Chapter 2 SourceSafe in Theory and Practice

So far, Chapter 1, "Visual SourceSafe Installation," helped you install Visual SourceSafe and configure it. This chapter looks at what SourceSafe does for you—how it stores information, how it presents your projects, and how you can check files in and out and track changes. This chapter explains what SourceSafe is doing with operations named checkout, check in, difference, branch, pin and merge. In this chapter, you get to run the Visual SourceSafe client, also called the Explorer, create a sample project and source code, and exercise the functions made available through the VSS Explorer interface. The last portion of this chapter reviews each of the dialogs presented by SourceSafe, pointing out important features and hazards.

The primary purpose of SourceSafe is to store, retrieve and track the history of multiple versions of computer files. These files could be source code for a development project, Word documents for a marketing campaign, or CAD drawings for a building renovation. The contents are generally not important to SourceSafe (although later on, the difference between binary and text files is discussed); SourceSafe's job is to track the documents and their changes over time. This chapter discusses the interface of the SourceSafe Explorer and, in the process of doing that, all of the functionality available within the product.

Many source code control systems are available on the market, either commercially or open-sourced. But few provide the richness of interface present in SourceSafe. Since early in the product's lifetime, SourceSafe has been marketed as a "Project-Oriented Version Control System." It is that *project orientation* that has been a distinguishing factor. While other vendors have done well in adding this feature into their software, early versions typically required all files to have a unique name, or they were stored in a flat namespace, or individual databases might be required for each project.

SourceSafe allows you to store the source to many different projects within the same database. It presents these projects in a tree view, with one root to the tree (designated $/) and a group of projects off that tree. Projects can have subprojects, just as disk directories can have subdirectories. With the hierarchical view of the source code (the "One Tree" of "One Tree Software"), different files on different branches could have the same name.

This chapter presents the basics of source code control, first from a theoretical standpoint, then with a practical tutorial, and finally through an exhaustive review of the SourceSafe Explorer interface.

The branching tree structure introduces a challenge as well as a benefit. Because each project's source can have its unique location, the challenge is to find a way to share source between the branches, and to control which versions of files are shared. That's covered in the section "The reality: Swinging through the branches."

Finally, there's a whole bunch of additional terms—journaling, archive, restore and shadow directories—that fall more into the realm of administrative functions than day-to-day operations. Those are covered in Chapter 7, "Administration."

The theory: Learning to climb the tree

SourceSafe's primary function is storing files and their differences. That function needs some supporting structure in order to be very useful. SourceSafe starts with the concept of a project, elegantly displayed in the SourceSafe Explorer as a folder. And what goes in a folder? Well, stuff. SourceSafe leaves the decision up to you. When using SourceSafe integrated with other tools, each project folder typically holds a set of files that create a single product, whether a Web site or a VB application. Folders within the folder map to subdirectories of the "root" project directory. However, when you are using SourceSafe directly from the Explorer interface, you can organize files however you like. A root folder can hold all of the client proposal letters. Folders within this folder can hold the additional work you did for clients accepting your proposals. You are in charge. Folders can go within folders, in a nested fashion (no one folder can be in two folders, although the contents can be shared, as covered later). Any kind of files can go in the folders—text, bitmap, word processing, database tables, XML—limited only by the capacities of the SourceSafe storage engine.

The folder metaphor can break down pretty quickly here, so let's not try to stretch it too far. Just as if you were to store a folder about a project in your file cabinet, a project folder should contain all of the files and folders needed to rebuild a single project. But the folder can be nested to any level, so you might choose to organize your top level as the individual clients you work for, the year that you create a particular project, or any other scheme that works for you. In the real world, if we wanted to reuse some forms from one folder in another, we'd photocopy them. SourceSafe gives us a couple of capabilities to do that, but we're peeking ahead here. Let's cover the basics first.

Imagine your company has an office manager in charge of the files. If anyone wants to use a file, they need to talk to the office manager and check out that file with the manager. "Checking out" is a process where you get a copy of the latest file from the office manager, and you get the right to make changes to it. The office manager (SourceSafe) *always has the master version of the file.* If anyone else is looking for that file, the office manager knows who has the file checked out. Others are free to view the contents of the files and folders. Under some circumstances, they might be able to check out the same files that are already checked out. If you only wanted to look at the contents of a file, you wouldn't need to check it out, only talk to the office manager and look at that file. That's the SourceSafe Get function. When you are done with the file, it is checked in. The changes you have made are identified, and a history of your check-in, as with the checkout, is recorded. When you are checking in a file, you are updating the SourceSafe master copy of the file.

Let's run the software and see what a real situation looks like.

Tutorial: Climbing the tree

There's nothing like learning by doing. In this section, you get to operate the application and can start to get some sense of how SourceSafe does what it does. You get to create a project, add a file to it, check out and modify the file, and examine the history that SourceSafe generates as it tracks your actions. Later on, this chapter exhaustively examines every nook

and cranny of the interface. For now, try out SourceSafe a bit so you can see how the elements fit together.

Start SourceSafe by locating the SourceSafe 6.0 item on the Start menu. If it was installed as a stand-alone product, or as part of Microsoft Office Developer, the Microsoft Visual SourceSafe submenu should be located directly off the Programs menu. If it was installed as part of Visual Studio 6.0, the submenu should be located under the Microsoft Visual Studio 6.0 menu item. Up comes the Explorer interface (see **Figure 7**, on page 27).

Highlight the top of the Project tree view by clicking on the topmost node of the tree view in the left panel, labeled $/. Create a new project by selecting the leftmost toolbar button (its ToolTip should say "Create Project") or by selecting the Create Project option from the File menu. Give the project the name of Demo and a comment that this is the sample project from this book. Click OK, and a new folder is added to the tree view. Click once on the Demo folder to make it the current project.

Let's create a file to add into SourceSafe. From the Start menu, select Run and type:

```
NOTEPAD c:\temp\demo.txt
```

(Select an appropriate drive and directory if you don't have a C:\temp directory.) Notepad appears. You'll be asked if you want to create the file. Click Yes. Let's type a quick little "program" in Notepad:

```
REM A Sample Program for SourceSafe
PRINT "Hello, World!"
```

Exit Notepad and save the file. Back in SourceSafe, select the second toolbar button, Add Files, or the File menu option of the same name. Navigate to the C: drive and temp directory. Select the demo.txt file and press the Add button. You'll be prompted with a dialog (see **Figure 11**, on page 35) where you can add a comment explaining what this file is—a very good idea. Add a comment that this is the demo file for the SourceSafe book, and select OK to add the file. A second dialog appears, asking whether you want to set C:\temp as the working directory for this project. A working directory is just the default folder where SourceSafe assumes you want files read and written to. You can override that later, but for now, select Yes. Note that the demo.txt file disappears from the Add File dialog, as it only shows files not yet added. Select Close to close the Add File dialog.

The demo.txt file appears in the right-hand panel file list, with the date and time the file was added to SourceSafe. If you examine the demo.txt file on disk, using the Windows Explorer and the Properties option, you will see that the file has been flagged as Read-Only. This is a reminder that you are on the honor system and should not make changes to the file without checking it out from SourceSafe, but you can read the file and use it to build your application, just not change it. The copy on disk is referred to as the *local* copy; the copy in SourceSafe is the *master* copy. Don't confuse this with *local* vs. *remote* referring to files on a network (your local copy may be on a network share); the local copy in SourceSafe is the copy of the file checked out for use by one user.

Next, check the file out and make changes to it. In the SourceSafe interface, right-click on the demo.txt file and select Edit from the context menu (see **Figure 9**, on page 32) that appears. A confirmation dialog appears (see **Figure 26**, on page 44) to confirm that you want to check

this file out and edit it. Select OK and the file appears in Notepad. Toggle back to SourceSafe for a moment, and you can see that the file appears in the SourceSafe interface with a red check mark on it. (If the file has a red outline around the icon, it means it is checked out exclusively.)

Toggle back to Notepad and make changes to the document. When SourceSafe checked the document out, it took the latest version of the file, stored in the SourceSafe database, and updated the version on your machine. It also switched the flags so that the file is now Read-Write. Modify the first line and insert a second line to the program so that it reads:

```
REM A Sample Program for SourceSafe, Changed Once
PRINT "Hello, SourceSafe!"
PRINT "Hello, World!"
```

Exit Notepad and save the file. Save your changes in SourceSafe by checking in the file. Select Check In from the right-mouse-click context menu, the toolbar or the application's menu, or type the keyboard shortcut Ctrl-U. In the Check In dialog that appears (see **Figure 33**, on page 50), add a comment indicating the changes you have made, and the reasons for them. Press the OK button to check the file back into SourceSafe.

That's it! You've now mastered the key functions of SourceSafe: Check Out, Modify and Check In. Let's look at a couple more items before we call it a day.

There is nothing worse than trying to remember days, weeks or months later exactly what changes you made to a file. This is one place where those comments you made along the way come in handy.

Right-mouse-click on the demo.txt file and select "Show History." (If a dialog appears asking for labels and From and To dates, just click OK.) The History dialog (see **Figure 39**, on page 54) appears, showing the changes made to the file, in a compact format. Select both version 1 and 2 from the list by using an extended selection of Shift-Click or Ctrl-Click until both versions are highlighted, and then press the "Diff" button. (Again, if a dialog like **Figure 42**, on page 56, appears, just click OK.) A Difference window appears (like **Figure 43**, on page 57) that shows you the difference between the two versions of the file. A key at the bottom of the form explains the color coding. Here is where you can see what changes you made between different versions. Close the Difference window. Click on the Report button. In the dialog that comes up, select both the Details and Differences check boxes, and then click the Preview button. Here is the report you could print, or copy to the clipboard, showing the changes that you made to the code.

To clean up from this tutorial, close the Preview window, cancel the Report dialog, and close the History form. Exit SourceSafe with the File | Exit option or by closing the main window.

In this quick tutorial, you have seen how projects can be created in SourceSafe, how files are added, and how a file is checked out, modified and checked back into SourceSafe. You have seen how to determine differences between different file versions, how to run a report comparing those files, and how to examine the history behind a file. These are the key functions of SourceSafe.

The devil is in the details, of course. While the example in the tutorial was intentionally simple, your applications are likely to involve dozens, if not hundreds of files, and a number of users checking files in and out.

The reality: Swinging through the branches

Like many products shown off at a vendor's dog-and-pony show, SourceSafe looks pretty easy to use when we run through the simple tutorial in the previous section. The challenge comes as we get into large, complex, inter-related projects. SourceSafe has the muscle to handle these as well.

One common situation is the need to share files between projects. Common "library" code or classes are created with the express purpose of being used over and over again, and one of the advantages of working with a common framework of code is that fixes to the framework can be made available to each of the projects.

There are many techniques for sharing code between projects, and each has its advantages, and its liabilities. This section presents three solutions, each with its advantages and drawbacks. The first does not take advantage of SourceSafe's complex versioning features. While it is simpler to administer from the SourceSafe point of view, it can lead to confusing situations. The second technique, using SourceSafe's sharing, branching and merging functions, is a little more complex to set up and manage, but is a more robust solution. The third is simple to administer, but requires more attention as each project is revisited. Which of the techniques you choose depends on your needs, the type of development that you do, and the resources that you have available to administer source code control maintenance.

Here's the scenario each of these techniques solves, with varied success:

You invest in a framework for building your application, let's say the Acme Framework. You use Acme to build applications for clients A, B and C. Each client gets their own customized forms and reports on top of the framework, but, due to the customizability of the framework, none requires changes to the framework itself. You freeze and ship the code for each client in turn. Between clients A and B you find a bug in the framework and recode a program that fixes it. Between clients B and C a new minor version of the framework, with bug fixes, comes out that you install over the older framework and use for client C.

Solution 1: Install the framework software as a "sibling" on the project tree to each project for clients A, B and C. Install the software on the development machines as sibling directories and use relative paths in your development tools to point to the framework software in the parallel directory.

Solution 1 works well for the original development efforts, but, later on, when you return to the projects and get the latest version of all of the software, the client A and B software would appear to be using the new framework, installed after those applications shipped. If you were trying to reproduce a problem reported by a client, you could not reproduce the source code used to develop the application without manually going through the framework and getting earlier versions of the software. Using the label functionality would simplify some of this, but you still will feel like you are swimming upstream.

Solution 2: Install the framework as a sibling application again, but then use the Share function (as explained later in this chapter) to share the framework files into each of the client's projects. Use labels and the Pinning functionality (explained later, in the "Show History" section) so that each client's project reflects the version of files they have. If necessary, branch off a client's files if they need different functionality from the framework than other clients. (By the way, this is a good indication that the framework might need some refactoring—ideally, a framework should support different behaviors in most cases.)

Solution 2 solves the limitations in Solution 1, in that each client's project will reflect the proper version of files that were actually used to build that client's application. However, the cost of this accuracy is the need for more care and vigilance in setting up and maintaining the project. This solution requires that the developers understand the principles of sharing source code in SourceSafe, and that they exercise care in working with the application. In many shops, this might be asking too much of the developers. In this case, a single point of contact might be designated as the source code control person, to ensure these changes are performed correctly. Or a single person could have rights to the shared code, controlling all of the changes to avoid cross-project problems. This could be a good use of security settings, too, as a safety net, where only certain developers might be granted access to change shared framework code.

Solution 3: This brute-force method is the simplest. Copy all of the files required, framework and custom code, into each client's project. Each client has their own copy of everything they need, and there is no interaction between the projects.

While this may seem like a crude solution to the problem, it does face the realities that in many shops there is not the expertise to maintain either of the other solutions, and that, in many cases, client developments are one-shot projects where each revision may require such a radical change in frameworks (say, moving from version 1.02 to 3.0) that trying to maintain an intricate web of shares and branches may be more effort than it is worth.

Multiple checkouts and the Dreaded Visual Merge form

In Chapter 1, "Visual SourceSafe Installation," in the section on using the Administrator tool to configure SourceSafe for integration, I recommended that you enable multiple checkouts. This setting needs some explanation, some cautions and more details.

Enabling multiple checkouts allows SourceSafe users to check out a file that is already marked as checked out, *only* if the file is a text file. SourceSafe determines whether a file is text or binary by reading a portion of the file, searching for binary zeros within the body of the file. Files can also be designated as binary by using the File Types tab of the Administrator's Tools | Options dialog. SourceSafe allows text files to be checked out by more than one user because SourceSafe has the ability to merge multiple changes to a text file into a coherent merged file. In most cases, SourceSafe will do this automatically. If you check out a file and modify line 100 at the same time that a co-worker has the file checked out and modifies line 200 and adds new lines 201 through 205, SourceSafe will merge the two changes as the file is checked in, without needing any interaction with you. The only time that SourceSafe will run into trouble is if you both choose to modify the same line. In that case, SourceSafe cannot tell what the change should be, and has to leave that decision to you. I refer to this process as "The Dreaded Visual Merge," as the interface and the dire warnings from dialogs, combined with the fact that you see this form very rarely, make this an uncomfortable experience for most developers.

It's not that bad. In this section, I will walk through an example of the merge process and demonstrate how it can be handled. First, to set the scene: Two developers check out a file named slcdata.prg. The first alters line 1086 and checks the file back in. The second developer also modifies line 1086, differently, and attempts to check the file back in. Whamo! The Dreaded Visual Merge dialog (see **Figure 1**) appears.

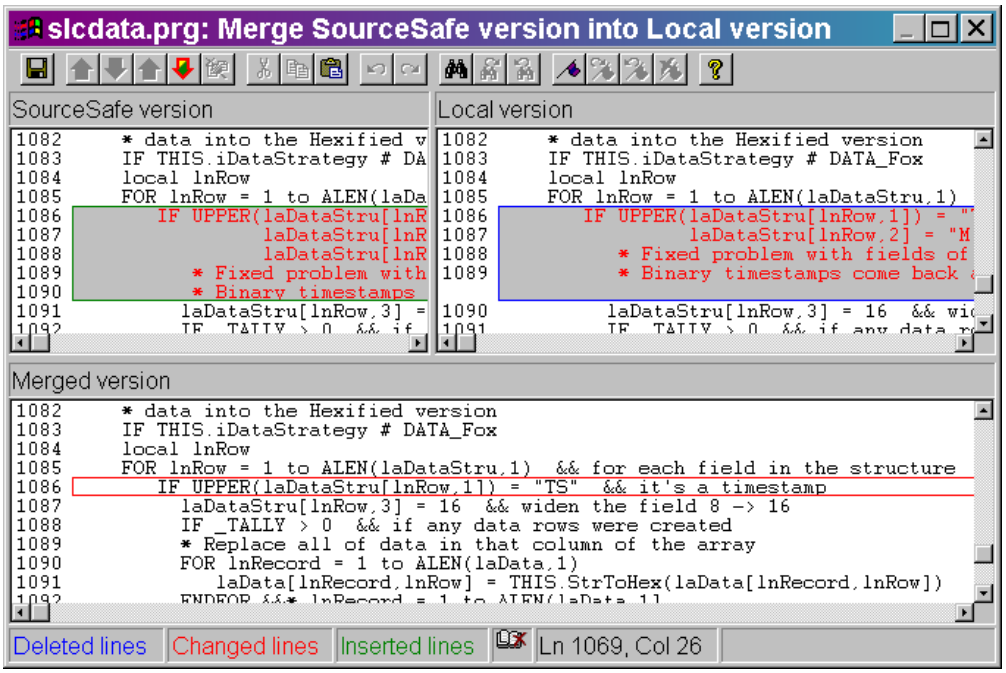

***Figure 1**. When you attempt to check in a file that had been multiply checked-out, Visual SourceSafe merges the changes when it can, and presents this dialog when there are conflicts it cannot automatically resolve.*

The Merge dialog resembles the Differences form discussed later in this chapter, with an additional pane added on the bottom. The top left pane, labeled "SourceSafe version," shows the file checked in by the other developer, with a change to lines 1086–1090. The top right pane, labeled "Local version," shows the version you have changed. The bottom pane, labeled "Merged version," shows the resulting file from the choices you make in this dialog. Note the red outlined arrow on the fifth toolbar button from the left. This shows that there is a merge conflict that has not yet been resolved. You may also see other arrows enabled, such as the second and third buttons on the toolbar, which allow you to navigate to *differences* between the two files. These differences will be resolved automatically by the SourceSafe merge engine, but can be helpful in understanding the context of the merge conflict.

You resolve the merge conflict by right-mouse-clicking on the conflict in the top left or right pane and selecting "Apply Change" from the menu that appears. This change will be reflected in the lower pane, in bold italic text, to show that you have resolved a conflict, and that the change you have selected will be saved when you are done with this dialog. When all conflicts have been resolved, you use the save button (the diskette icon on the first toolbar button) to save your changes. That's it! You're done.

If you are not able to resolve all of the conflicts—perhaps you need to confer with the other developer—save the file anyway when you have made those changes that you can make. You will be greeted with a confirmation dialog (see **Figure 2**) and—here's the key point—the file will be saved, *but not checked in*. You still have the file checked out, the other developer's

changes are safe within SourceSafe, and you have all of your changes, on disk, to use to resolve the conflicts.

After saving the file, you'll get a third dialog (see **Figure 3**) that lets you know you still have some work to do. At this point, you have a couple of choices. If it has turned out that you and the other developer have made the same changes, with some differences in syntax, you can abandon your changes by performing an Undo Checkout. If you and the other developer did different things with the same code, you've got a bigger problem than SourceSafe—you need to figure out what should be done with the code. When you've resolved that problem, you'll be ready to move on.

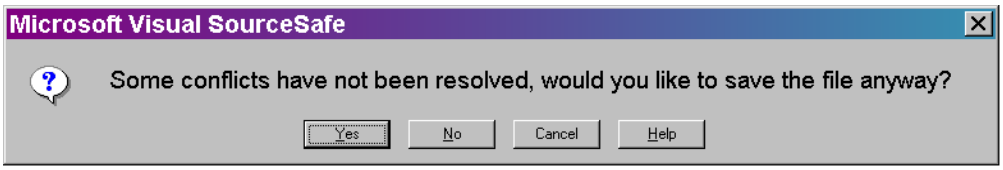

***Figure 2**. If you don't choose either the SourceSafe or local version (using the right-mouse menu) to resolve each conflict, you will be warned.*

***Figure 3**. …and warned. You can now use your editor of choice, and SourceSafe's features to display older versions of the file, to determine what changes to make.*

SourceSafe (and the integrated tools) will show you that you have the file still checked out and in need of manual changes by displaying the failed merge icon (see **Figure 4**). If you determine that you need to integrate the two sets of changes, you need to make changes to the local version of the file that you still have checked out. You can use the SourceSafe Explorer or the integrated SourceSafe menus to View History, Show Differences or View older versions of the file to determine how the resultant code should appear.

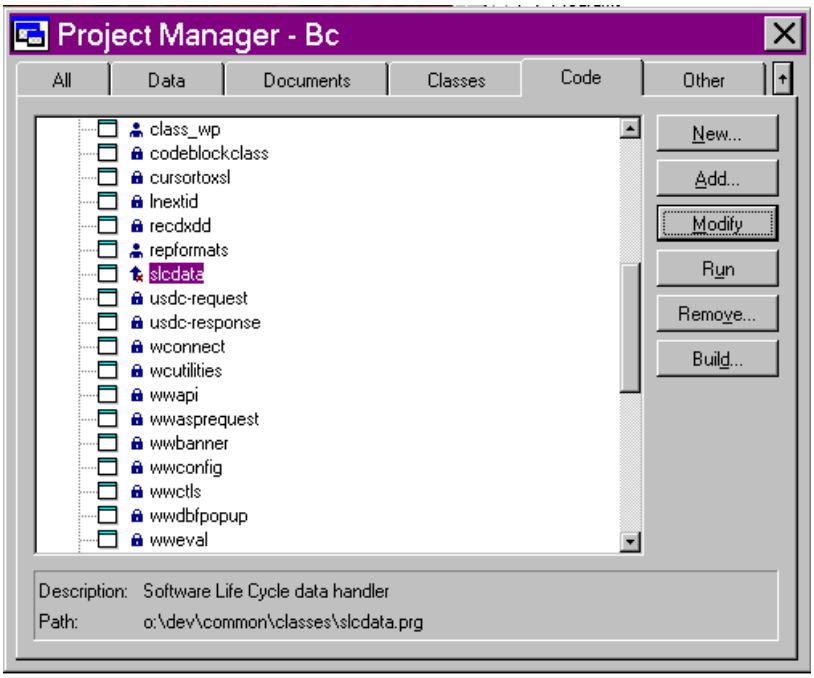

***Figure 4**. The FoxPro Project Manager shows a failed merge as a red X superimposed on a blue merge symbol.*

When you have finished manually merging all of the changes, save the file. A confirmation dialog (see **Figure 5**) will appear to verify that all conflicts have been resolved. Select Yes if you are done resolving conflicts, or No if you want to save the file as is and continue at a later time.

***Figure 5**. Be honest, now. This version of the file will be saved as the most recent version. Make sure you have manually merged the code correctly.*

When you are done resolving the merge conflicts and answer Yes to the confirmation dialog, the icon representing the file's source code control status changes to show that the merge has been completed successfully (see **Figure 6**). Check in the file to complete the merge process.

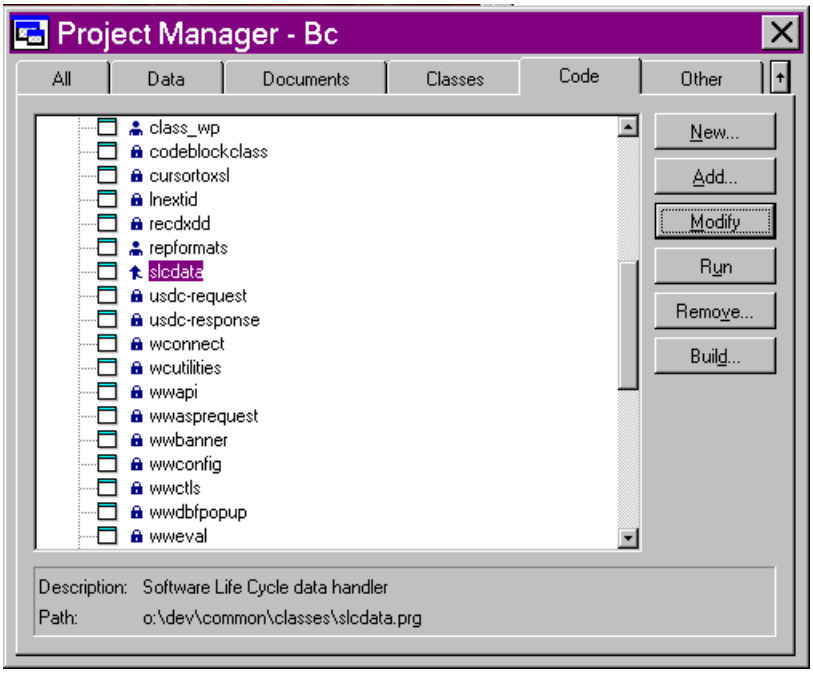

***Figure 6**. The FoxPro Project Manager displays the file with an icon showing it was merged successfully. The file is still checked out.*

Is the hassle worth it?

The first time you emerge from a battle with the Dreaded Visual Merge form, you are likely to swear "never again!" But the hassle is really not all that bad, if you are careful to read the dialogs and understand the underlying processes. There are advantages to multiple checkouts. By allowing multiple checkouts, no one developer can keep all of the files to him or herself, and the SourceSafe merge logic handles 99 percent of the merging automatically. Developers get to spend more time getting work done, and less time chasing each other to get files checked in.

Like sharing and branching, merging could be considered a fairly advanced option. If your team has less experienced members, it can be a good policy to have one member of the team designated as the SourceSafe guru, someone to call upon as a coach when the Dreaded Visual Merge form appears.

The Explorer interface

Starting Visual SourceSafe presents you with a typical three-pane Explorer interface (see **Figure 7**). The menu and toolbar are on top; a tree view listing projects is on the left. Details of the files in the current project are on the right. The bottom of the screen is occupied by the results pane and a status bar. The following sections briefly skim what's available from the parts of the interface. Because many of the functions can be invoked in three or more ways, the

details of how each function works, typically via a dialog, follow those sections with more detailed coverage.

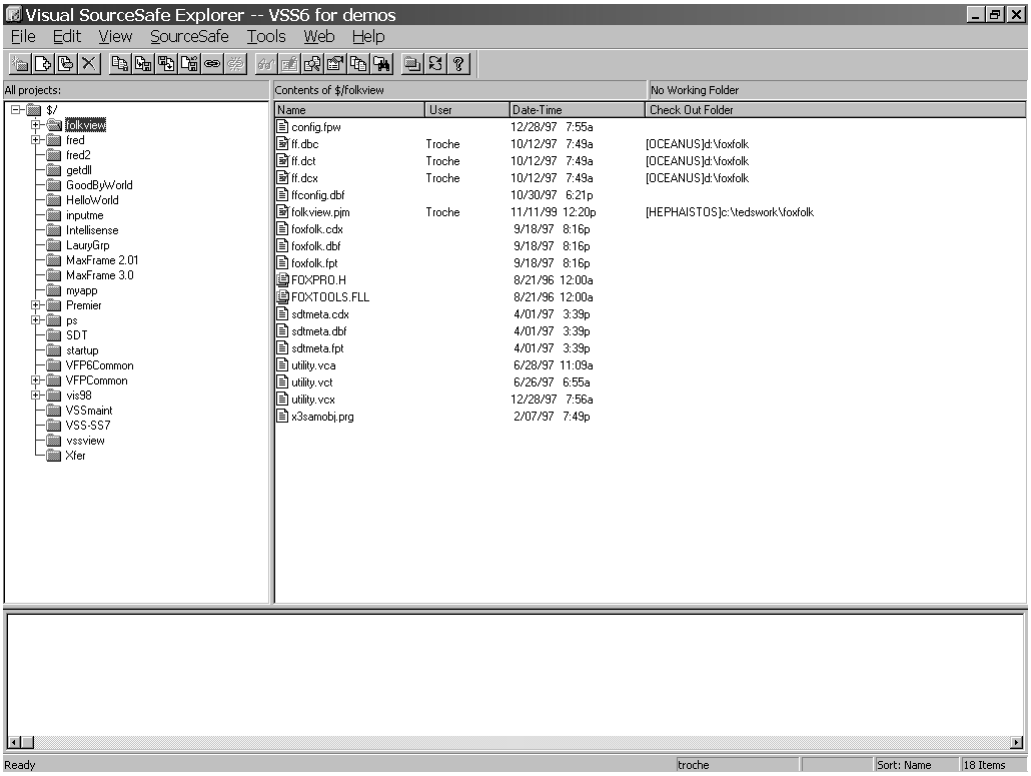

***Figure 7**. The SourceSafe interface—menu, toolbar, tree view, file list and status window.*

Toolbar

The toolbar (see **Figure 8**) is used to perform actions quickly with a single mouse click. From left to right, the toolbar contains commands to perform the following:

- **Create Project**—a new project is created under the current project.
- **Add Files**—brings up a dialog to add files to the current project.
- **Label Version**—allows you to label the current project or file.
- **Delete Files/Project**—brings up a confirmation dialog to confirm and optionally permanently delete ("Destroy") files and projects. Destroy is usually a bad idea—let the administrator purge all deleted files when you no longer need them.
- **Get Latest Version**—creates a copy of the currently selected files in the working folder.

- **Check Out Files/Project**—checks out and copies files to the working folder.
- **Check In Files/Project**—checks in a file or project.
- **Undo Check Out**—releases the checkout on a file, restores it to pre-checkout condition.
- **Share Files**—brings up a dialog to bring files from other projects into this one. See the "Share" section later in this chapter.
- **Branch Files**—breaks the sharing of a file by creating an independent copy in the current project.
- **View File**—brings up the file in the registered viewer or the SourceSafe text editor.
- **Edit File**—checks the file out, then brings it up in the appropriate editor.
- **File/Project Difference**—shows the difference between the currently selected file and either a different version or different file on disk, as specified by the user.
- **Show Properties**—displays the Property sheet for the selected item.
- **Show History**—displays the history of actions taken and labels applied against this item.
- **Find in Files**—pops up a dialog for searching for strings within files.
- **Set Working Folder**—sets the default destination folder for getting files from the SourceSafe database.
- **Refresh File List**—queries the database and updates the status of each of the folders and files.
- **Help**—brings up the HTML Help engine and the SourceSafe Help file.

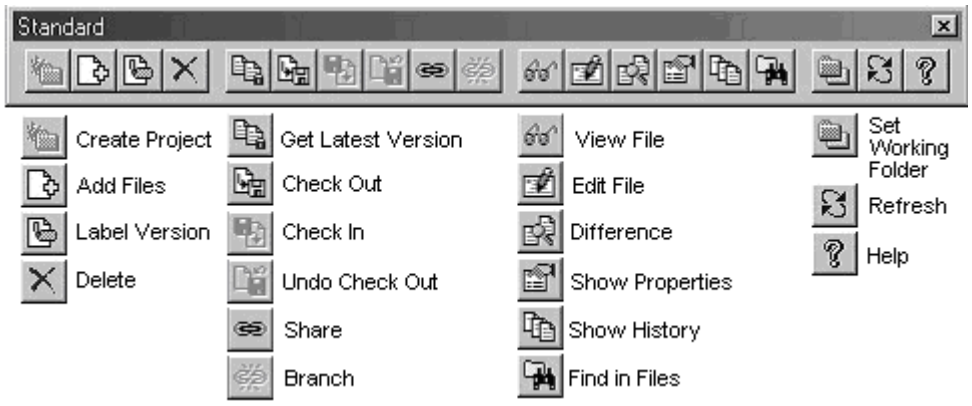

***Figure 8**. The SourceSafe Explorer toolbar.*

The toolbar behaves like older Windows toolbars, with a few funny exceptions. This toolbar is of the pre-Office 97 style, without the "Smart ToolBar" or "CoolBar" looks of the most recent Microsoft applications. You can drag the toolbar by clicking and dragging on the space between buttons. Once it's undocked from the normal toolbar space, you should be able to resize the toolbar, stack the buttons two, three or four high, and split on the spacers between the button groups, but this resizing functionality is missing. The toolbar does have the option to be customized, through the Customize Toolbar dialog available on the Tools menu (see **Figure 54**, on page 66). Finally, if you click the close button on the floating toolbar, the toolbar will disappear, and it can be a little daunting to bring it back. There is no option to restore the toolbar on the View menu, like in many other applications. Use the View tab of the Tools | Options dialog (see **Figure 47**, on page 61) to toggle the toolbar's visibility.

Menu

The Visual SourceSafe Explorer menu displays most of the functions available from within the interface, although a few are hidden a little deeper, coming off some of the dialogs. Because nearly every menu function has a corresponding dialog associated with it, the functions of the menu are just mentioned briefly here, and described in detail in the "Dialog by dialog" section later in this chapter. Check there for much more information.

File

- **Open SourceSafe Database**—opens a different database, optionally making it the default.
- **Add Files**—adds files to the currently selected project.
- **Create Project**—creates a new project, as a subproject of the current one.
- **Delete**—deletes the selected file or project.
- **Rename**—renames a file or project.
- **Properties**—displays a tabbed dialog of detailed information on the selected item.
- **Set Working Folder**—sets the default folder for local files for the project.
- **Create Shortcut**—creates a desktop shortcut for the current project and database.
- **Label…**—adds a label to facilitate retrieving earlier versions of source code.
- **Move**—copies a file to a new destination, erasing it from the original location.
- **MRU**—a most recently used list of Visual SourceSafe databases.
- **Exit**—exits the VSS Explorer application.

Edit

- **View File**—opens the file for viewing.
- **Edit File**—checks the file out for editing.
- **Select…**—allows you to pick multiple files by matching a file pattern.
- **Select All**—selects all files in the file list.
- **Invert Selection**—reverses the selection of files.

View

- **Sort (Name, Type, User, Date, Checkout Folder)**—sorts the file list by column.
- **Search (Wildcard Search, Status Search)**—shows only files matching your criteria.
- **Cancel Search**—clears the search and displays all files.
- **Refresh File List**—re-reads the SourceSafe database.

SourceSafe

- **Get Latest Version**—refreshes your local copy from SourceSafe's database.
- **Check Out**—reserves the file for you to make changes.
- **Check In**—returns your changes to the SourceSafe database.
- **Undo Check Out**—restores your local file to the master copy in SourceSafe.
- **Share**—lets a file be used by more than one project.
- **Branch**—splits off a shared file into an independent file.
- **Merge Branches**—merges changes from branched files.

Tools

- **Show History**—lists actions against a file or project.
- **Show Differences**—displays differences between any two files or versions.
- **Find in Files…**—searches for text within files.
- **Files Report…**—lists files displayed in file list to printer or clipboard.
- **Options**—controls many settings and interface elements.
- **Font**—sets the font used in the tree view and file view panels.

- **Customize Toolbar**—lets you add or drop buttons from the toolbar.
- **Change Password**—allows you to change your password.

Web

- **Deploy**—allows you to publish your Web site directly out of SourceSafe.
- **Check Hyperlinks**—checks the Web project for internal or Internet consistency.
- **Create Site Map**—generates an HTML page mapping your Web project.
- **Help**—there's nothing remarkable or non-standard here; if you've worked with any other Microsoft product, you know how to operate this one. If not, a few minutes of exploration makes this menu pretty self-explanatory.

Project tree view

The tree view interface has become familiar, thanks to the Windows Explorer interface. SourceSafe refers to each of the folders as a project, which can be a little confusing to developers who may view a project as a set of folders. Folders may be nested within folders, and many commands offer "recursion"—letting the effect of the command be repeated in each of the subfolders.

While folders can be mapped directly to a series of directories, in many cases, you may choose to accumulate subprojects logically under the project they apply to. For example, in development shops I have worked in, we kept design documents, contracts and diagrams in subprojects of a software development project. Using the Set Working Folder feature, these subprojects may be assigned to different directories, or even different disk drives, but logically aggregated with the source code they are associated with.

File pane

The file pane shows files that exist in the currently selected project. Note, however, that unlike the Windows Explorer, the file pane does not show subprojects. You still must use the project pane on the left to navigate between projects.

The file pane shows the filename, user (if the file is checked out), date and time of the file and the folder to which a file is checked out. Clicking on the column headers sorts the display by that column. Clicking again on the column header reverses the order.

Right-mouse-clicking in the file pane (or pressing the "Properties" key on a Windows-enhanced keyboard) brings up a context-sensitive menu (see **Figure 9**) with options to operate on the currently selected file. These menu options duplicate those available on the toolbar or the application menu.

***Figure 9**. The context-sensitive menu available in the File window conveniently duplicates functionality available from the toolbar and application menu.*

Results pane

The bottom window, referred to as the "results pane" in the SourceSafe documentation, reports the result of executing some commands. It is not always that informative, and many users choose to either minimize its size or close it altogether. Use the Tools | Options View tab (see **Figure 47**, on page 61) to hide this panel from sight. Alternatively, you can temporarily hide it from view by resizing the upper panes, dragging the mouse on the divider between the panes.

Status bar

The status bar displays the login name of the current user, the sort order used in the file window, the number of files in the current project, and routine messages. When a long operation is going on, the left-hand corner of the status bar displays a Cancel button where you can stop the operation. Like the results pane, you can toggle the display of the status bar through the Tools | Options View tab.

Drag and drop

A discussion of the SourceSafe interface would be incomplete without mentioning a feature that's missed by many users. Drag and drop is well implemented in SourceSafe, both within the application and between SourceSafe and the Windows Explorer. Files and entire projects can be shared between projects by dragging the source item to the destination. Drag with the right mouse button to get a menu of options—Share, Share and Branch, Move or Cancel—when you release the mouse. Dragging with the left mouse button only shares.

Files can also be added to SourceSafe by dragging them, or the folders they are in, from the Explorer into the SourceSafe project tree view. Unfortunately, dragging *from* SourceSafe onto the Explorer doesn't seem to be supported. This would make a nice Get function but was not implemented by the SourceSafe developers.

Reports

Many users are surprised to find there isn't a Reports option as one of the main items off the main menu bar. Others have asked me what kind of driver they would need to access the data engine from a third-party reporting tool. There is no centralized reporting facility in Visual SourceSafe, nor is there a way to access the data for reporting from a third-party tool. However, there are still many reports available from different parts of the interface. There is also the possibility of setting up many routine reports through batch files or scripts that use the command-line interface into SourceSafe (see Chapter 8, "Beyond the Basics," for more on the command-line interface).

More details are available in the following sections, but here are the places you can look for reports. To see all files, or a list of files meeting specific criteria, use the Sort and Search options off the View menu to limit the files displayed. Then, print the list using the Files Report menu option off the Tools menu. For more details on a project or individual file, you can print each page of the Property sheet, using the Report button, and you can also print the file or project's history using the Report button on the History page.

In these days of WYSIWYG report engines, SourceSafe has fallen behind. For day-to-day use, the plain text output of the reporting tools is usually adequate. But if you have a file that needs more slick presentation, use the Clipboard output option, available on each report dialog, to copy the results to the clipboard, and then you can paste them into a word processor to add styles, font sizes and even color to the reports.

Dialog by dialog

In this section, each dialog is examined in turn, with some additional commentary about the unique items available on that dialog, or the effects the choices presented in that dialog have on the overall operation.

You are unlikely to see all of these dialogs during routine operations, and, in fact, if you are seeing all of these dialogs, it's hard to believe you're getting any work done. As I show much later in this chapter, the Warnings and Command Dialog tabs of the Tools | Options dialog let you turn many of these dialogs off, so the default behavior is performed without all of the pesky confirmations.

However, there are some situations where you want to see these dialogs, either to confirm settings or because you want to override one of the default behaviors. In those cases, hold down the Shift key in order to force the dialog to appear. Surprisingly, the Shift-key trick works not only with SourceSafe's menu, but also with SourceSafe's toolbar and with the right-mouse-click shortcut menus as well.

The dialogs that you see when you are operating SourceSafe may appear slightly different from the dialogs illustrated here. In this book, all of the dialogs have been expanded to display all of the options available. Dialogs can be expanded by pressing the button on the lower right portion of the dialog labeled "Advanced >>."

The File menu

Items on the File menu, not surprisingly, mostly have to do with files. Here, many of the commands also located on the toolbar and the context-sensitive menu can be found.

Open Database

Use this dialog (see **Figure 10**), available as the first option off the File menu, to open a different database. The dialog has several more functions, though.

The check box on the bottom of the dialog designates that the database you open will be the default database opened the next time you start SourceSafe. This check box is selected when the dialog appears, and this feature trips me up all the time. Because integrated source code control depends on the currently selected database, accidentally changing default databases with this dialog can wreak havoc with your integration. Watch this check box carefully when you use the dialog, and remember to turn off the check box unless you mean to change the default behavior of SourceSafe!

The Browse… button allows you to add another SourceSafe database to the dialog's list. (This list, by the way, is stored in the Registry, under the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SourceSafe\Databases key.) The Remove button only removes the database from the list, but it does not erase files from disk. The Username text box lets you put in the name that should be used as the default to log into the database.

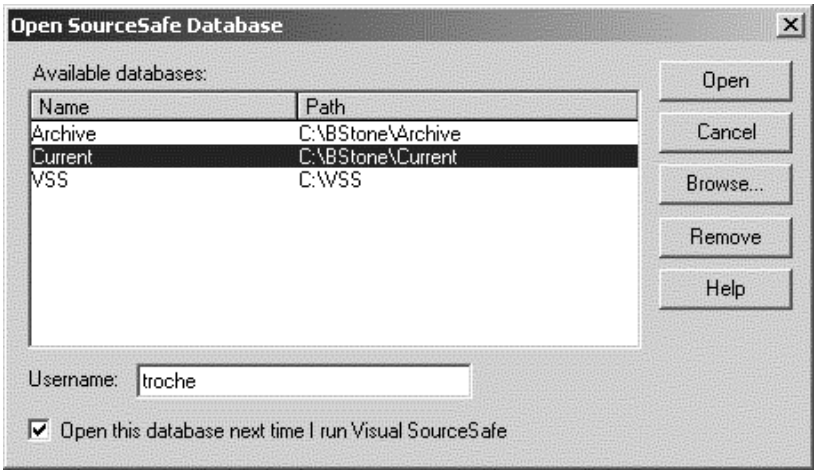

***Figure 10**. The Open dialog also lets you set the default database opened the next time you start SourceSafe.*

Add Files

Use the Add Files toolbar button or the File | Add File menu option to add files to the currently selected project (see **Figure 11**). Multiple files can be selected from the list on the left with the Windows standard Shift-Click and Ctrl-Click shortcuts. You can limit the list of files by selecting the "List files of type:" drop-down (those file types can be modified in the File Types tab of the Administrator's Tools | Options dialog, or by editing the SRCSAFE.INI file directly). Use the Add button to add the selected files. Use the Close button to close the dialog. Use the View button to view the selected file (see the discussion of the View dialog later in this chapter for options). Use the Network button to locate files not available on a mapped drive.

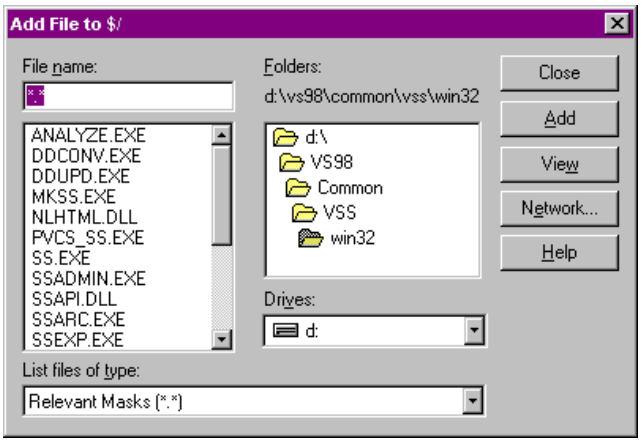

***Figure 11**. The Add Files dialog allows you to add multiple files to the current project.*

Create Project

Create a project using the toolbar button or the File | New Project menu option. Use care when creating a new project, as the project is created as a subproject of the currently selected project. Check the caption of the dialog (see **Figure 12**) for the path where your new project is going to be created. (You can right-mouse-drag-and-drop it to the correct location if you do create the project in the wrong location by mistake.) Adding a comment (visible in the Property dialog) helps you and others determine what the project is about.

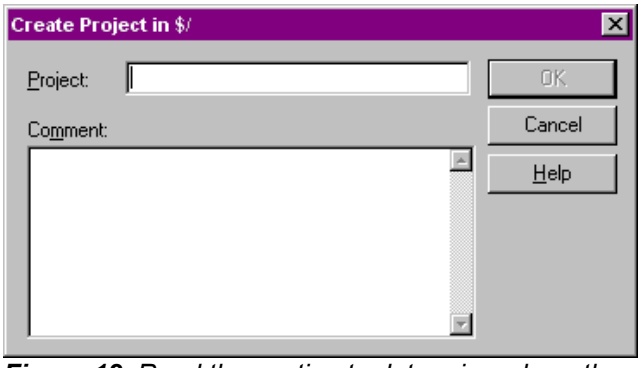

***Figure 12**. Read the caption to determine where the project is going to be created.*

Delete

The Delete option, available from the toolbar or the File | Delete menu item, confirms your intention to delete the currently selected file or project (see **Figure 13**). If you have been given Destroy rights to the project, you will also see the "Destroy permanently" check box enabled. Destroying the file permanently removes the file completely from SourceSafe and does not allow you to retrieve the file later. As a general rule, it is better to choose not

to destroy the files here, but rather have the Administrator choose to Purge the items after archiving a backup copy.

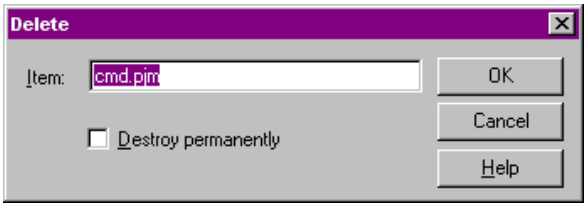

***Figure 13**. The Delete dialog confirms the deletion, and optionally offers to obliterate the files.*

Rename

Rename doesn't have a dialog that goes with it; rather, it uses the same metaphor as the Windows Explorer tool in allowing you to edit the selected project or file in place, replacing the file or project name with a text box with the current name filled in and selected.

Properties

Properties brings up a rich dialog with a lot of information, as well as the ability to print the information presented. The dialog has four tabs: General (see **Figure 14**), Check Out Status (**Figure 15**), Links (**Figure 19**) and Paths (**Figure 21**).

The General tab displays general information on the file, including its size, type, version and comment. Note that three options on this dialog can be changed. The Type drop-down box allows you to designate the file as either binary or text. SourceSafe is pretty clever about figuring this out by itself—it searches the file for a CHR(0). If it finds one, it knows the file is binary. If not, it assumes the file is a text file. The type of the file is important when it comes to displaying differences for the file—a visual display of differences is attempted only for text files.

The second changeable option is the check box "Store only latest version." If this option is selected, file differences are not retained. This means you can't Diff the file against previous versions, nor can you Rollback to an earlier version. This option might save you some space, particularly if you are storing large binary files, and if you either have the ability to restore earlier versions from another source, or don't have a need for the earlier versions.

The Comment edit box is also editable. This dialog edits the comment associated with the first version of the file, and doesn't change the comments added to later versions as the file is checked in and out.

Changing any of these options pops up a "Save these Changes?" dialog to confirm you want to save the changes.

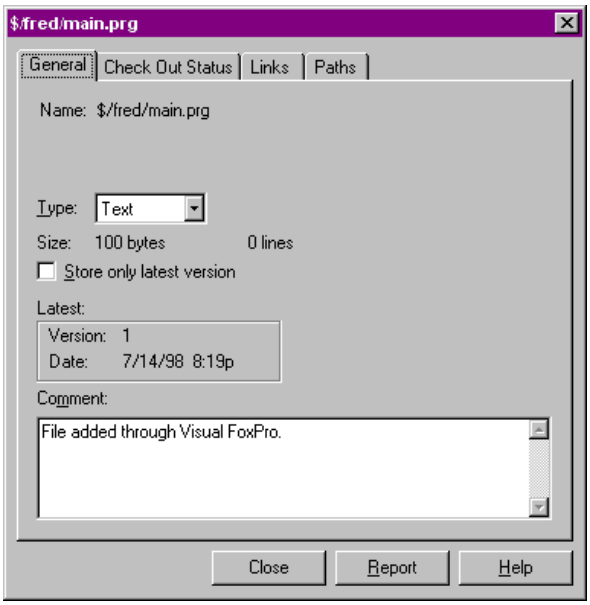

***Figure 14**. The General tab of the Properties dialog displays the status, version and comments associated with the file. It also provides the ability to specify the file type, modify the original file comment and store only the most recent file version.*

The Check Out Status tab of the Properties dialog shows information on the current checkouts of a file. This includes both the user and the machine onto which the file was checked out, as well as the date and time. The comment edit box displays the comment typed in when the file was checked out. This comment is often overwritten when the file is checked in.

If more than one user has the file checked out, the Check Out Status tab appears quite different (see **Figure 16**). The list shows all of the users who have the file checked out, and the folder to which the file was checked out. If you have the file checked out, the buttons to check the file in or undo the checkout are enabled when your name is highlighted. Highlighting a name and pressing the Details button brings up a dialog very similar to the original, single-checkout tab (see **Figure 17**).

Finally, the Properties dialog has a Report button. The Report button brings up a dialog with choices of output destination (printer, file or clipboard) and allows you to preview or cancel the operation. Preview forms are shown as part of **Figure 18**, **Figure 20** and **Figure 22**. The line numbers shown in the preview window do not actually appear in the output. (Hint: To get rid of the line numbers in the preview window, de-select the “Show line numbers” check box on the Differences tab of the Tools | Options dialog.) Each of the tabs of the Property sheet has a report that matches the items displayed on that page.

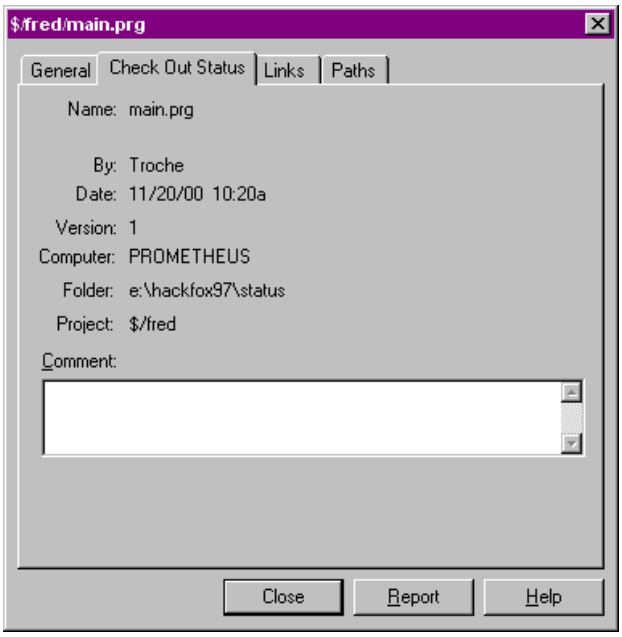

***Figure 15**. The Check Out Status tab of the Properties dialog shows who has the file checked out, when it was checked out, and on which machine and project.*

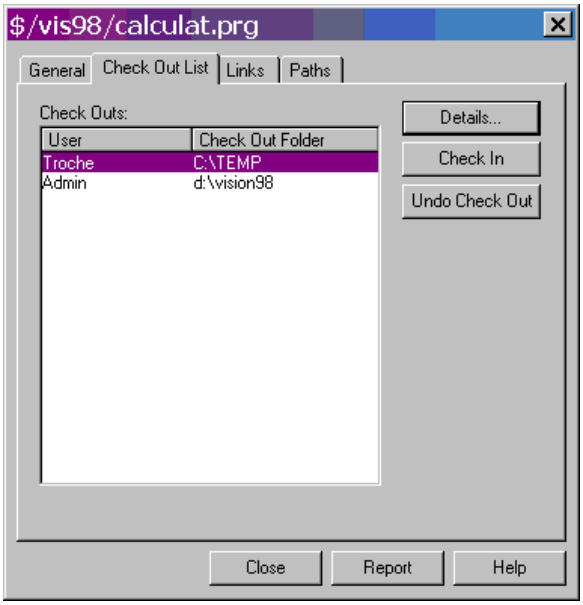

***Figure 16**. The Check Out Status tab appears quite different when the file has been checked out more than once.*

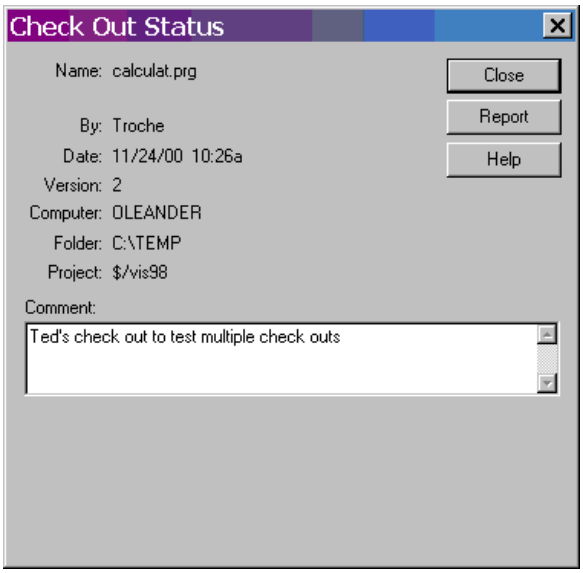

***Figure 17**. Selecting the Details button from the Check Out Status tab displays the information for a single checkout.*

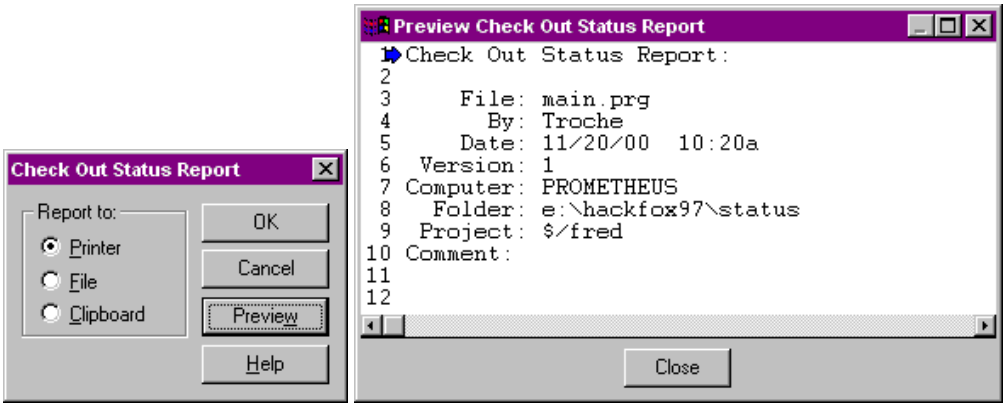

***Figure 18**. The Check Out Status Report can give you a hard copy of the information available from the Check Out Status tab of the Property sheet, and can be accessed by clicking the Report button of that tab.*

The Links tab of the Property sheet show a list of all of the project paths that have this file shared into it. This shows one of the coolest features of the Share functionality. With a Links list, when you make a change to a shared file, you know all of the projects that you are affecting. Many developers have run into the problem of changed shared library code only to discover, weeks or months later, that unanticipated side effects of the change break other applications. With the Links functionality, you can identify which applications require testing whenever a change to shared code takes place.

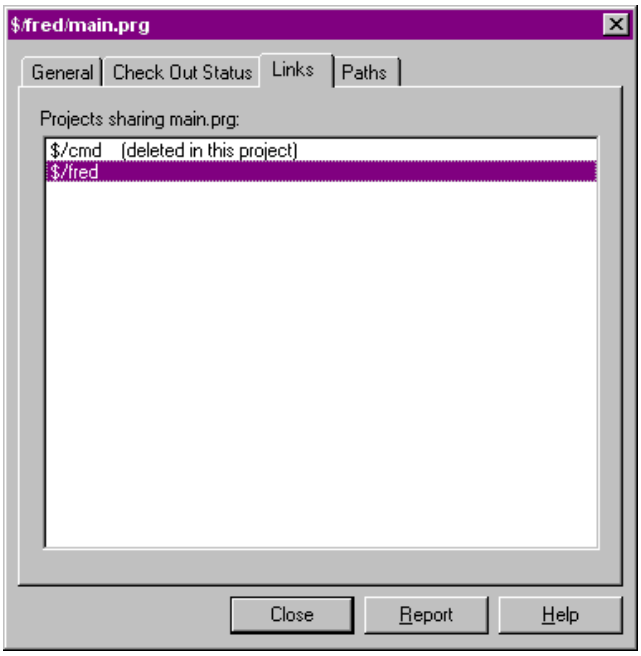

***Figure 19**. The Links dialog shows all projects that share this file.*

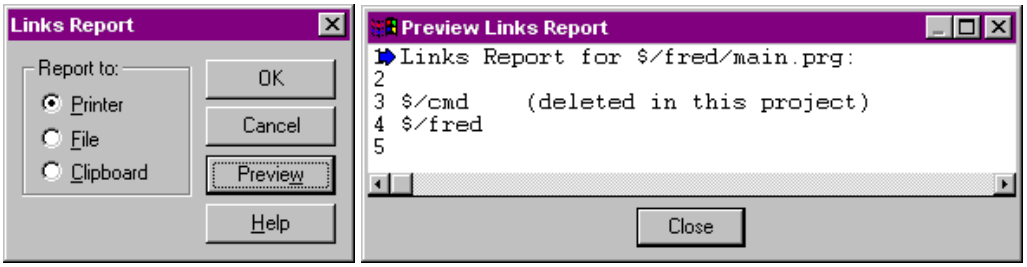

***Figure 20**. The Links report, available from the Report button on the Links tab of the Property sheet, gives you a hard copy of the shares a file has.*

The Paths tab of the Property sheet, along with its corresponding report (see **Figure 21** and **Figure 22**) show the branches of a source code file. Branches are independent files that started life as one file, which was shared between two projects, and then the share (the link) was broken, creating two separate files. Like the shares shown in the Links tab, this history lets you trace the paths source code took during development, and lets you consider whether the two branches' code paths might need to be merged together again.

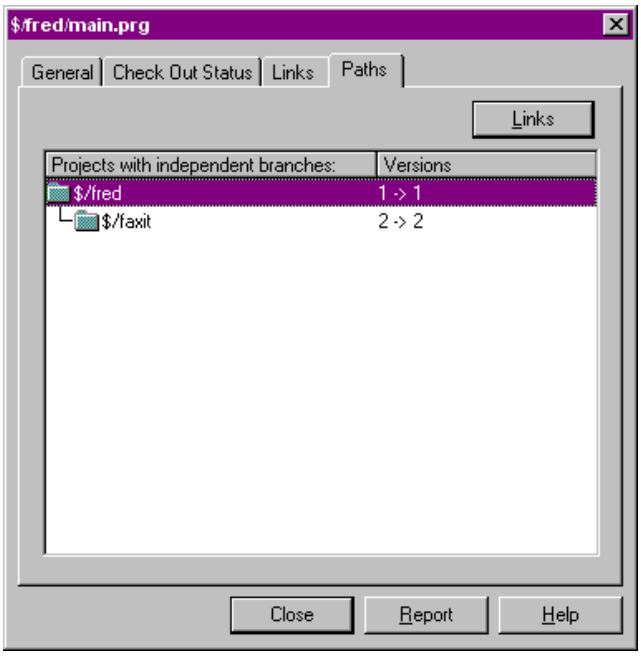

***Figure 21**. The Paths tab of the Property sheet show the branches (separate files created from shares) created from the original file.*

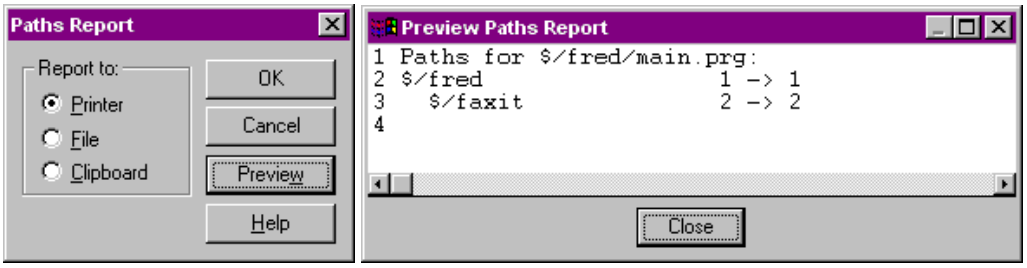

***Figure 22**. The Paths Report, from the Paths tab of the Property sheet, can provide a printed copy of the Path history.*

Set Working Folder

The working folder is the default directory where a file is written when you check it out or you get the latest version. In order to check out or get files, SourceSafe asks for a working folder if one is not already set up. Use the Set Working Folder menu option off the File menu, the Set Working Folder button on the toolbar, or the Ctrl-D keyboard shortcut to bring up the dialog (see **Figure 23**). If a working folder is set for a parent project, the child projects are assumed to have a working folder under their parent folders, unless explicitly overridden. So, if the parent Accounting project is assigned a working folder of C:\Accounts, the child project of Menu is assumed to have a working folder of C:\Accounts\Menu, unless a different working

folder is assigned to it. If the folder or subsidiary subfolders do not exist, you will be prompted to allow SourceSafe to create them when you check out or get the latest files.

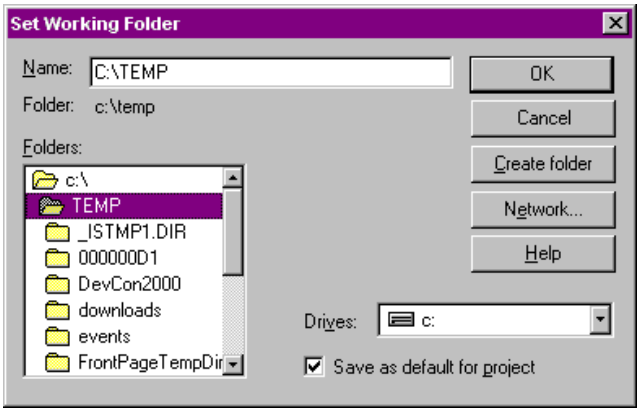

***Figure 23**. The working folder is your local work area for the project.*

Create Shortcut

Create Shortcut is a new option for Visual SourceSafe 6.0, appearing only on the File menu. It creates a desktop shortcut to the current Visual SourceSafe Explorer executable and includes the –P and –S command-line switches to specify the project and SrcSafe.INI file, respectively. The command-line switches are discussed in Chapter 8, "Beyond the Basics."

Label

Labeling is a very important feature of source code control. With labeling, a milestone or checkpoint can be applied to all files in a project, and all subsidiary projects. This label can be used later to retrieve all of the files at that milestone. This is an essential feature that allows you to continue development on a project, but be able to retrieve an earlier version if needed for customer support or to reproduce a problem identified in a later QA process. Use labels for each major and minor release of your source code, and anytime you want to be able to return to a snapshot of your code.

Apply a label by first selecting the item you want labeled. While it is possible to label an individual file, you are more likely going to want to select the highest project within the project tree to which the label applies. Select Label… from the File menu, or the Label Version toolbar button. A dialog (see **Figure 24**) appears that allows you to specify the label string that appears in the project and item's history listings, as well as a longer comment that is available from the Details button of the History viewer (see the History dialog, discussed later in this chapter).

Labels are always applied recursively to all files and projects under the selected project.

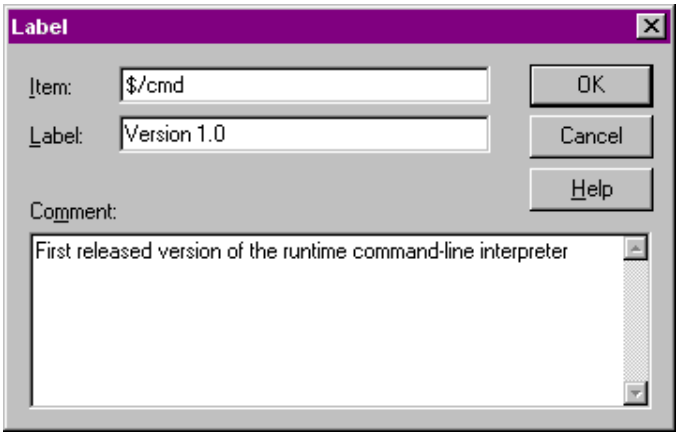

***Figure 24**. Use the Label dialog to mark all files with a milestone you can use to retrieve the files later.*

Move

The Move dialog allows you to move a file or project from one project folder to another. If you examine the item's Properties, you will see that SourceSafe is simply performing a share and then deleting the original file. The Move dialog (see **Figure 25**) allows you to pick the destination folder.

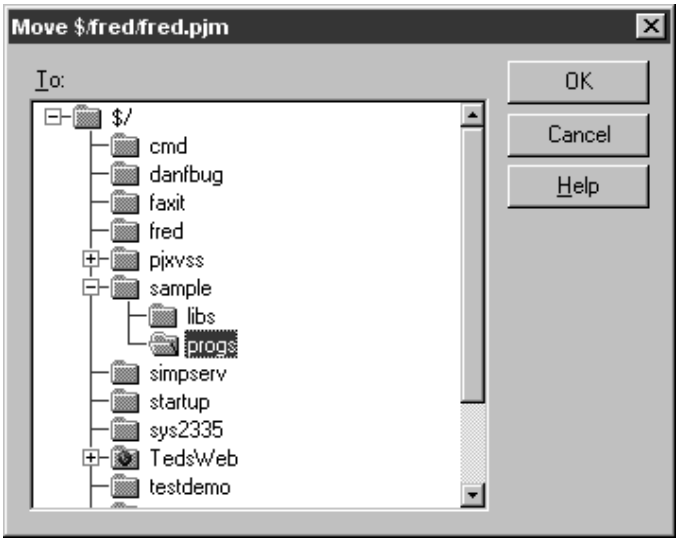

***Figure 25**. The Move dialog moves the selected file or project by performing a share and then deleting the original item.*

The Edit menu

The Edit menu is the place where you would expect to see the Windows standard Copy-Cut-Paste menu items. But because Visual SourceSafe isn't truly an editor, such functions are inappropriate. Instead, those items most closely related to editing can be found here. View and Edit options allow you to look at or modify the selected file. Select, Select All and Invert Selection options allow you to select one or more files from the file window, in order to perform some command on the files.

View File and Edit File

The View File and Edit File dialogs are the same, differing only by which of the two option buttons at the top of the dialog (see **Figure 26**) is selected.

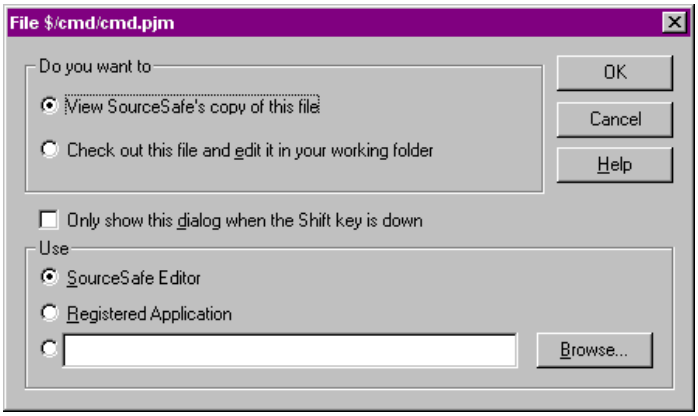

***Figure 26**. The View and Edit options share the same dialog. Use caution when viewing a file with its registered application—some run, rather than edit, the file.*

SourceSafe is very clever about the options it presents. SourceSafe determines, based on the file extension, which application it should use to view or edit the file. Files with an extension of "txt," for example, are brought up in Notepad, while files of "doc" type are opened in Microsoft Word. However, this feature has its hazards. Some applications, like the Registry editor REGEDIT, are registered to "edit" a file type that they actually execute. If you try to view a file with an extension of "reg," the REGEDIT program actually reads the file and inserts the contents into the Registry! This is obviously not the desired effect and can have disastrous consequences. For that reason, I strongly recommend clicking on the Advanced button in the Edit dialog to confirm which program will edit your file anytime there is the slightest doubt in your mind.

Microsoft has actually prevented the REGEDIT problem (and similar problems with VB and VC++) by adding the following lines to the default SS.INI file:

```
; The following lines force SourceSafe not to execute certain file types.
.reg (Win) = notepad.exe
.vbp (Win) = notepad.exe
.vcp (Win) = notepad.exe
.mak (Win) = notepad.exe
.bat (Win) = notepad.exe
```

FoxPro developers are likely going to want to add the "prg" extension to that list, to avoid accidentally running programs from SourceSafe when they only mean to look at them.

When the Advanced button has been selected on the View/Edit dialog, options appear to allow you to select which editor to use. If an application has registered that it can edit the file, that application appears as the second option. If you have already selected an editor for this file type during this editing session, that selection would appear as the third option. You can type your preference directly into the text box for the third option, entering, for example, Notepad.exe. There is another way to have a default editor appear, and that is by specifying an editor on the General tab of the Tools | Options dialog (see **Figure 45**, on page 59). If you have specified an editor there, that choice appears as the default in the third option.

Select

The Select dialog allows you to multiple-select files in the current project without the hassle of having to use one of the extended selection methods (Click and Shift-Click for a range, or Click and Ctrl-Click for multiple, individual items). Use the DOS wildcard characters of ? for a single character and * for zero or more characters to specify a filename skeleton, and press the Select button to highlight those matching files (see **Figure 27**). Multiple file skeletons can be specified at the same time by separating them with semicolons. When dealing with a large list of files, this can be a time-saver, and also help to avoid errors.

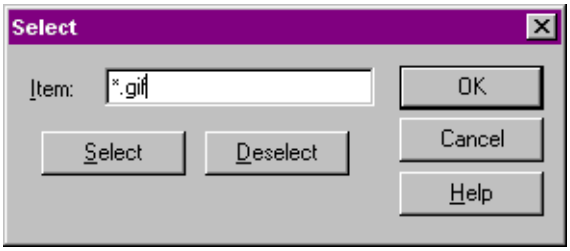

***Figure 27**. The Select dialog lets you select or deselect all files that match a file template that you supply. This is a lot easier to manage than trying to Ctrl-Click and scroll a file list.*

The Select dialog can be used several times, and selections can be added together. So, for example, to specify all of the GIFs and JPGs in a folder, except for those that begin with M, you can first specify the skeleton *.GIF;*.JPG and press Select. Then, change the skeleton to M*.* and press Deselect.

Select All

The Select All option selects all files in the current project.

Invert Selection

The Invert Selection option reverses the files selected and deselected.

The View menu

The View menu changes the appearance of the files presented. The Sort commands sort the list of files in the file list. The Search commands limit the display to only files that meet specified search criteria. The Refresh command requeries the database for changes that may have been made since the last update.

Sort (Name, Type, User, Date, Checkout Folder)

The Sort commands reorder the display of the files by the column specified. Clicking on the column header of the file list can also perform these menu commands. The menu option only sorts in descending order, while clicking on the column headers toggles between descending and ascending order. The current sort order is displayed in the status bar.

Search

The Search commands are unique in that they display only files meeting the specified search criteria, and hide the other files. Many SourceSafe neophytes are confused when many of their files fail to appear in the file list. If you suspect that some of your files have disappeared, remember to check the status bar, which displays "Searched" if a search is in effect, and use the menu option Cancel Search to restore the display of all files.

Wildcard Search

The Wildcard Search (see **Figure 28**) lets you search for files matching a file skeleton, similarly to the Select dialog. Use the DOS standard wildcards of ? and *, and separate multiple file specifications with a semicolon. The Search Area options let you specify whether the search is confined to the currently displayed project, whether it can include subprojects of the current project, or whether it should start from the tree root and include all projects. The latter two options display files from more than one project at a time in the file list, as shown in **Figure 29**. The status bar displays "Recursive" if you have selected to view files in a project and subprojects, and "Global" if you are searching all projects.

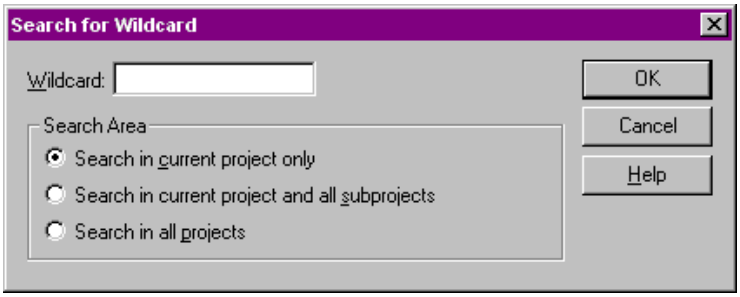

***Figure 28**. The Search for Wildcard dialog does not let you find one-eyed jacks. It does, however, locate files matching the wildcard template you supply.*

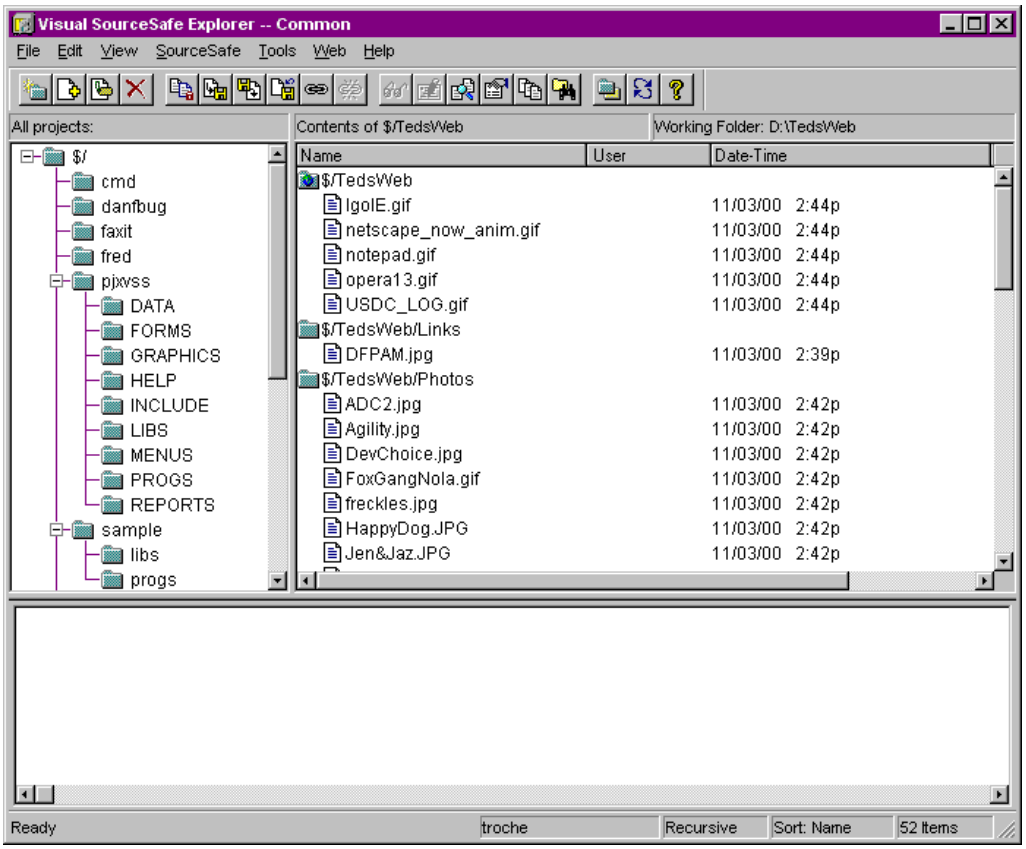

***Figure 29**. A wildcard search for *.GIF and *.JPG from all projects shows the path to multiple projects in the file list, and "Recursive" in the status bar.*

Status Search

The Status Search option (see **Figure 30**) lets you look for files that are checked out. This is a great tool for making sure that everything is checked in before a major beta or production build, or as a routine maintenance item to make sure that files aren't being kept checked out for excessive periods of time. You can also choose to search for files checked out by one person, a handy check before you send someone on vacation, or overseas on assignment. The Search Area options work exactly the same as for the Wildcard Search.

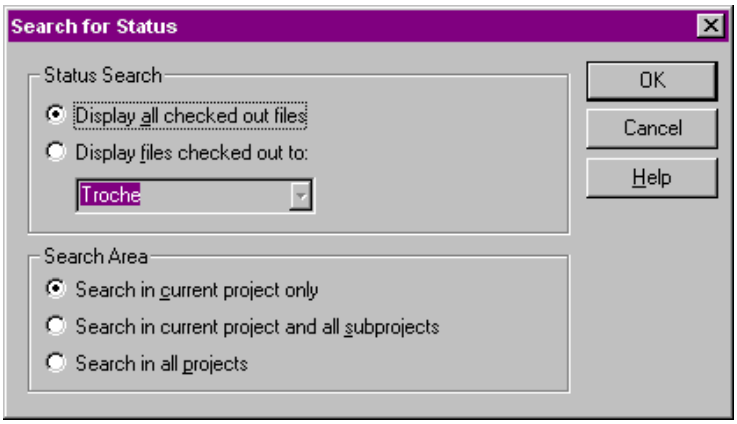

***Figure 30**. The Status Search dialog lets you display only those files meeting the criteria you specify.*

Cancel Search

The Cancel Search menu item clears the current search criteria, restoring all files to the display.

Refresh File List

The Refresh File List menu item searches the database again for files meeting the current search criteria, or all files if no criteria is selected, and displays the appropriate file list. You'll want to refresh if the underlying files are likely to have changed from the actions of others on the network.

The SourceSafe menu

The SourceSafe menu contains those functions that make up the core of the SourceSafe functionality—the ability to get, check in, check out, share, branch and merge files.

Get Latest Version

The Get dialog (see **Figure 31**) lets you specify where you want the copy of the file to be placed. You also have the option of flipping the Read-only flag on the file so that it is writable. Use this option with care, as the Read-only flag is a nice reminder that the file is not checked out.

You also have options on the dialog to set the timestamp on the file. Typically, you want to leave this set to "Current." In most development environments, having the timestamp updated to the current time means that the file will be recompiled the next time you recompile or make the project. If this file were left at the time the file was last modified or updated, it is possible that it would be ignored by the compiler as not having been changed, and you would not gain the benefit of the most recent code in your project. So, change the "Set file time" option with care, and ensure that you understand the implications for your development environment.

The final option, how to handle writable files, is also one you should change with care. The default, "Ask," alerts you whenever SourceSafe attempts to overwrite a writable file. The other options may have side effects you won't like: "Replace" just replaces the file anyway,

potentially erasing changes you have made (when you should have checked out the file), "Skip" leaves a potentially older or out-of-date file on your disk, and "Merge" could allow changes in your local edition to remain in the local version, but still does not add them into SourceSafe's database. In many cases, none of these other options are ones you want to consider. Change from the "Ask" default only after serious consideration.

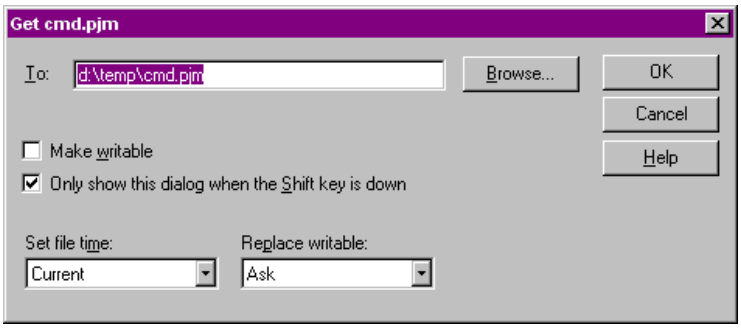

***Figure 31**. The Get dialog offers several key options, like making the file writable or changing the file's timestamp.*

Check Out

The Check Out dialog (see **Figure 32**) allows you to confirm or override where you want the file to be checked out to, and also gives you options similar to the Get dialog discussed in the preceding section. However, there's one other option that may seem strange at first: the option to check the file out without getting a local copy.

There are several situations where not getting a local copy makes sense. The most common situation is one where you have had to make changes to code you did not check out. Perhaps the SourceSafe files were unavailable due to network problems, or you had your machine disconnected from the network. In any case, you've made changes, and now you want to integrate them back into the SourceSafe database. Perform a Diff (explained later) to ensure that the only differences between your copy of the file and the one in SourceSafe are those changes you made, and then check out the file, checking the box to not get a local copy. You can then check your changes in, so that the SourceSafe database is up-to-date. Whew!

The Comment edit box allows you to make a comment on why you are checking out the file. This is a good idea, to let your fellow developers know why you have a file checked out. It also appears as the default comment when you go to check the file in. That's both a handy reminder and a potential time-saver.

If you want to check out a particular file exclusively, the "Allow multiple checkouts" check box allows you to override the global setting allowing multiple checkouts (set in the Tools | Options dialog of the Administrator tool). If you de-select this check box, no other user can check this file out while you have it checked out. Use caution with this check box, however; clicking once seems to disable the check box, and you may need to click twice in order to clear the checkmark.

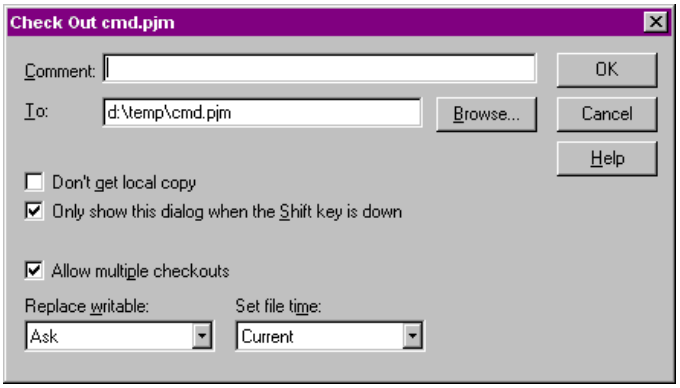

***Figure 32**. The Check Out dialog offers the option to check out the file without getting a local copy, a nice feature if you have had to make file changes without checking out the file, and now need to check back in your changed version.*

Check In

The Check In dialog (see **Figure 33**) allows you to update SourceSafe's master copy of a file with the changes you have made. Use the "Keep checked out" check box if you have more changes to make and you only want to update the copy in the SourceSafe database. Use the "Remove local copy" check box if you want to save diskspace, don't require a local copy, or want to make sure you can never edit an old copy of a file. Use the Comment edit box to detail what changes you made to the code. The comments can be very helpful in describing the overall flow of a development effort. Comments can be extracted from an entire project using the History options, described later in this chapter. So, add comments with care, so that they are useful later on when retrieved in a report.

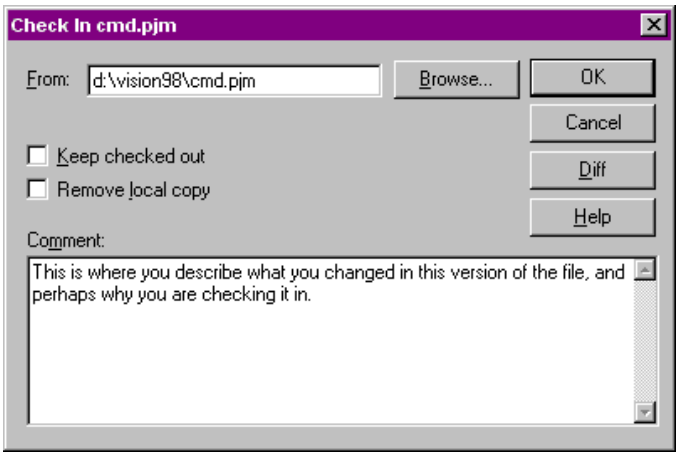

***Figure 33**. Use the comment area to describe your changes, useful information that you can extract from the database later. Use the Diff option to remind yourself of the changes you made.*

Undo Check Out

You would choose to undo a checkout (see **Figure 34**) when the file was checked out by mistake, or when the changes you have made to the file don't work and aren't worth saving. Undo normally replaces the file on your local working folder with the most recent file in the SourceSafe database. One other option, Leave, would leave your changes in your working folder, but clear your checkout, leaving you in a somewhat unpredictable state. You might use this option if someone else needed to exclusively check out the file, but you wanted to keep working on your changes. Bear in mind, however, that you will have to clean up the situation later. The last option, Delete, lets you remove the file from your working directory if you no longer require it.

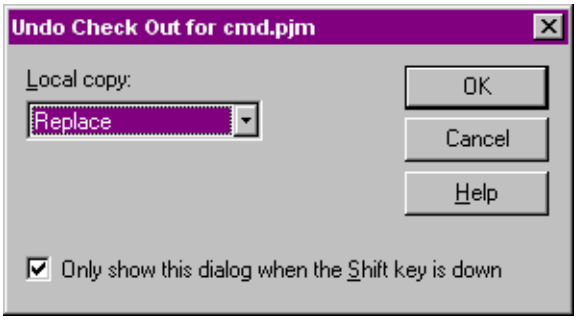

***Figure 34**. Undo Check Out lets you restore a file you checked out inadvertently.*

Share

The Share menu option lets you place a file in multiple branches of the SourceSafe tree. It's important to understand that a shared file does not "belong" to any one project more than any other. A shared file is simply pointed to from multiple locations, from different project hierarchies.

To share a file among projects, first select the project *into which* you want to share the file—not the project that owns the file currently, but rather the project to which you want to add the file. Select the Share menu option from the SourceSafe menu (or the shortcut menu, if highlighting a project) and the Share dialog (see **Figure 35**) appears. Navigate the project tree on the right to locate the source project, and select the file from the list on the left. Use the View button to bring up the View dialog (explained earlier). Press the Share button to share the file into the current project. Select the "Branch after share" check box if you want the current project to have an independent copy of the file selected—see the next section, "Branch," for more on branching.

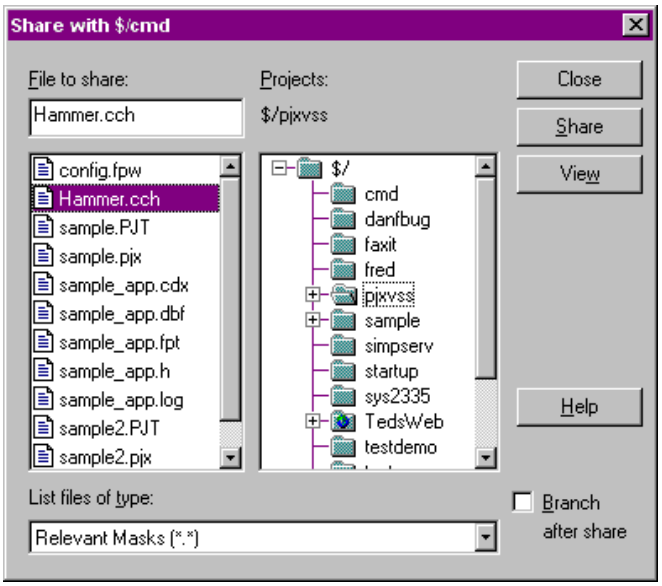

***Figure 35**. The Share dialog lets you link files from other projects into the current project.*

Branch

Branching breaks the link on shared files (see **Figure 36**). Use branching when you want to perform changes on a file in one project and not have the changes affect the other projects into which the file is shared.

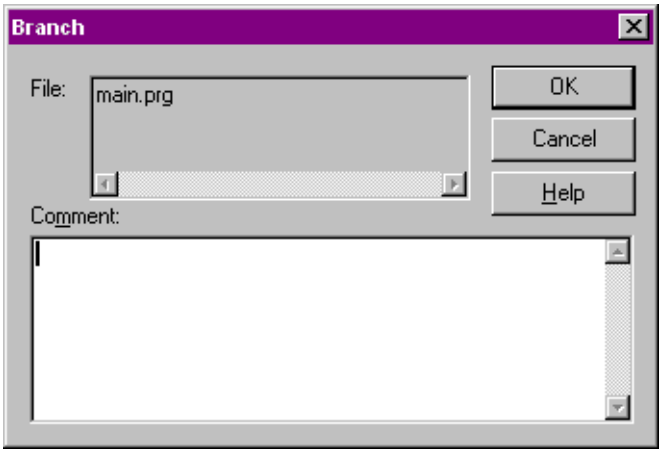

***Figure 36**. Branching breaks the link between shared files, creating a stand-alone file in the current project.*

Merge Branches

Merging branches allows you to bring back together two separate lines of source code development. Merging does not actually reunite two independent files into one shared file again, but rather merges the changes to one file into the other, leaving the two files independent. Calling up the Merge dialog (see **Figure 37**) allows you to pick the file from which the changes are read for merging. Note that the caption shows the destination file into which the merged changes are written. Once you have created the merged file, you can share it back into the other project, optionally pinning it to this version, so that both projects benefit from the merged file.

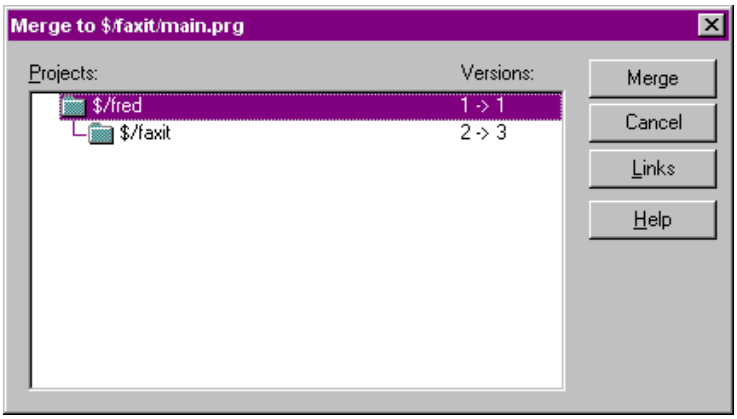

***Figure 37**. Merging branches allows changes to be reapplied across files that have been branched.*

The Tools menu

The Tools menu provides the utilities for working with SourceSafe files—the ability to trace version history and look at differences between files.

Show History

The Show History option, available from the menu, the toolbar, and the shortcut menu, lets you see what has been done to a file or a project over time. It also presents a tool, the History form, that lets you perform several complex operations against the file or project.

The first dialog that appears (see **Figure 38**) lets you limit the display of history to include labels, only labels, a range of dates (the "From:" and "To:" prompts) or only see actions performed by one user. After you make your choices and select OK, the main History form (see **Figure 39**) displays the history you've selected.

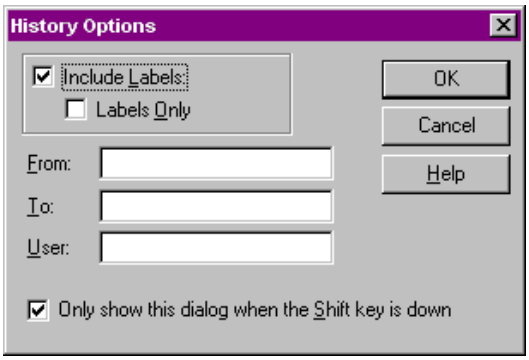

***Figure 38**. Before seeing the history of a file or project, the History Options dialog allows you to limit the information displayed to include or exclude labels, select a range of dates, or see actions by only one user.*

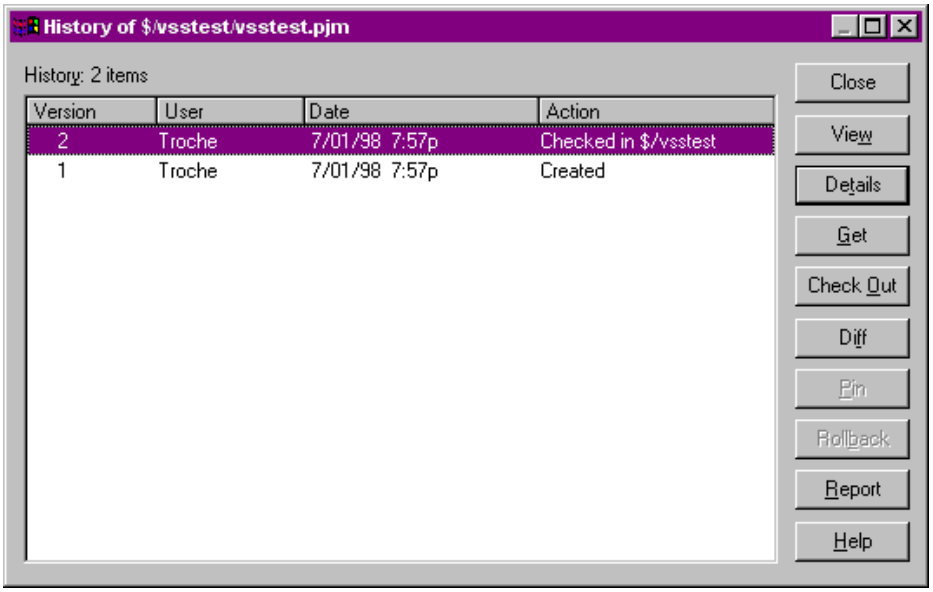

***Figure 39**. The History form is a powerful dialog, with a number of options available from the buttons on the right side of the form.*

History form

This History form shows the changes performed on a file over time, and provides tools to manipulate the file.

Most of the buttons on the right of the form perform functions that you've seen before, but at least two are unique to this form. The Pin button and the Rollback button provide functions worth reviewing in detail.

The Pin button holds the version of a shared file at the level highlighted when the Pin button is pressed. When a Get operation is performed against the source code for this project, the pinned version of the source is provided. A pinned file cannot be checked out until it has been unpinned, avoiding changes to previous versions. Pinning allows you to share a source code file across several projects, but control the release of new features by pinning them in the shared projects until it is appropriate to release them. When pinned, a version shows a pushpin in the left margin and the Pin button becomes the Unpin button (see **Figure 40**).

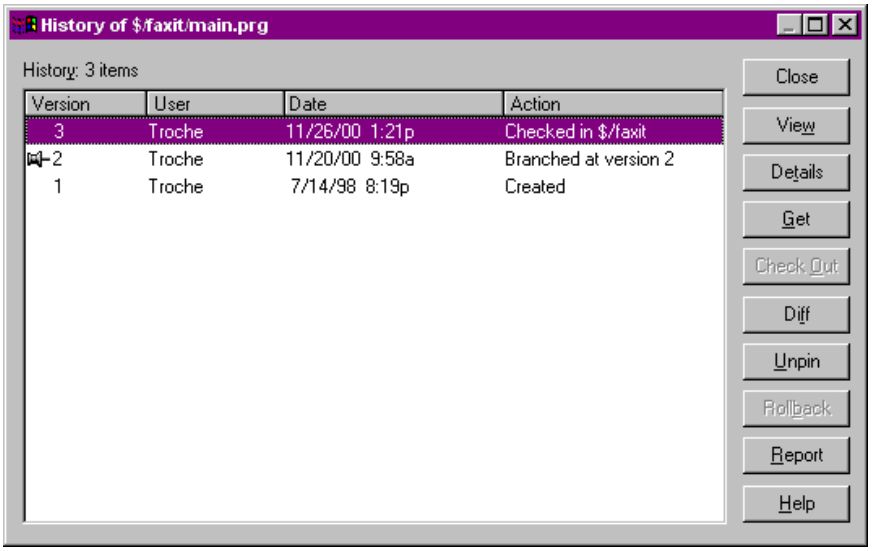

***Figure 40**. Pinning a version of a file displays the pushpin in the left margin, changes the Pin button caption to Unpin, and locks the file's contents at the pinned version.*

The Rollback button is a brute-force way of reverting to earlier versions of code, and while it has its uses, there are better ways to accomplish the same feat. Rollback, as indicated by the dire warning message that comes up when selecting it (see **Figure 41**), destroys history on the files. When you choose to roll back to an earlier version, SourceSafe wipes out the history and changes that occurred after that version and restores that version as the current one. This defeats some of the reason for keeping history on the files, as you cannot follow exactly what it is that was done. Instead, consider a "soft rollback": Check out the current version of the file, and then use the History form to Get an earlier version, overwriting the current one. Finally, check in the now-overwritten version. You get exactly the same result—you have the older version of the file as the current version—but you do not lose the history of the changes you went through to get there. It is quite possible that later in the development process, you will decide that those intermediate versions had some value, and you can use the History dialog to retrieve them.

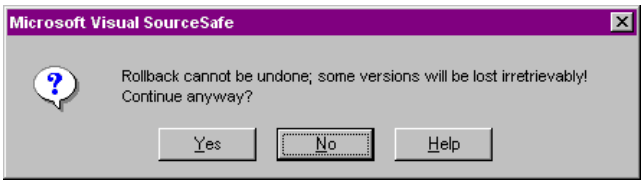

***Figure 41**. The Rollback confirmation dialog does its best to warn you that this is not a good idea. Try the suggestion included in the text instead.*

The other buttons on the History form call up other dialogs covered here. The Details button shows the file properties, with Next and Previous buttons to scroll through the history. The Get and Check Out buttons bring up the dialogs of the same name. Diff calls up the differencing engine, which is covered a little further along in this chapter. The Report button calls up a standard SourceSafe report dialog, with options to display details of each action and differences between the files at each version. This can be an excellent, detailed audit tool for reviewing all of the changes to a program.

If a project is highlighted in the SourceSafe Explorer window when History is selected from the application or context-sensitive menu, a few options that are only appropriate to projects are displayed. A Recursive check box is added to the first dialog (see **Figure 38**) to allow the inclusion of subproject files as well. Pin and Rollback buttons are not included in the main History form.

Show Differences

As mentioned earlier in the book, the purpose of SourceSafe is to be able to determine differences between files. Storing those differences and applying them to re-create different versions of the source code is what source code control is all about. The Difference Options dialog (see **Figure 42**), available from the main menu, toolbar and shortcut menu, exposes that differencing engine with a powerful dialog with many options.

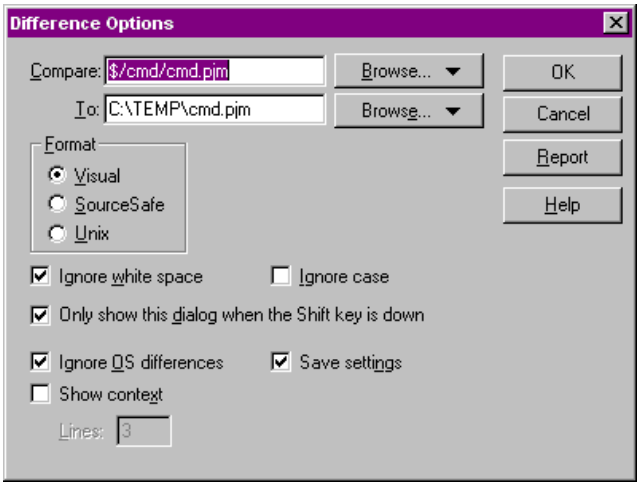

***Figure 42**. The Difference Options dialog provides numerous ways to view the difference between any two files, or versions of files, either in SourceSafe or on disk.*

The two items to be compared are specified at the top of the dialog. A specific file version can be specified in the format $/Project/Subproject/File;Version, such as:

```
$/Accounting/AR/Mainmenu.prg;2
```

The Difference dialog is not just limited to versions; it can be used to display the difference between any two text files, or between any text file and a SourceSafe version. The drop-down buttons to the right of the text boxes allow you to specify "SourceSafe Projects" or "Windows folder" and bring up an appropriate browser for each.

The formatting options determine the output of the differencing engine. In most cases, you would want the full-screen Visual difference (shown in **Figure 43**). However, if you wanted to take the differences and run them into a text processor or database engine, you could choose the text-based SourceSafe or standard UNIX difference formats for an easily parsable output.

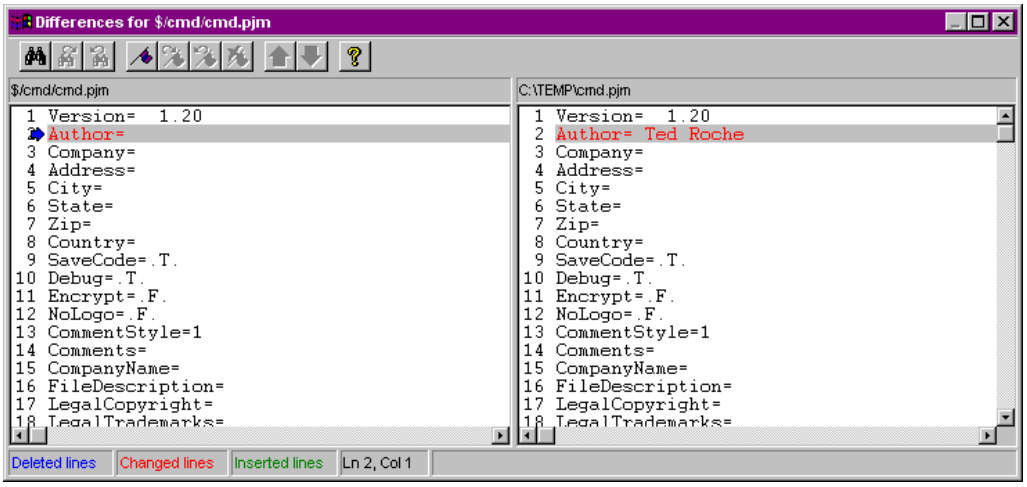

***Figure 43**. The Differences window shows what's changed. A key is included on the bottom status bar. A toolbar on top makes it easy to navigate between changes.*

Most of the other options are pretty self-explanatory. Ignoring white space prevents highlighting differences simply due to spaces, carriage returns or tabs. OS Differences would be the end-of-line differences between carriage return-line feed pairs (DOS/Windows), carriage returns (Apple Macintosh) and line feeds (UNIX). Context displays would be the ability to display lines surrounding a difference, to put the difference in context.

When you have the settings to your liking, press the OK button to bring up the Differences form.

Differences form

The Differences form shows the difference between the two selected files/versions. Directly below the toolbar are indented labels showing the names of the files or versions being compared. In the status bar at the bottom of the window is a key showing the meaning of the different colors within the display.

The toolbar lets you navigate a large document easily. The binoculars icon brings up a Find dialog. The next two buttons allow you to find the next or previous in the document. The next group of toolbar buttons allow setting a bookmark, navigating to the next or previous, and clearing all bookmarks. The final set of buttons, the large up and down arrows, let you jump to the previous or next difference in the file.

Find in Files…

The Find in Files dialog (see **Figure 44**) lets you search through all files containing a specified string. You have the typical options to match case, use "regular expressions" (DOS wildcards, not UNIX regular expressions), recursively search through subprojects, and list all shared files once or separately. The latter two options are available if you have highlighted a project to search, and do not appear if you highlighted a single file to search.

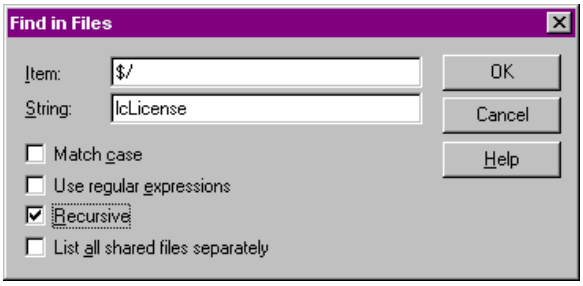

***Figure 44**. The Find in Files dialog lets you locate a string of characters in any of the files in your project.*

Files Report

Choosing the Files Report option from the menu will give you the option of printing a listing of those files shown in the File List. A typical SourceSafe dialog gives you choices of printing recursively through subprojects, or listing only filenames and not the details associated with them. The usual destinations of Printer, File and Clipboard are supported.

SourceSafe Explorer's Tools | Options dialog

It is very difficult to describe exactly what happens when a user interacts with SourceSafe because the product is extremely customizable. It is nearly always true that if you don't like the way SourceSafe does something, you can customize it to behave more to your liking. Many of these options are available in the Tools | Options dialog. The settings chosen in this dialog are unique to each user, and are stored in the user's SS.INI file. (For more information on configuring the INI files, see Chapter 8, "Beyond the Basics.") The Tools | Options dialog has eight tabs that cover a remarkable number of options.

The General tab (see **Figure 45**) provides the general options for working with the interface. Each of these can change the behavior of the product to adhere better to your needs. "Always keep files checked out" is ideal for a single developer. After the developer checks out files initially, they remain checked out, and the "Keep checked out" check box of the Check In dialog is checked when the dialog appears. This can speed operations for the developer who doesn't need to share files with others. The "Act on projects recursively" option similarly

enables the Recursive check box in those dialogs that have one. “Reuse last comment” can be a real time-saver when you are performing a set of similar repetitive operations.

“Check in unchanged files” gives you several options. The default, “Undo Check Out,” is the most efficient in low-bandwidth situations, as it does not try to rewrite the entire source code file if it cannot detect any differences. However, if you have set the file comparison to a quicker and less reliable means (see the discussion of the Local Files tab later in this chapter), you might prefer to set this option to “Check In” or “Ask.” In most LAN situations, this should probably be left at its default.

The “Use visual merge” option asks how often you want to confirm the changes SourceSafe merges quite well by itself. If you are having trouble with Visual SourceSafe’s automatic merge functionality, change this option to “Yes.” The “Double-click on a file” drop-down box lets you specify whether double-clicking on a file in the file list should edit the file, bring the file up in the viewer or, the default behavior, Ask. If you always want the file for editing, or never want the file for editing, consider changing this option to save yourself a mouse click. The “Editor for viewing files” option lets you specify your preferred editor as described in the discussion of the Edit Files dialog earlier in this chapter. The temporary files setting lets you point to a local drive, if writing files to the network location of the SourceSafe database is slowing down operations.

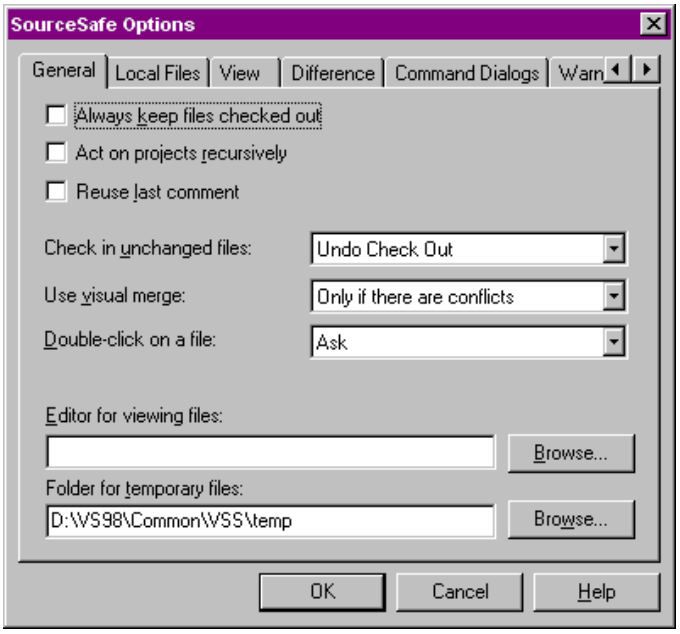

***Figure 45**. The General tab of SourceSafe Explorer’s Tools | Options dialog.*

The Local Files tab of the Tools | Options dialog (see **Figure 46**) offers several options you can use to configure SourceSafe more to your own liking. And, again, the SourceSafe default is usually well chosen, and you should only try changing it if you feel you need that particular change.

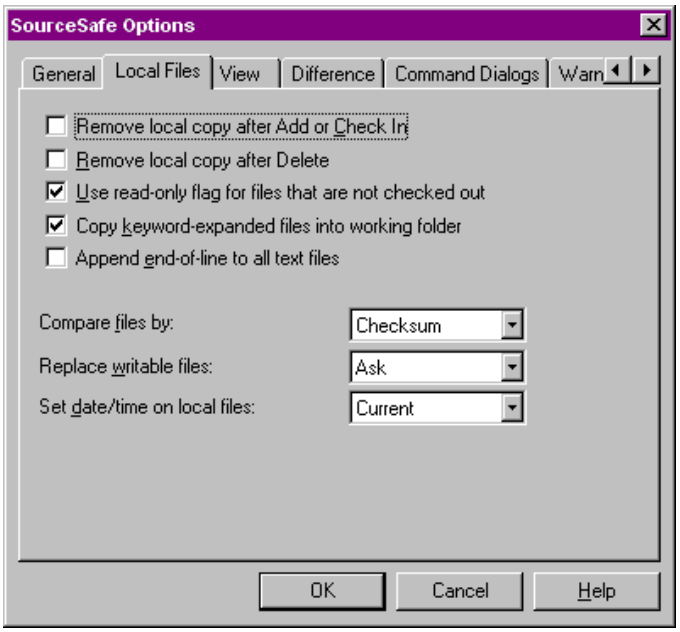

***Figure 46**. The Local Files tab of SourceSafe Explorer's Tools | Options dialog.*

The first two options remove local copies. Use these when you are very limited in space or when you prefer keeping the master copy off your machine. These choices would be appropriate if you were using shadow directories to do the application build. The option for using the read-only flag has two purposes. First, the read-only flag can remind you that you are on the honor system and shouldn't be changing files that you haven't checked out. The second reason is that the overwrite logic of SourceSafe assumes that it is safe to overwrite a read-only file when doing a Get or Check Out. If the file is read-write, SourceSafe takes action based on the settings you have specified. You could find yourself answering many more dialogs than you would prefer if you change this setting.

The next option, "Copy keyword-expanded files into working folder," causes there to be a second file write if a file you have checked in has keywords. SourceSafe reserves a number of words, set off with dollar signs, such as Author, History, Date and so forth, that SourceSafe automatically expands upon check-in. While this makes for some nice automated documentation, it does require another file write over the network. Change this in low-bandwidth situations if it makes sense for you. The last check box lets you specify that the end-of-line character(s) are appended onto the last line of text files if they are missing. Some utilities depend on text lines being terminated with EOL characters and operate incorrectly without them.

The file comparison drop-down box lets you specify how SourceSafe determines that files differ when checking in a file. The most exact is the Contents comparison, which compares each value within the files. This is also the most resource-intensive and tends to be the slowest. The default method, Checksum, calculates a number for both the original file and the file being checked in. It is extremely rare, though theoretically possible, that the two files will generate the same checksum and therefore changes might be discarded. The final

option, Time, depends on the next drop-down box for its behavior. Depending on how local files are time-stamped, a difference between the local file timestamp and the SourceSafe timestamp dictates whether SourceSafe decides these files are different. While very efficient over low-bandwidth connections, the risk of a computer with a wacky time setting upsetting this scheme is too great for my preference.

The final setting determines how files should be time-stamped when they are written locally. At first, it might appear confusing that SourceSafe chooses to set the date and time to the current setting when you get a file. However, if you use a project-building tool like Make or the FoxPro Project Manager, these products decide which files to recompile based on the timestamp of the source code vs. that of the object code. If the source code were to have an earlier timestamp than the object code, no compilation should be needed, and thus, new code you got from the SourceSafe database would not be compiled into your application. While you can change this setting (the other options are to set the timestamp to the modification date or the update date), you would have to compensate for this difference in dates by forcing compilation of all files or use some other technique.

The View tab of the Tools | Options dialog (see **Figure 47**) determines how the SourceSafe interface is presented to you. The first three check boxes let you turn on or off the results pane, toolbar and status bar. If you have ever dragged the toolbar off the application frame and closed it, you will be relieved to know that this is the place (and the only place—it ought to be on the View menu, too) where you can get the toolbar to be displayed.

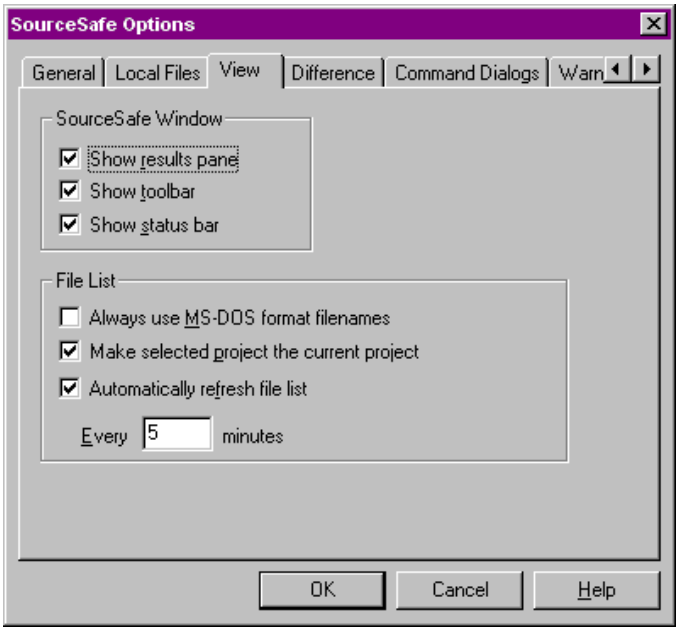

***Figure 47**. The View tab of SourceSafe Explorer's Tools | Options dialog.*

The second set of options force MS-DOS format filenames (necessary only if you are creating an MS-DOS cross-platform product) and allow you to automatically shift the current project with the focus. Finally, there's an option to refresh the file list (F5 or View |

Refresh File List do it interactively) automatically and at the interval you specify. Raise this setting on low-bandwidth or heavily used connections, and lower it if you need to see updates more frequently.

The Difference tab on the Tools | Options dialog (see **Figure 48**) lets you set the characteristics of the Differences window (see **Figure 43**) by specifying the colors, fonts and appearance of each of the difference elements. Also, the check box for "Show line numbers" determines whether line numbers appear in the report preview windows.

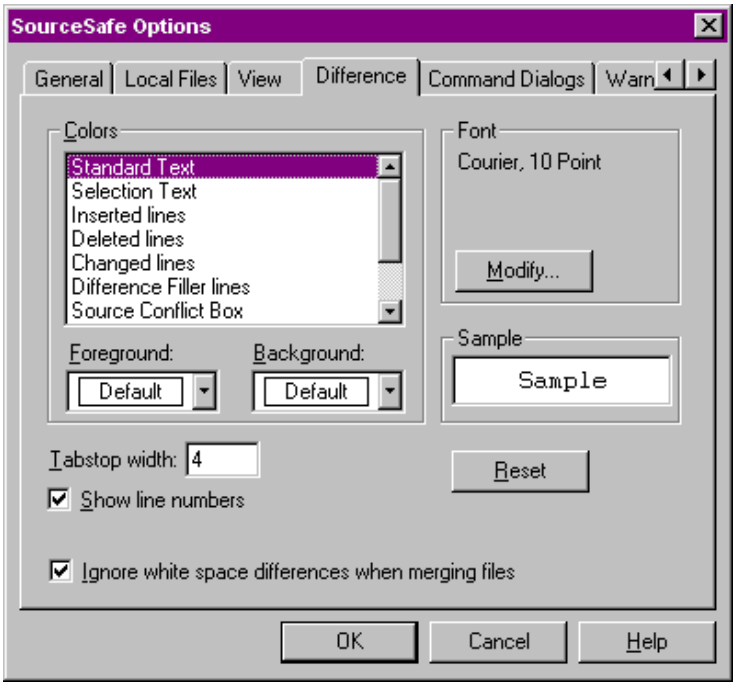

***Figure 48**. The Difference tab of SourceSafe Explorer's Tools | Options dialog.*

The Command Dialogs tab (see **Figure 49**) determines which commands get a dialog popping up with them. You've probably noticed that many of the dialogs have a "Show this dialog only when the Shift key is pressed" check box—this is the place where all of those settings can be made at once.

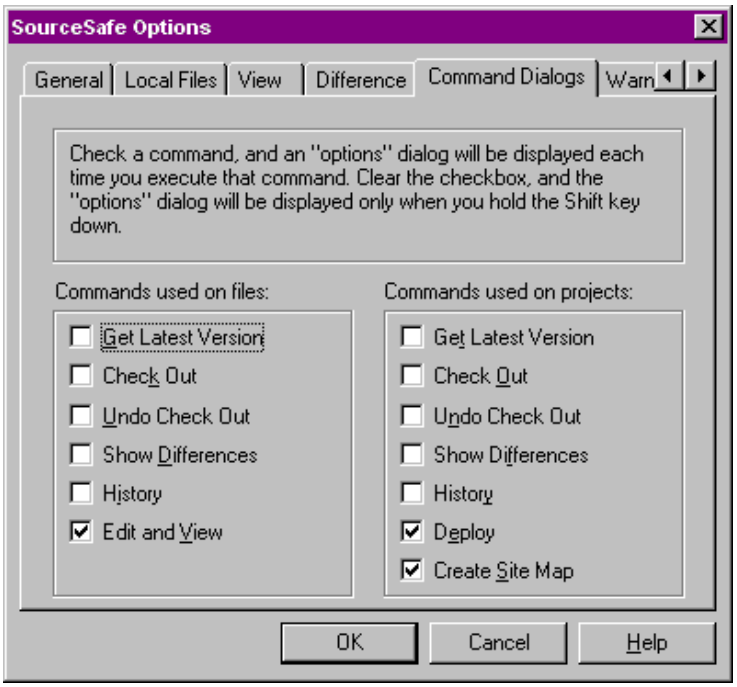

***Figure 49**. The Command Dialogs tab of SourceSafe Explorer's Tools | Options dialog.*

The Warnings tab of the Tools | Options dialog (see **Figure 50**) lets you decide which operations should provide a warning dialog, confirming the operation you want to perform. In all cases, these warnings are letting you know that code is going to be lost, overridden or erased by the operation you've chosen. Unless you are very consistent (or very lucky), you might want to choose to keep these warnings active.

The File Types tab of the Tools | Options dialog (see **Figure 51**) serves several purposes. The "Binary files" text box lists file skeletons that SourceSafe should always treat as binary. This can save a bit of processing time, as SourceSafe does not have to scan the file, as it usually does, to determine whether the file is text or binary. The "Create SCC file" text box specifies file extensions for which SourceSafe should create a matching MSSCCPRJ.SCC (my guess is that's a MicroSoft Source Code Control PRoJect file) to store project information for Visual Basic projects. If you are storing files with those extensions in other languages, and you don't need to store VB projects in SourceSafe, consider dropping the extension from the dialog, and eliminating the unnecessary file.

The final portion of the dialog deals with file groups and their related buttons. File groups are used to populate the Add Files dialog "Files of type" drop-down box (see **Figure 11**) with the appropriate file extensions for each language. The File Types tab allows you to add your own language and matching file types.

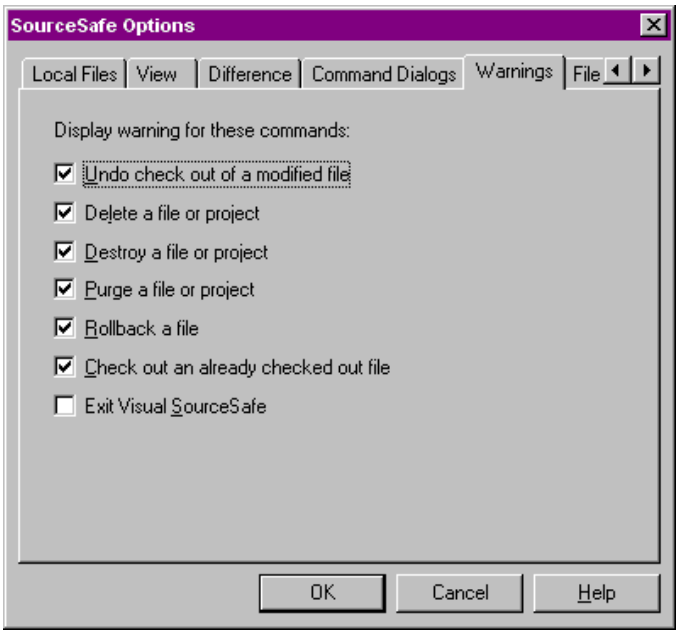

***Figure 50**. The Warnings tab of SourceSafe Explorer's Tools | Options dialog lets you enable or disable many of the warning dialogs. I appreciate the warnings in most cases, and leave them enabled.*

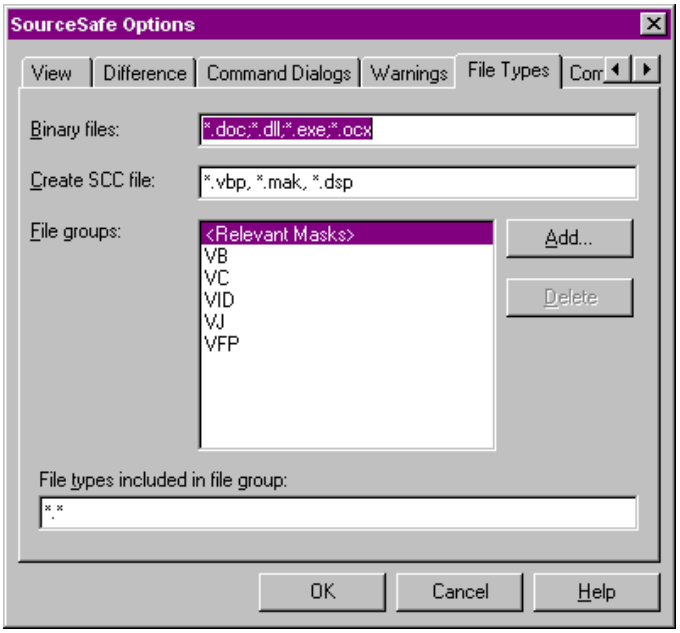

***Figure 51**. The File Types tab of SourceSafe Explorer's Tools | Options dialog.*

The file groups also have a *special* group, named "<Relevant Masks>." This group is the default group used in the Add Files dialog, and the masks here determine which files are used in Project Compare and drag-and-drop operations. If you are restricting your use of SourceSafe to only a few types, consider setting them up here. Add a new file group by selecting the Add button and giving it a name. Add new file extensions to a group by highlighting the group and adding the file skeletons to the text box labeled "File types included in file group."

The last tab of the Tools | Options dialog is the tab for Command Line Options, shown in **Figure 52**. These options take effect only when you're working with SourceSafe from the command line, as explained in Chapter 8, "Beyond the Basics." SourceSafe commands typically take effect in the current directory; the first option uses the project's working folder rather than the current directory. Projects are specified from the command line with the CP (current project) command; they can, instead, be assumed from the current directory matching a project's working folder, if the second option is selected. The first set of option buttons on the tab let you decide whether you want to require a comment on checkout, and whether each file should get its own or whether one comment should be applied to them all. The next set of option buttons determines whether comments are prompted for on the command line or in an editor.

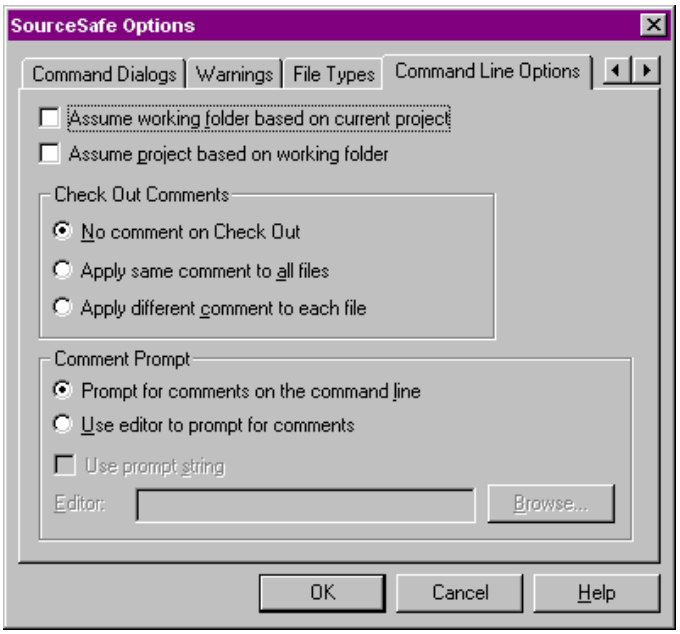

***Figure 52**. The Command Line Options tab of SourceSafe Explorer's Tools | Options dialog.*

Font

This menu option allows you to change the font used to display the project tree view and the file list (see **Figure 53**). This is a really handy option if you need to do presentations with the

SourceSafe interface. My experience is that any font will do for presentations, as long as it's Courier New, 14-point, bold.

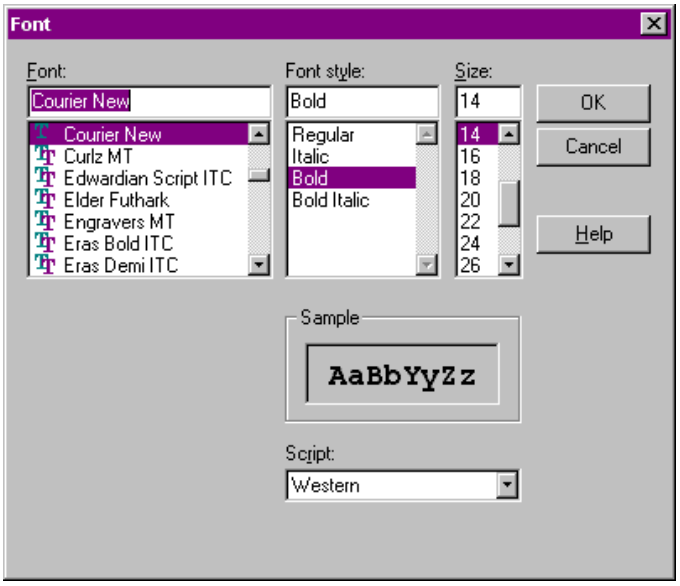

***Figure 53**. Use the Font dialog to adjust the tree view and file lists for your comfort.*

Customize Toolbar

The Customize Toolbar dialog (see **Figure 54**) allows you to add a few of the less commonly used menu options to the toolbar, and also lets you rearrange the order of the buttons on the toolbar. As you become more familiar with the product, take a look at this option to see whether there may be toolbar buttons available for you to add to speed up common tasks.

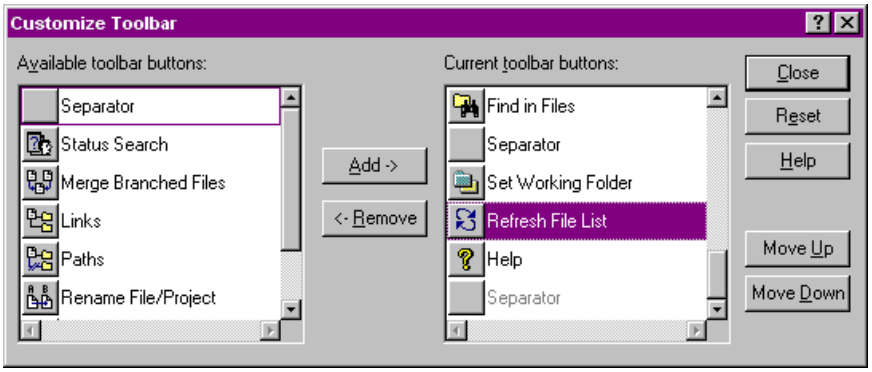

***Figure 54**. The Customize Toolbar dialog lets you add, remove and reorder buttons on the toolbar.*

Change Password

The Password dialog (see **Figure 55**) allows you to change your password. To prevent mischief, you must know your current password in order to change it. If you have forgotten your password, the SourceSafe Administrator can change it.

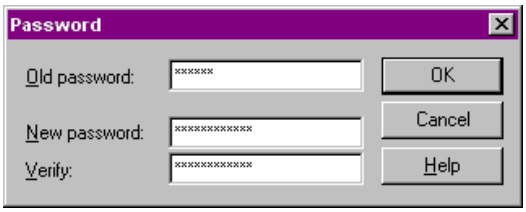

***Figure 55**. Change your password with the Tools | Password dialog.*

The Web menu

The Web menu is a recent add-on. As Microsoft turned its attention to the emerging Internet, all of its products shipped with Web features. While SourceSafe had always been ideal for version control of HTML, because HTML is always plain text, such control still required action outside of SourceSafe to transfer files to and from the Web site. These functions were added to the product.

Deploy

The Deploy dialog (see **Figure 56**) actually lets you upload or transfer a set of files, representing a Web site, directly from SourceSafe to the Web server. The configuration needed to make this happen is done in the Tools | Options dialog of the Administrator executable, covered in Chapter 3, "Configuring Applications." Note that you must have Destroy permissions (the highest level) in order to be able to deploy a Web site. That's not a typo. While no files are destroyed, permission to deploy the Web site should probably be restricted to a small number of users.

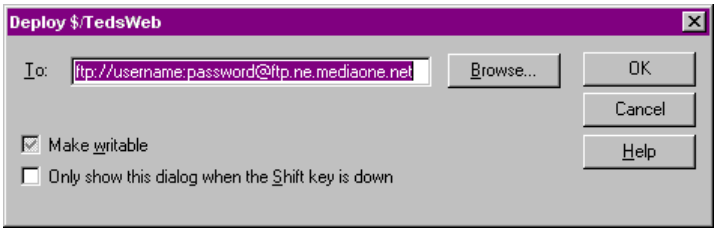

***Figure 56**. You can deploy a Web folder from SourceSafe to the Web using this dialog.*

Check Hyperlinks

The Check Hyperlinks dialog (see **Figure 57**) presents a neat feature—and one that seems a bit odd in the middle of a source code control package. This dialog lets you check files, either in the working folder or in the SourceSafe project's image, to ensure that the files actually exist.

Optionally, SourceSafe will actually go out to the Web to verify that the URIs specified are legitimate as well. This is a great last-minute check before deploying a Web site for the world to see, I suppose, but I still don't understand why it is included in the SourceSafe package.

The result of the Hyperlink check is shown in a Results dialog (see **Figure 58**) that lists all of the documents checked, the hyperlinks found invalid, and those that were ignored. Clicking on each document that was checked populates the lower two list boxes with the list of links. In the case shown here, it appears that SourceSafe does not believe that relative links (those in folders below the documents) can be referred to without a "./" preceding it. When the document was changed to lead with the ./, all links checked out correctly.

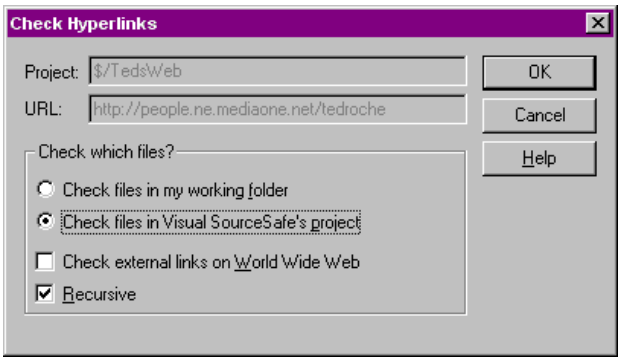

***Figure 57**. Check the links in either the working folder or the SourceSafe project to verify that the Web pages, graphics and URIs exist.*

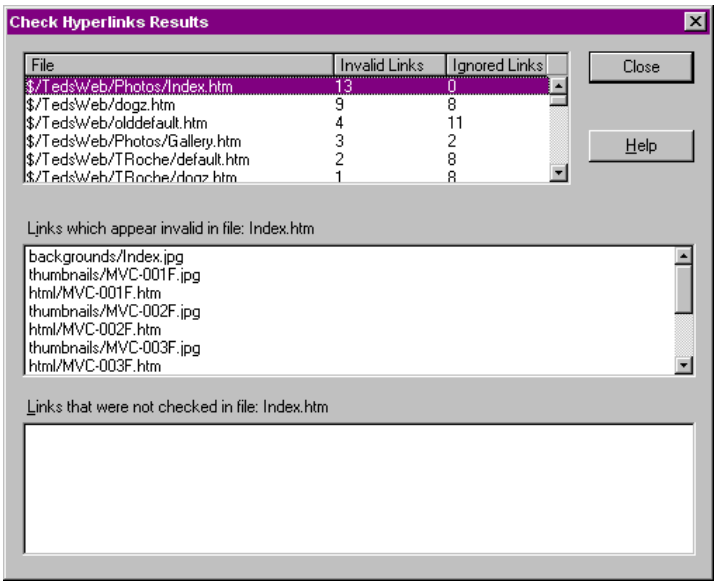

***Figure 58**. The results of Check Hyperlinks show files that couldn't be located, as well as listing those that were not checked.*

Create Site Map

The final option on the Web menu in SourceSafe is the Create Site Map option (see **Figure 59**). This dialog creates a simple HTML page (see **Figure 60**) that lists the contents of the Web project, and displays a tree of the subprojects and their contents as well. The graphics used for the site map—the folder and HTML file icons—can be modified by changing their entries in the SRCSAFE.INI file, as described in Chapter 8, "Beyond the Basics."

***Figure 59**. The Site Map feature allows you to create a list of those files in your SourceSafe Web folder.*

***Figure 60**. The Site Map created from Visual SourceSafe.*