10.0 File Tool

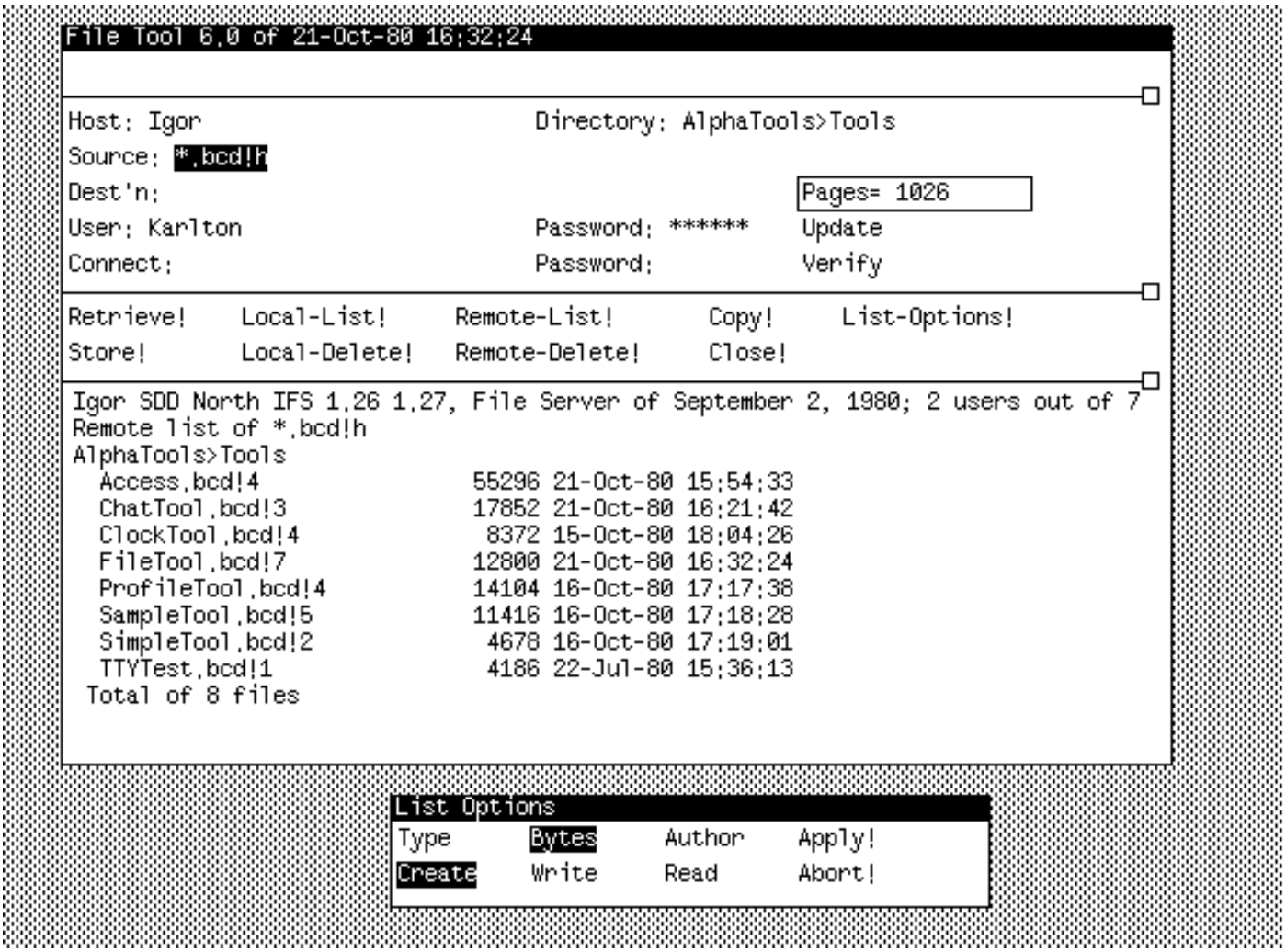
The File Tool provides a means of dealing with local as well as remote file systems from within the Development Environment.

10.1 The User Illusion

The File Tool employs the standard features of the Development Environment. See section 3 for further details.

10.2 Tool Appearance

Below is an illustration of a File Tool with the List Options window (explained below) visible.



10.3 Parameter Subwindow

The upper form subwindow contains parameters that can be set by the user; they will be used by the next File Tool command.

Host: the name of the host to be used for remote file operations. If a connection is already open, any editing of this field causes it to be closed; if a transfer is in progress, the connection will not be closed until it is complete.

Directory: the default remote directory. If empty, the value in the User: field is used.

Source: a list of files (separated by spaces or returns) to be operated on. If the first character of a file name is "@", then the file is taken to be an indirect file and its contents are used as a list of files. Indirect files may nest.

Dest'n: file name for the destination of a transfer. If this field is left blank, the file name is the same as the source.

Pages= number of free pages left on the disk. This item is read only.

User:, Password: the primary directory and the associated password. This field is initialized from the value of the user's last Alto Operating System login. Editing of this field is local to the File Tool and does *not* affect the user's login in the Alto Operating System.

Update only store or retrieve the file if the source is newer than the destination (comparing creation dates). The default is false.

Connect:, Password: the secondary directory and the associated password.

Verify request confirmation for each file operation. The default is false.

10.4 Command Subwindow

File Tool commands are available in the second form subwindow. Some of the commands are accomplished by a background process. Those commands clear the Command subwindow so that a second operation cannot be invoked while one is under way. The Copy! command operates only on the local disk. It does not take a list of files to operate upon. Close! closes a remote connection (if there is one).

It is important to remember that the commands are postfix; e.g., fill in the Host: and Source: fields before invoking the Retrieve! command. The following commands are available:

Retrieve! transfers the file specified in Source: from the remote file system to the local disk. The file name must conform to the file-naming conventions on the remote host. You may designate multiple files by the use of * expansion only to the extent that the remote server supports it. If the local file is already in use, the transfer will not be made and the message "<filename>: can't be modified" will be displayed in both the message window and the log window. See warning in Section 10.6

Local-List! lists all files on the local disk corresponding to the name in Source:. This command will expand *s and #s.

Remote-List! lists all files on the remote file system corresponding to the name in Source:. This must conform to the file naming conventions of the remote host. You may designate multiple files by the use of * expansion only to the extent that the remote server supports it (currently Maxc and IFS do, but differently).

Copy! makes a copy of a local file on the local disk. Only a single file may be copied and *s and #s are not allowed.

List-Options! creates a List Options window if one does not already exist.

Store! transfers the file specified in Source: from the local disk to the remote Host. Alto file name conventions apply to the local file.

Local-Delete! deletes the files specified in Source: from the local disk. If the local file is already in use, the delete will be skipped and the message "<filename>: can't be modified" will be displayed in both the message window and the log window. See warning in Section 10.6

Remote-Delete! deletes the file specified in Source: from the remote file system. You may designate multiple files by the use of * expansion only to the extent that the remote server supports it.

Close! closes the currently open FTP connection.

If Verify is TRUE, then for each file that might be acted upon, the following commands are displayed

Confirm! do the operation.

Deny! abort the operation.

Stop! abort the operation and terminate the command. This will close the connection with the server if a retrieve is being aborted.

10.5 List Options window

The List Options window is created by the List-Options! command. The properties that will be displayed, in addition to the file name, by a Local-List! or Remote-List! are governed by the booleans in this window. After changing the options, use Apply! to effect those changes. The Abort! command will restore the options which existed before the List-Options! command was selected. Choosing either of the commands in the List Options window will cause that window to be removed.

If the Type attribute is requested for a Local-List! and the type is unknown, it will be listed as such to prevent the time it would take to read the file and determine the type.

10.6 Exceptions

The actual transfer takes place in a background process, so the user is free to issue other commands or even change the values in the parameter subwindow without affecting the command currently executing. Changing a parameter while the File Tool is waiting for Confirm! will **not** affect the name of the destination file; you should skip the transfer (by using Deny!) and reissue the command with the desired parameter correctly set.

Warning: Do not attempt to use a file while it is being retrieved. This includes issuing commands to the Debugger that cause it to try to reference the file. For example, Display Stack may cause the Debugger to reference symbols contained in the file being retrieved.

Warning: If you are using the File Tool in the Debugger, be careful not to change any files out from under the program you are debugging; the file tool makes no provisions for checking if the file is in use *in the client world* when you modify a local file.