The Next, Best Thing in File Browsers[*]

Michael A. Harrison[†] Thomas A. Phelps[†]

University of California, Berkeley

Abstract

In general, the task of information browsers is (1) to aid in navigating through data bases to locate the desired information, and (2) to examine or otherwise manipulate this information. Heretofore, file selection boxes—which an application uses to request a file name—have had only limited navigation capabilities and almost no file manipulation or inspection capabilities. This paper argues for the need of *file browsers*, for use in conjunction with and apart from an application. This paper introduces NBT, which extends the design of NeXTSTEP's file selection box and sets a new standard for file browsers in the areas of directory navigation and file inspection and manipulation.

1 Introduction

With hundreds or thousands of megabytes of personal disk space augmented by much larger central file servers, the task of locating specific information can be laborious. Once one moves from editing a small number of files in one or two directories and attempts to incorporate images from clip art or data libraries, reference archived documents, or tap centralized resources, one faces the problem of managing information. In the case of a file system, the information is typically organized into a hierarchy, but lacks any further annotation of or links between files, as contrasted to hypertext systems. Current file selection boxes fail their users in each of the scenarios mentioned above, for at the point where the user must make a decision about whether to select a particular file or not, he does not have sufficient information. Forcing the user to choose a particular file, which subsequently can be deleted from the workspace and replaced, is obviously an inelegant, second-best solution. When referring to libraries of information, file names may be too similar to make the correct choice; one would like a general preview mechanism. When referencing archived files that may have been stored in a compressed format, which must be uncompressed before use, one would like to apply directly whatever massaging is necessary to make the file acceptable to the application. And when searching large repositories of information (as on a central server) one would like an efficient means of navigating the large directory hierarchy, annotating selected locations for future reference.

Succinctly, the problem of information management as applied to file browsing is that of (1) establishing location quickly and efficiently, and (2) working there (e.g., viewing, searching, printing, unarchiving), for after having invested the effort of establishing a context one should not be forced to a separate "work shell" to accomplish the task.

Part of the Ensemble Multimedia Editor of structured documents project [GHM92], NBT addresses these concerns by bringing together various means of navigation, providing an extensible means of viewing and manipulating files, and learning from user interaction to customize itself. NBT is useful for navigating and inspecting files apart from or in conjunction with selecting them for use in a application. NBT is the next, best thing in file browsers.

2 Navigation

File selection boxes for graphical user interfaces have existed for at least a decade. Yet even today, many X Windows and even Sun's OpenWindows applications merely present a line to the user on which he is expected to type the full, often lengthy path name of the desired file [NeX92]. Motif [Ope91] and some X and Tcl/Tk programmers have improved on this by offering a point-and-click (no typing) mechanism.

NBT consolidates the best features of file selection boxes from the Macintosh and NeXT, which it resembles, as shown in Figure 1. Like NeXTSTEP, NBT presents a contextual view of the current directory and the n previous ones in the current path. One may navigate to nearby directories by clicking on directory names to descend into that directory, or by clicking on the left arrow at the leftmost edge to move up. As on the Macintosh, NBT's folder icon in the upper-left lists all the directories in the current path in a pulldown menu; releasing the mouse button over any component jumps to that location. Further, the user may save commonly-accessed directories on the Shortcuts

[*]This research has been sponsored in part by the Defense Advanced Research Projects Agency (DARPA) under Contract N00039-88-C-0292, monitored by Space and Naval Warfare Systems Command, and under Grant MDA972-92-J-1028. The content of the information in this paper does not necessarily reflect the position or the policy of the Government. Thanks are due to Vance Maverick, Brian Dennis, Kannan Muthukkaruppan, and Wayne Christopher, all of whom reviewed drafts of this paper.

[†]Computer Science Division, 571 Evans Hall / University of California, Berkeley / Berkeley, CA 94720. E-mail addresses: {harrison, phelps}@cs.berkeley.edu.

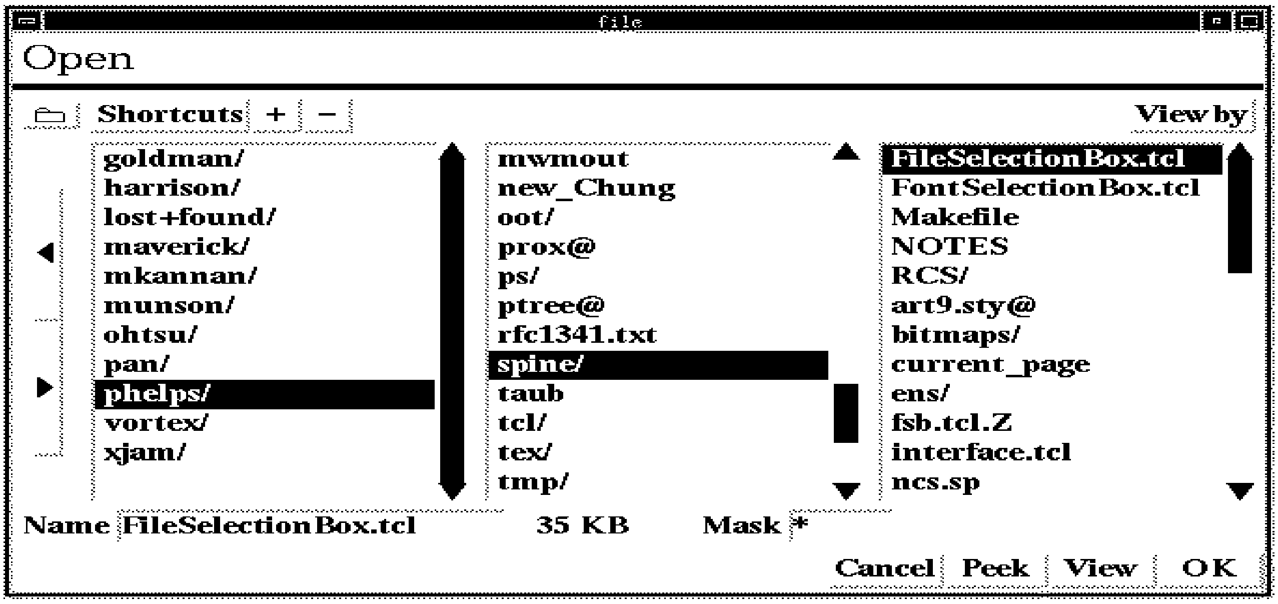


Figure 1: **Navigation.** Based on the NeXT's File Viewer, NBT shows previous directories as context and allows direct access to stored directories through the Shortcuts menu. The folder icon at top left implements Macintosh-like access to any part of the current path. Keyboard navigation is also available.

menu. Clicking '+' adds the current directory to the list and '-' removes it; subsequently selecting this entry from the Shortcuts pulldown jumps to that location. Thus, NBT brings together and gives direct access to three modes of navigation: local (a single directory level up or down), global linear (any number of directories up, but only along the current path), and global (to any location in the full hierarchy, so long as it has been previously saved).

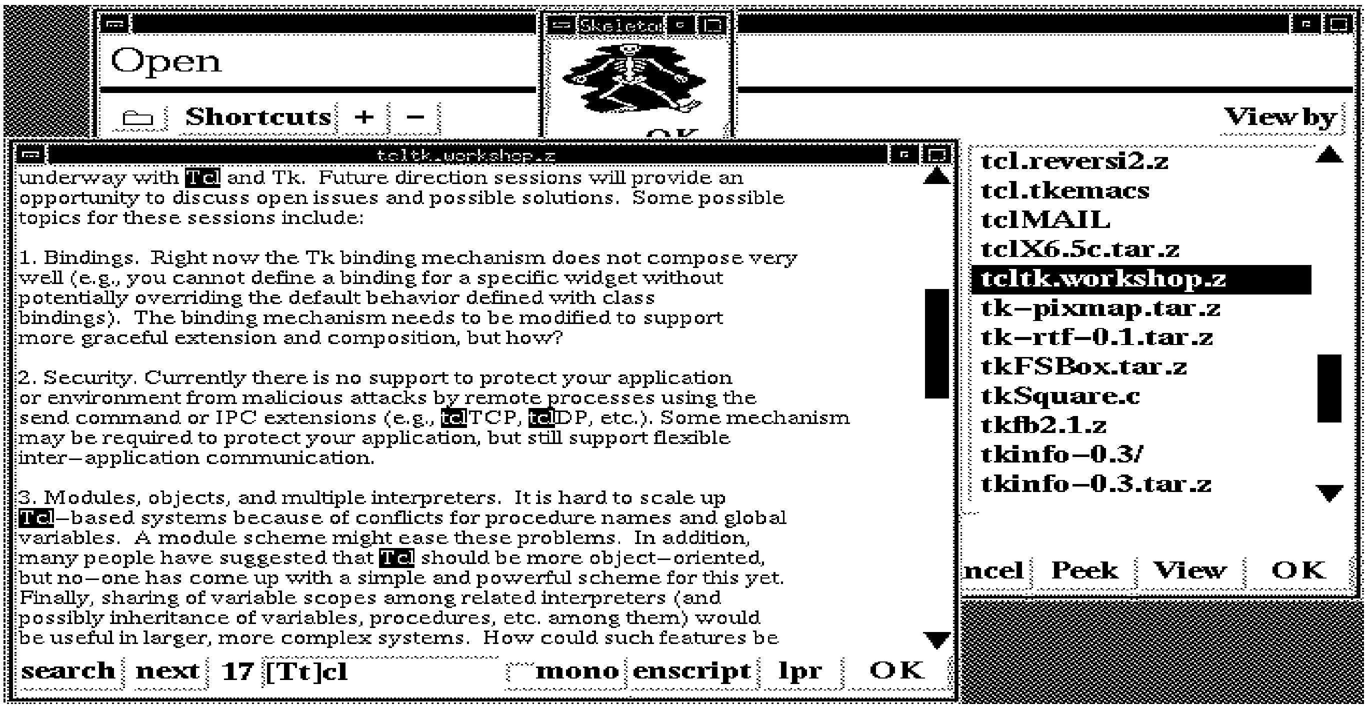
Finally, consider the following scenario. In putting together a newsletter, a user must repeatedly switch between a photos directory and a story text directory. Most present file selection boxes present the user with the previously-selected directory the next time it is invoked, which in this case maximizes inconvenience. With NBT, one could save the two directories as Shortcuts and switch between them quickly. However, NBT takes a further step by keying the default directory to the application-supplied title, so that a request from the main application to NBT to, say, "Import Bitmap" presents the last directory accessed with this message, which may be very different than the "Import Text" directory.

3 Inspection and Manipulation

Having located a particular file with the navigation facilities described above, one must decide what to do with it. One may (from a 'Utilities' palette, not illustrated) execute simple commands that apply to a single file, for example deleting, unarchiving and uncompressing. For file inspection, NBT gives both "raw" and "semantic" views, both of which are illustrated in Figure 2.

Upon selecting a file, users may 'Peek' at it or 'View' it. 'Peek'ing shows the raw ASCII contents of a file, along with file meta information, e.g., instance creation time and file size. As in Emacs [Sta87], one may search the contents incrementally, character by character, either forward or backward. Regular expression searching is available in a text entry line at the bottom of the window. One may print out this information through a paper-conserving filter (`enscript`) or raw (`lpr`, useful for PostScript files), or select a region and copy it to the X clipboard. 'Peek' is useful for lengthy files in which only the header contains any human-decodable information (e.g., PostScript sources). Both 'Peek' and 'View' (described below) automatically negotiate common (e.g., compressed) formats for saved files.

'View'ing tries to determine the file's type by consulting a user-extensible suffix-to-type association list. If nothing matches, its full ASCII contents are shown. If a match is detected, the file is viewed in its "semantic" form, for instance a PostScript file is shown in a WYSIWYG view. Semantic matches may either (1) start up a process which is given the full path name of the file to view (e.g., `.ps` files start up a PostScript previewer such as Ghostscript and `.au` sound files are played on the speaker, if any), (2) use Tcl's `send` command to communicate this information to a running Tcl interpreter (e.g., `.dvi` files call a Tcl/Tk-aware `dvi` previewer [Phe93]), or (3) invoke custom Tcl (and hence C) code (e.g., bitmaps are shown on a `canvas` widget). The most complex example of a custom viewer is that for `tar` archive files. The semantic view shows that `tar`'s table of contents; selecting an entry (perhaps `README`) and clicking 'extract' retrieves the file from the (possibly compressed) `tar` archive, closes the semantic view, and selects the newly-extracted file so that a single



underway with Tcl and Tk. Future direction sessions will provide an
opportunity to discuss open issues and possible solutions. Some possible
topics for these sessions include:

1. Bindings. Right now the Tk binding mechanism does not compose very
well (e.g., you cannot define a binding for a specific widget without
potentially overriding the default behavior defined with class
bindings). The binding mechanism needs to be modified to support
more graceful extension and composition, but how?

2. Security. Currently there is no support to protect your application
or environment from malicious attacks by remote processes using the
send command or IPC extensions (e.g., tclTCP, tclDP, etc.). Some mechanism
may be required to protect your application, but still support flexible
inter−application communication.

3. Modules, objects, and multiple interpreters. It is hard to scale up
Tcl−based systems because of conflicts for procedure names and global
variables. A module scheme might ease these problems. In addition,
many people have suggested that Tcl should be more object−oriented,
but no−one has come up with a simple and powerful scheme for this yet.
Finally, sharing of variable scopes among related interpreters (and
possibly inheritance of variables, procedures, etc. among them) would
be useful in larger, more complex systems. How could such features be

tcl.reversi2.z
tcl.tkemacs
tclMAIL
tclX6.5c.tar.z
tcltk.workshop.z
tk−pixmap.tar.z
tk−rtf−0.1.tar.z
tkFSBox.tar.z
tkSquare.c
tkfb2.1.z
tkinfo−0.3/
tkinfo−0.3.tar.z

Figure 2: **Inspection.** Here we are 'View'ing a "semantic" view of a bitmap (the skeleton) and 'Peek'ing at the raw ASCII of another file (in this case the workshop announcement, which is stored on disk in a compressed form). In the Peek window, seventeen matches of the regular expression `[Tt]cl` have been found and those visible in the current region are highlighted.

click on 'View' will show it. Thus, viewing a PostScript file buried deep within a compressed tape archive file is only a few clicks away.

4 Conclusions

To choose a file from a complex file system, a user needs to navigate the directory structure and inspect or otherwise manipulate files of multiple media types before making a choice. This necessity is growing ever more urgent as file systems grow, incorporating large central libraries of files. NBT sets a standard of capabilities as a file browser useful apart from an application, but which should always be available in conjunction with an application as well.

References

[App87] Apple Computer, Inc. *Human Interface Guidelines: The Apple Desktop Interface.* Addison-Wesley, Reading, Massachusetts, 1987.

[GHM92] Susan L. Graham, Michael A. Harrison, and Ethan V. Munson. The proteus presentation system. *Proceedings of the Fifth ACM SIGSOFT Symposium on Software Development Environments*, pages 130–138, 1992.

[NeX92] NeXT Computer, Inc. NeXT versus Sun: a comparison of development tools. Technical Report Seven in the NeXT Computer, Inc. White Paper Library, NeXT Computer, Inc., Redwood City, CA, January 1992.

[Ope91] Open Software Foundation. *OSF/Motif Programmers Reference.* Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[Phe93] Thomas A. Phelps. `dvi2x`: A Tcl/Tk-based previewer for TEX. *Ensemble Working Paper*, 1993. To appear.

[Sta87] Richard M. Stallman. *GNU Emacs Manual, Sixth Edition, Version 18.* Free Software Foundation, Cambridge, MA, March 1987.