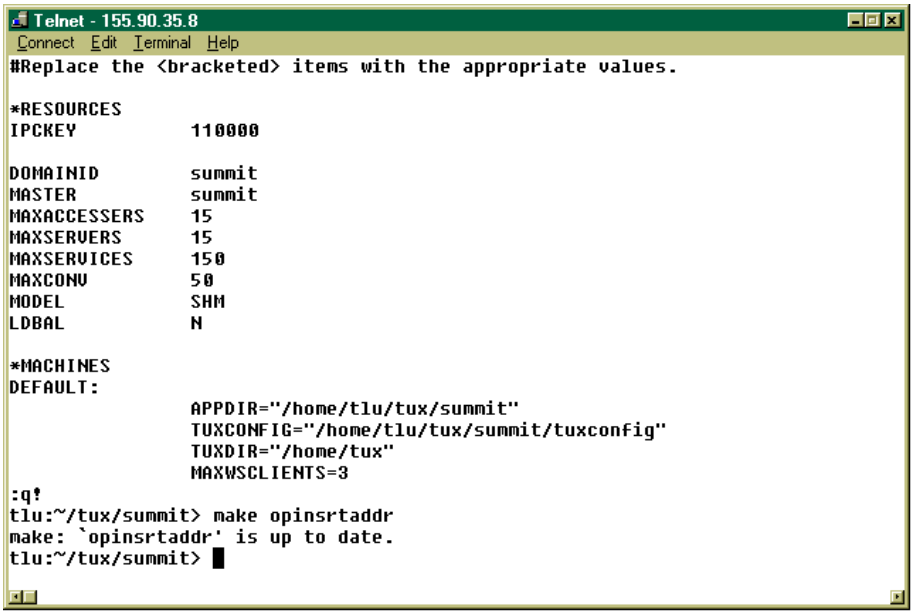
📖 *You can add the name of the new service anywhere before the final quotation toward the end of each server list.*

14) Save the changes you made to the **ubbsummit** file by typing **:wq**.



15) Press <ENTER>.

16) Type **make <servicename>** to do a make on the service. In the example below, the user's service is **opinsrtaddr**.

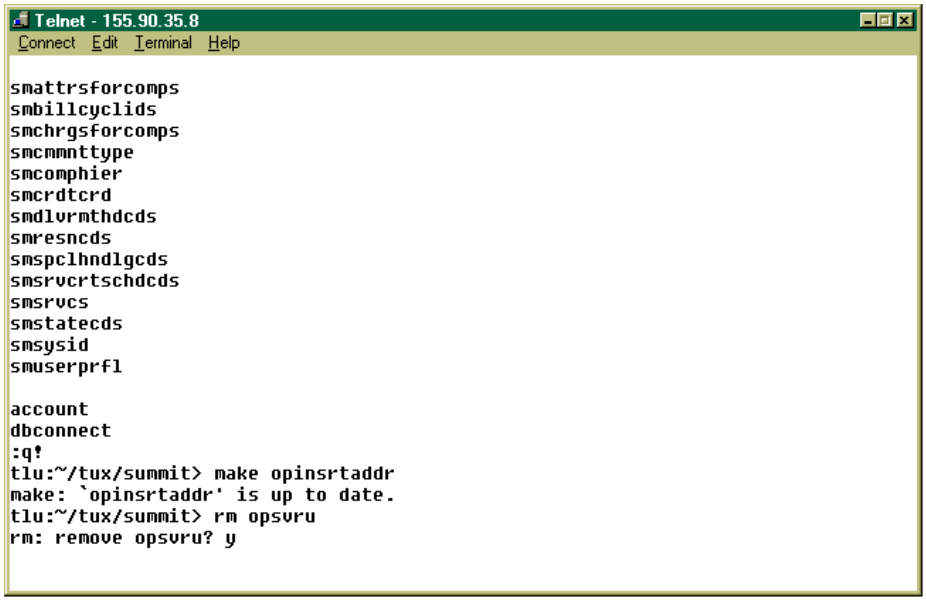


17) Press <ENTER>.

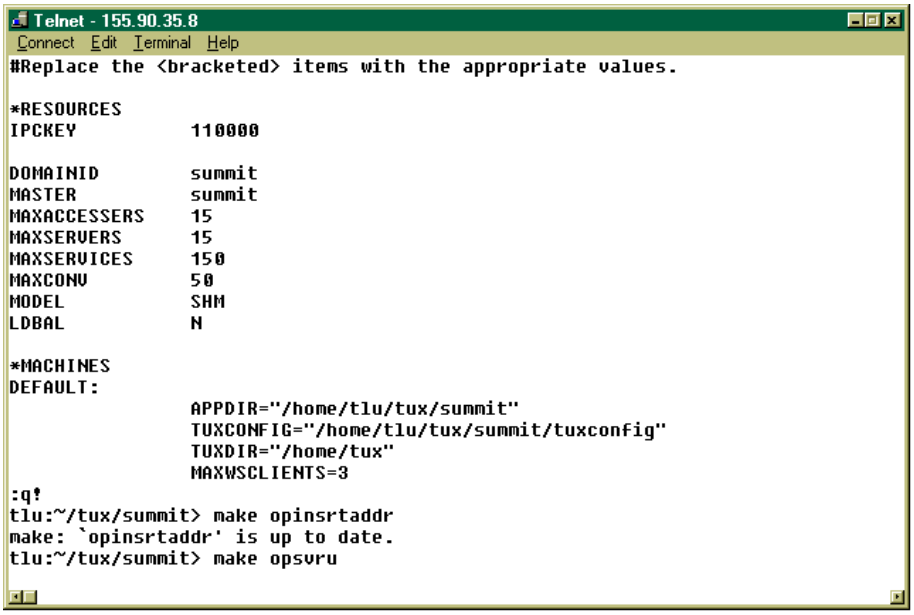
18) Remove the server by typing **rm <servername>**. In the example below, the server's name is **opsvru**.

19) Press <ENTER>.

20) At the **rm: remove opsvru** prompt, answer **y**, as in the figure below.

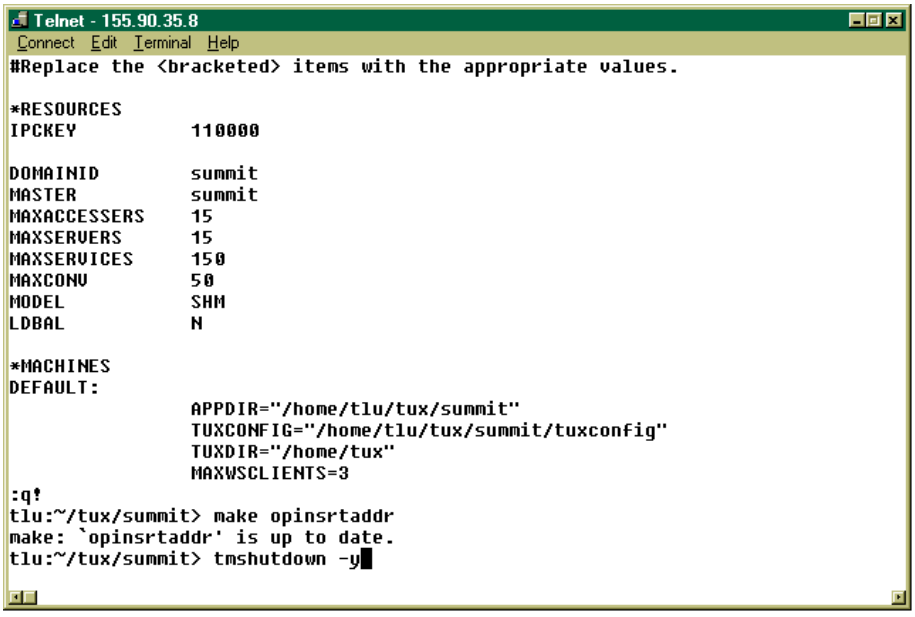


21) Type **make <servername>** to do a make on the server. In the example below, the server's name is **opsvru**.

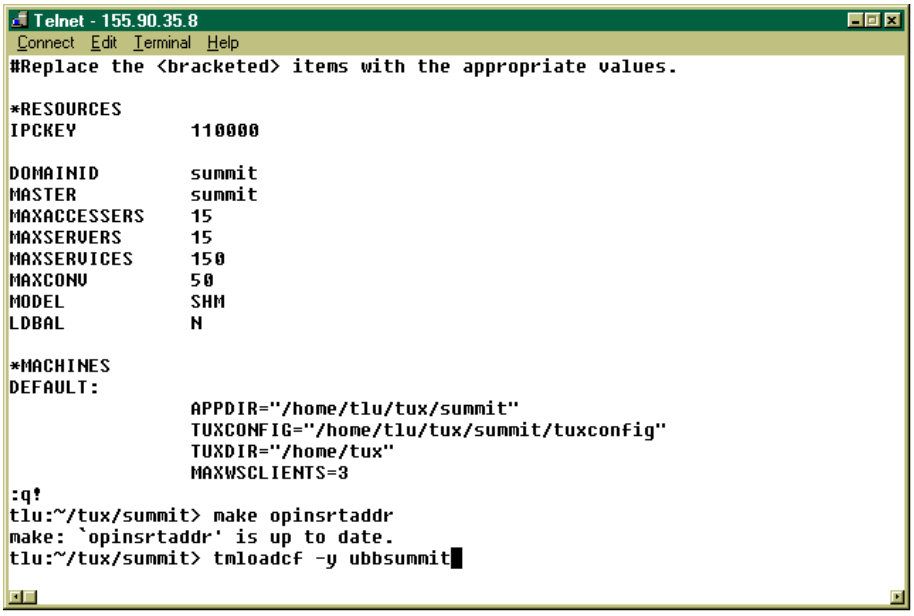


22) Press <ENTER>.

23) Shut down the bulletin board by typing **tmshutdown –y**.

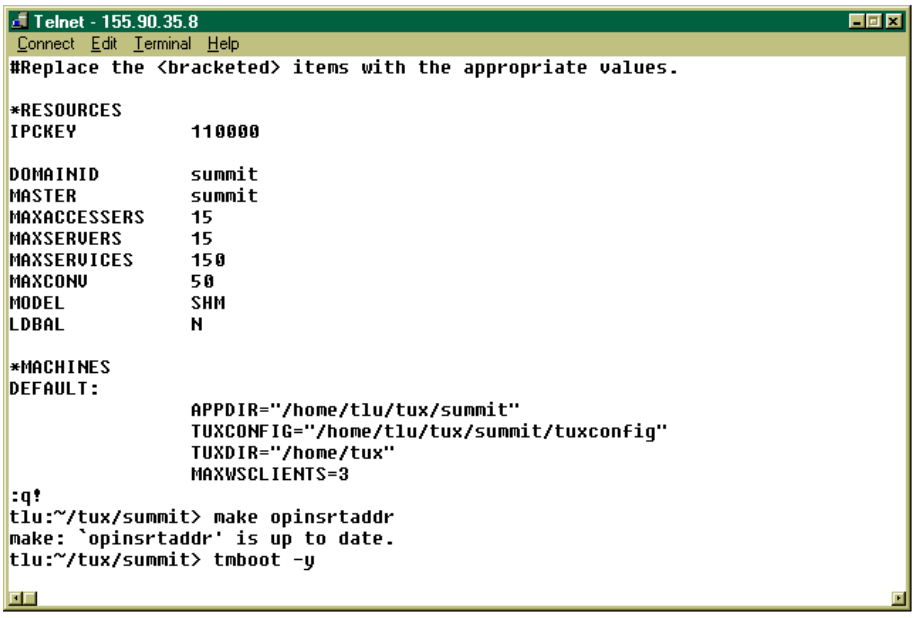


24) Load the new configuration of the bulletin board by typing **tmloadcf –y ubbsummit** so that the new service you added shows up on the bulletin board when it is booted up.



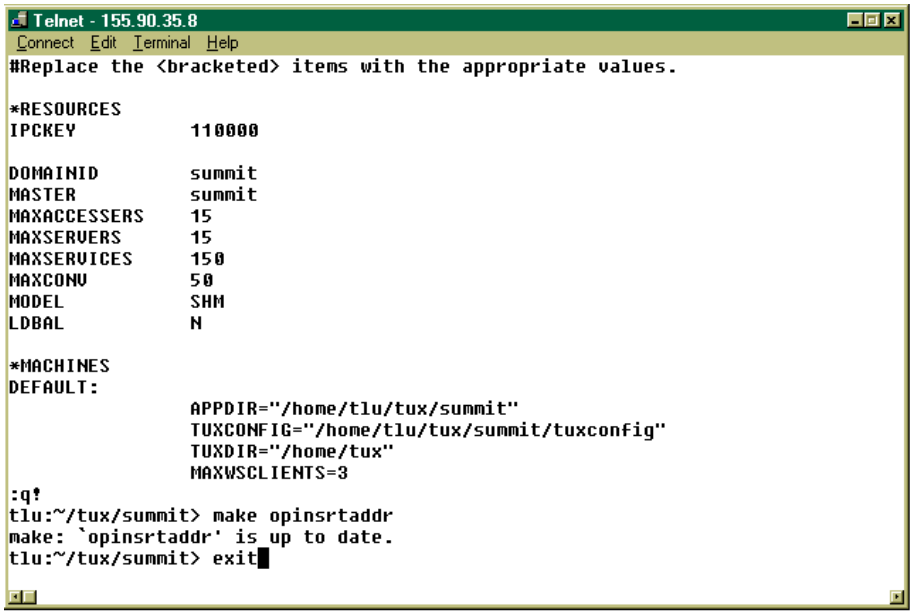
25) Press <ENTER>.

26) Activate (or boot) the bulletin board by typing **tmboot –y**.



27) Press <ENTER>.

28) Exit the system by typing **exit**.



Exercise

6

Printing Source Code

Overview

This exercise takes you through the steps of printing source code for presentation layer classes, application services, and data services.

Printing Presentation Layer Classes and Application Services

Printing source code for presentation layer classes or application services is a simple, two-step process.

1) Load the source code into Microsoft Developer Studio.

2) Select Print from the File menu.

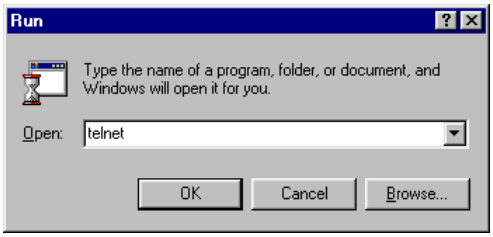
Printing Data Services Source Code

To print data services source code, you will need to log on to the UNIX server either by using Telnet or by using FTP. This exercise lets you practice using each of the methods.

Printing from the UNIX Server Using Telnet

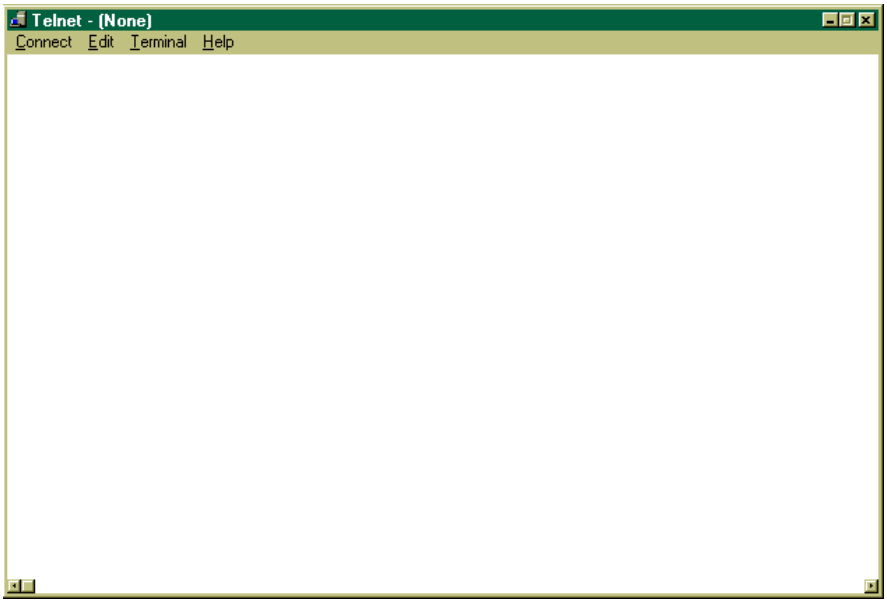
To print source code by using Telnet to log on to the UNIX server, take the following steps.

1) From your Windows desktop, select Run and type **telnet** into the input box.

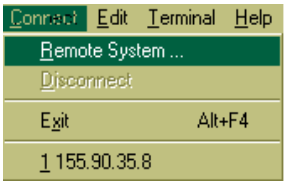


2) Select OK.

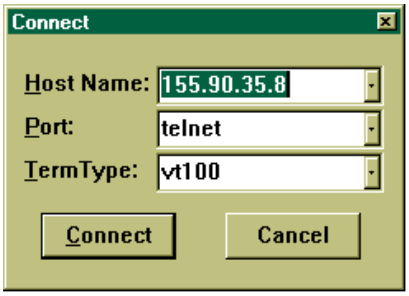
✓ *The Telnet application window opens.*



3) Select Remote System from the Connect menu.



4) When the Telnet connect window appears, type **155.90.35.8** in the Host Name text box. Ensure that **telnet** appears in the Port drop-down and **vt100** appears in the TermType drop-down.

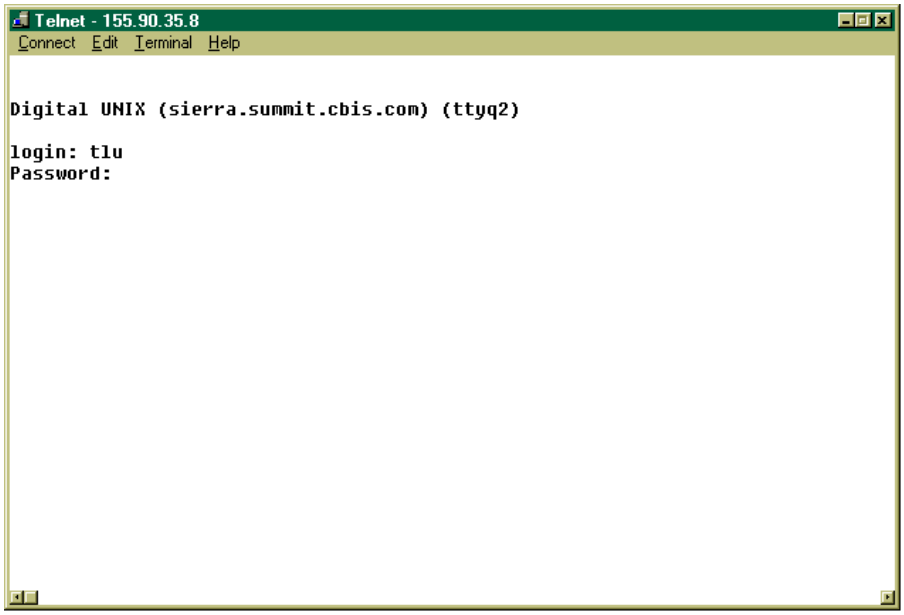


5) Select Connect.

✓ *The UNIX login screen appears.*

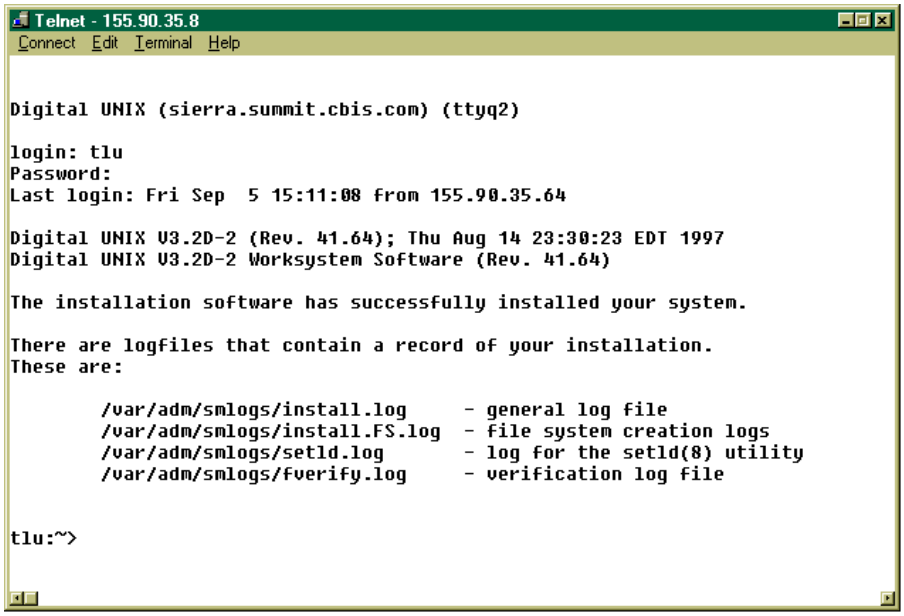
6) At the **login:** prompt, type your user name, and press <ENTER>.

✓ *The UNIX password prompt appears.*

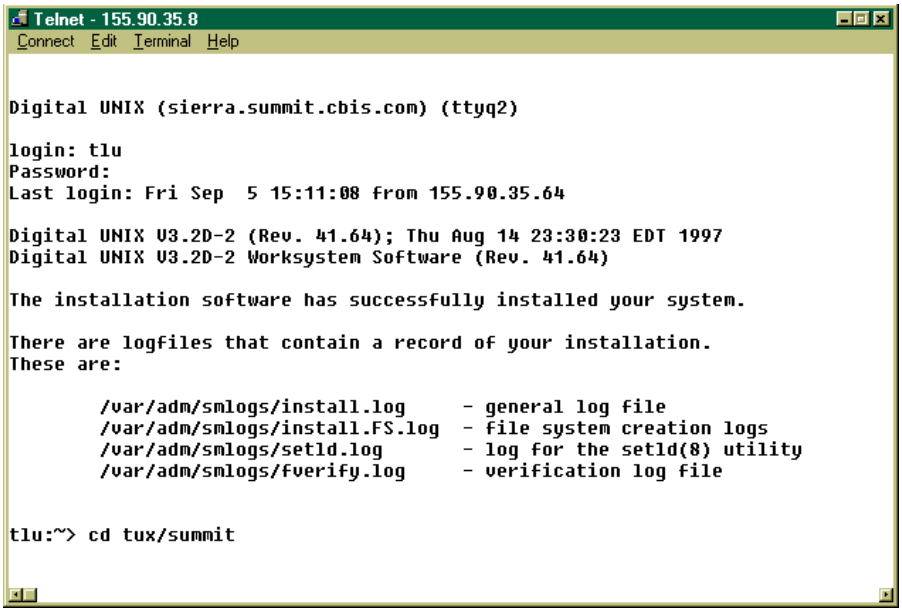


7) Type your password into the **password:** prompt, and press <ENTER>.

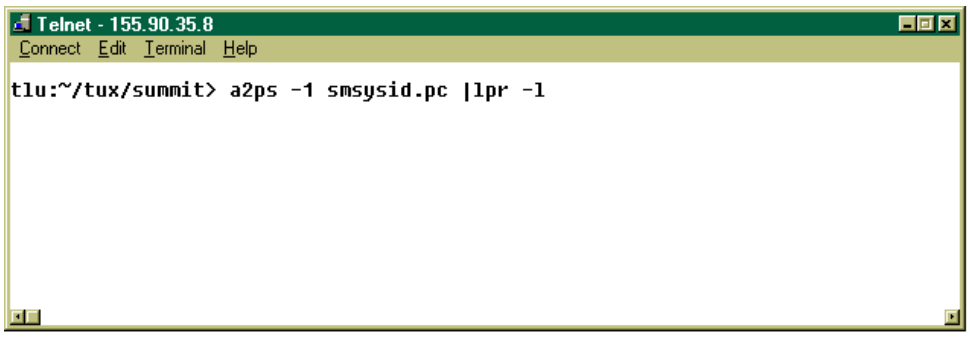
✓ *A screen similar to the following one will appear on your monitor.*



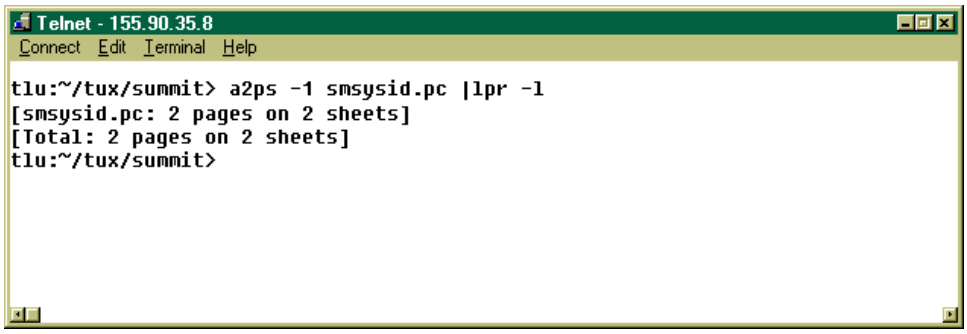
8) At the prompt, type **cd tux/summit** and press <ENTER> to change to the working directory.



9) To print, issue the command **a2ps –1 <filename> |lpr –l**, where **<filename>** is the name of the file you wish to print. In the following figure, the user is printing a file called **smsysid.pc**.



10) Press <ENTER>, and the system will return status lines similar to those in the figure below.



11) To log off of the UNIX server, type **quit** at the command prompt.

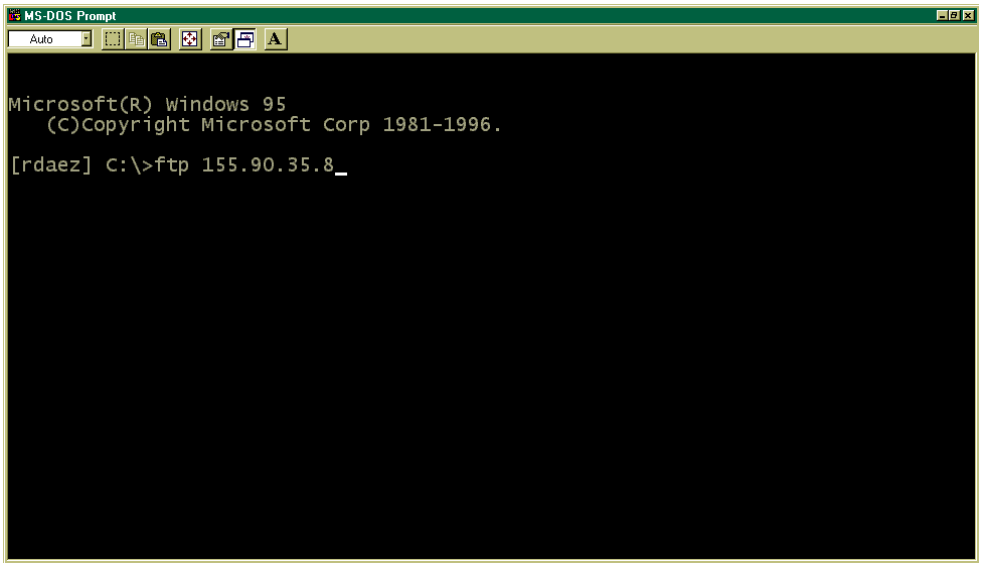
12) Choose Exit from the Telnet Connect menu to quit the program.

Printing from the UNIX Server Using FTP

You can also use the FTP utility to log on to the UNIX server, retrieve the file onto your local system, and print the source code from there using, for example, Microsoft Developer Studio. To do so, follow these steps.

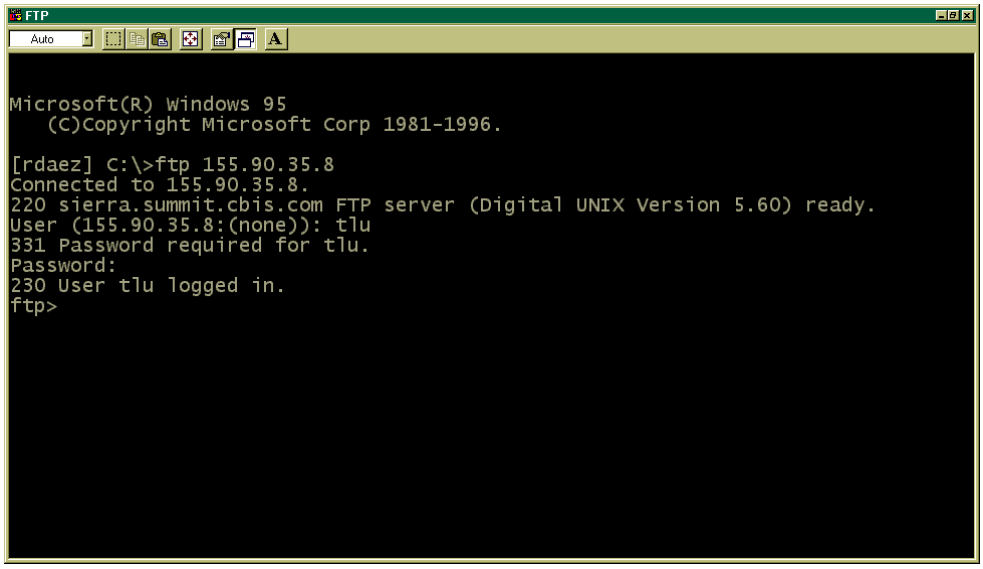
1) Open an MS-DOS command prompt from Windows by selecting Start, Programs, MS-DOS Prompt.

2) At the MS-DOS command prompt, type ftp 155.90.35.8, as illustrated in the figure below.

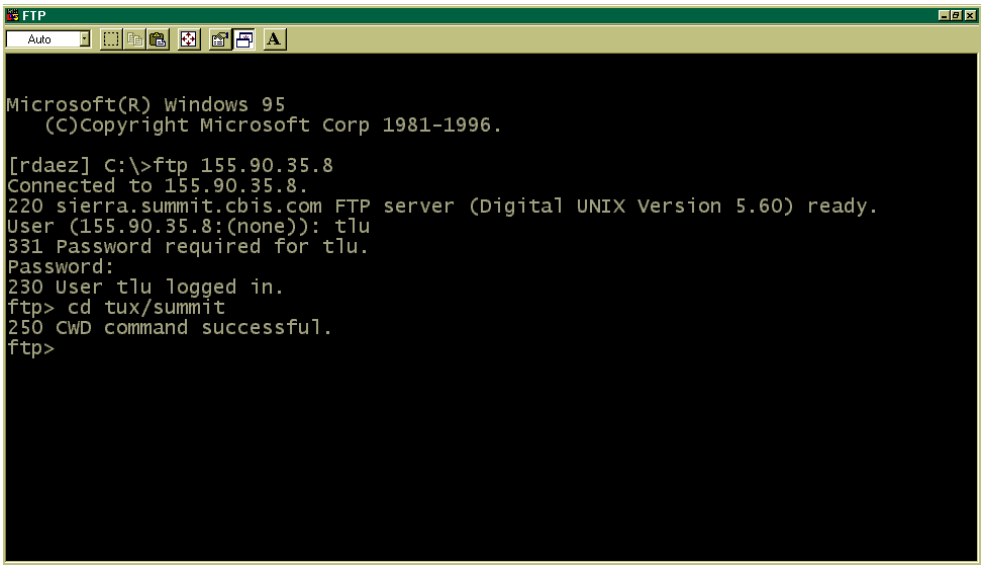


3) Press <ENTER>

4) Enter your user name and password into the prompts that follow, and press <ENTER>. If successful, you will see a screen similar to the following.



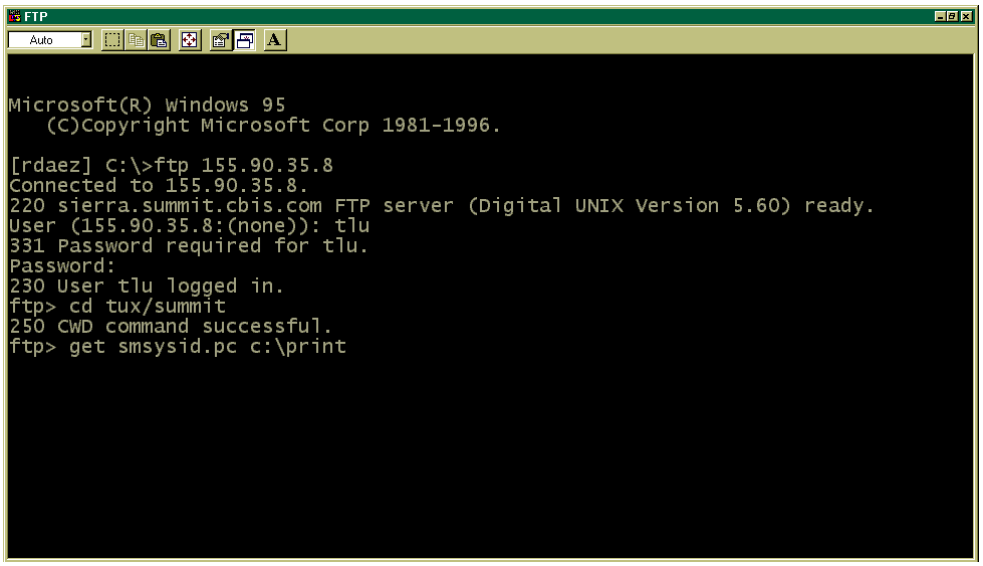
5) Type **cd tux/summit** at the UNIX command prompt to change to the ProBiller directory.



6) Issue the UNIX **Get** command to retrieve the file you wish to print onto your local hard drive. The syntax for the **Get** command is

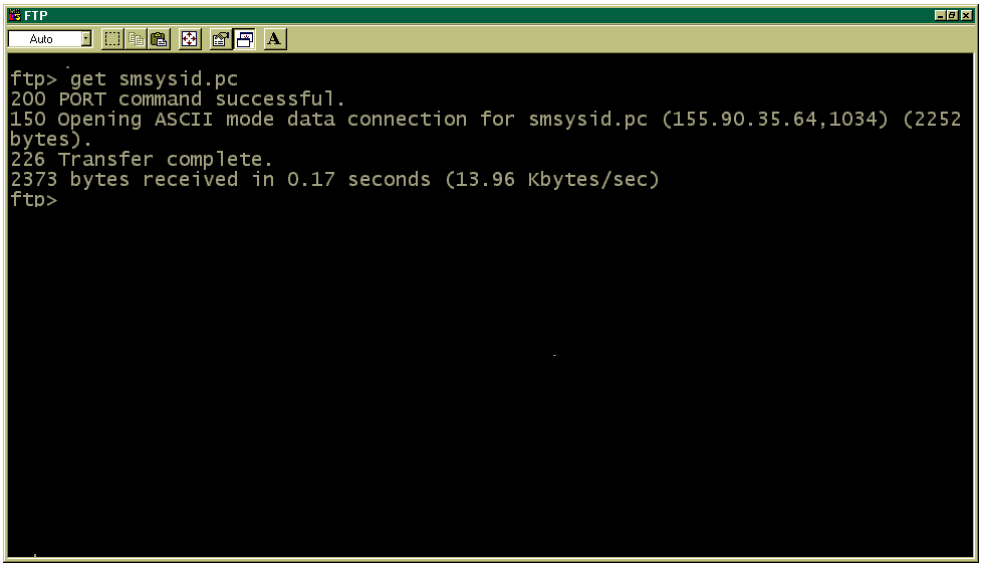
- **get <sourcefile> <destination drive and directory>**

  In the figure below, the user has entered a **Get** command to retrieve the file **smsysid.pc** to the local directory **c:\print**.



7) Press <Enter>.

✓ *UNIX transfers the file to your local hard drive.*



8) Quit the FTP session by typing **quit** at the FTP prompt.

9) Print the file using any appropriate application.