Both of these routines return non-zero on success and zero otherwise.

There are two routines used to fetch a particular entry in the database. The routine `getisodocumentbyentry` maps textual descriptions into the internal form.

```
struct isodocument *getisodocumentbyentry (entry)
char   *entry;
```

The parameter to this procedure is:

`entry:` the descriptor of the object.

and returns the `isodocument` structure describing that document. On failure, the manifest constant `NULL` is returned instead.

The routine `getisodocumentbytype` performs the inverse function.

```
struct isodocument *getisodocumentbytype (tpye)
OID     type;
```

The parameter to this procedure is:

`type:` the identifier of the document.

On a successful return, an `isodocument` structure describing the document is returned.

Chapter 4

UNIX Implementation

The File Transfer, Access, and Management (FTAM) standard is the OSI file service. Included in the release is a fairly complete FTAM implementation in the context of the particular file services it offers. It is a minimal implementation in as much as it offers only four core services: transfer of binary files, transfer of text files, directory listings, and file management. The implementation included has been tested on both Berkeley and AT&T SVR2 and SVR3 UNIX. Both the FTAM initiator and responder programs have UNIX manual entries.

4.1 Implementation

If you have access to the source tree for this release, the directory *ftam2/* contains the code for the responder and initiator.

4.1.1 The Initiator

There is currently one initiator which uses FTAM: *ftam*(1c). Supported are: the no-recovery FTAM-QoS; any of the transfer, management, and transfer and management service classes; the kernel, read, write, limited file management, enhanced file management, and grouping functional units; and, the kernel and storage attribute groups. Only three document types are supported as of this writing: unstructured text files (FTAM-1), unstructured binary files (FTAM-3), and filedirectory files (NIST-9).

The *ftam* program is an interactive FTAM initiator which prompts the user for commands. Generating an interrupt, usually by typing control-C (‘^C’), at the top-level does nothing, but generating an interrupt twice in a row at the top-level terminates *ftam*; generating an interrupt during additional prompting causes *ftam* to abort the command; typing generating an interrupt during file transfer causes the transfer to be aborted.

Commands

Here are the commands to *ftam*:

append `source destination` Appends to a file in the filestore.

cd `[dir]` Changes the working directory on the virtual filestore. This requires the **realstore** variable to be set appropriately.

chgrp `group file ...` Changes the account attribute of the named files.

close Terminates the association with the virtual filestore.

dir `[file]` Prints a long directory listing.

echo `file ...` Simply echoes any arguments. Useful for seeing how globbed expressions will evaluate.

fdir `stream [file]` Prints a long directory listing to a file or program. If `stream` starts with a vertical bar (‘|’) then the named program is invoked; otherwise the named file is written.

fls `stream [file]` Prints a directory listing to a file or program. If `stream` starts with a vertical bar (‘|’) then the named program is invoked; otherwise the named file is written.

get `source destination` Retrieves a file.

help `[command]` Prints help information. For detailed information, try “`help␣?`”.

lcd `[file]` Changes the working directory on the local system.

ls `[file]` Prints a directory listing.

mkdir `dir ...` Creates a directory.

mv `source destination` Renames a file.

open `host user [account]` Associates with the virtual filestore.

put `source destination` Stores a file.

pwd Prints the working directories.

quit Terminates the association with the virtual filestore and exits.

rm `file ...` Deletes a file.

set `variable value` Displays or changes variables. For detailed information, try "`set␣?`".

status Shows the current status.

Variables

Here are the variables which effect *ftam*'s behavior.

bell Rings the bell after each command terminates. Useful for long file transfers when you want to attend to other matters and be notified when you can type another command. Boolean (values: **on** or **off**).

debug This enables voluminous output during file transfers, among other things. Boolean.

glob This enables the expansion of shell meta-characters. Operations which perform globbing require the **realstore** variable to be set appropriately. Boolean.

hash This enables the printing of hash marks during file transfers. Values: `off`, `on`, `total`.

override This sets the creation override mode for files being written to the virtual filestore. If the file being created already exists, then one of four alternatives is taken. Values:

`fail:` the creation operation;

`select:` use the existing file with its old contents and attributes;

`write:` zero-truncate if it already exists, and use the existing file with its old attributes; and,

`delete:` if it already exists, then create a new file with new attributes.

This defaults to `write`.

qualifier This sets the "qualifier" portion of the srevice which *ftam* will associate with. It is needed when using the current implementation of the MITRE FTAM/FTP gateway. This defaults to `filestore`.

query This determines if *ftam* should ask the user to confirm operations involving globbing that expand to more than one filename. Boolean. This defaults to `on`.

realstore Sets the type of remote realstore associated with the virtual filestore. This is used to help *ftam* act friendlier to the user! Values: `unix`, `unknown`.

> **NOTE:** The concept of a **realstore** is contrary to the notion of open systems as it is an $N * M$ (not $N + M$) method.

trace This enables the tracing of FTAM PDUs. Boolean.

tracefile This defines the file where tracing information is appended.

type This defines the file transfer mode to use. Values: `default`, `binary`, and `text`.

verbose This enables printing of informative diagnostics during operation. Boolean.

watch This enables watch mode, something in between debug mode (too voluminous), and verbose mode (not informative enough). Boolean.

userdn This defines the Distinguished Name to be used when binding to the Directory for AE-lookup. DN-string.

*xyz***sapfile** This defines the file where *xyz*PDU tracing information is appended. Values: any filename, or **-** for the diagnostic output.

*xyz***saplevel** This enables tracing of the *xyz* module.
Values: `none`, `exceptions`, `notice`, `pdus`, `trace`, and `debug`.

Options

Here are the command line options:

-a *acct* Sets the account to be used on the virtual filestore.

-d Sets **debug**.

-f Inhibits reading of the user's *$HOME/.ftamrc* file on startup.

-h Sets **hash**.

-o *mode* Sets **override**.

-t Sets **trace**.

-u *user* Sets the initiator identity to be used on the virtual filestore.

-v Sets **verbose** (default for interactive use).

-w Sets **watch**.

4.1.2 The Responder

The *ftamd*(8c) program implements the file service. It implements *filestore* abstractions directly on the UNIX filesystem. Supported are: the no-recovery FTAM-QoS; any of the transfer, management, and transfer and management service classes; the kernel, read, write, limited file management, enhanced

file management, and grouping functional units; and, the kernel and storage attribute groups. Only three document types are supported as of this writing: unstructured text files (FTAM-1), unstructured binary files (FTAM-3), and filedirectory files (NIST-9).

Authentication

An FTAM initiator must be listed in the *passwd*(5) file and have a non-empty password. Further, as with the *ftpd*(8c) daemon, the username must not appear in the *ftamusers* file in the ISODE `ETCDIR` directory or in the */etc/ftpusers* file. (In fact, many of the mechanisms in *ftamd* are based on the *ftpd* program supplied with Berkeley UNIX.)

If the username `ANON` or `ftp` is given, then *ftamd* treats this as a guest access, similar to the "anonymous" facility supported by the *ftpd* daemon. An entry in the */etc/passwd* file for user `ftp` must be present with a non-zero UID. For guest access, a *chroot*(2) to the guest home directory is executed to restrict access to the system.

> **NOTE:** The anonymous account is inherently dangerous and should be avoided when possible. It is also inherently useful.

The Berkeley UNIX version of this program runs with the effective UID of the FTAM initiator, but also with the real UID of the super-user. This is necessary to change the account attribute on files using *chown*(2). The possible security holes have been extensively considered, but may be incomplete.

The AT&T UNIX version, which lacks kernel support for this technique, acts differently. Immediately upon association establishment, it changes both the real and effective UID to that of the FTAM initiator. To change the account attribute on files, it invokes the *chgrp*(1) program. Similarly, to create or delete directories, it invokes either the *mkdir*(1) program or the *rmdir*(1) program. Finally, it is unable to change the filesize attribute to a non-zero value if this value is smaller than the current filesize.

Finally, on Berkeley UNIX systems, the *wtmp*(5) file is updated as appropriate. (We couldn't figure out how to update *wtmp* under AT&T UNIX using the description in the SVID!)

Virtual Filestore

Here are the file attribute mappings. Most attributes are derived by doing a *stat*(2) on the file and then examining the indicated field in the resulting structure.

filename A single component, relative to the user's *$HOME*. Changing this attribute is equivalent to a *rename*(2).

contents-type Based on the `st_mode` field:

NIST-9 for directories;

FTAM-1 for regular files appearing to be textual; and,

FTAM-3 for all other regular files.

Files that are neither regular nor directories are inaccessible via this implementation of the VFS (i.e., special files).

account The `st_gid` field according to *group*(5). Changing this attribute is equivalent to a *chgrp*(1).

date-and-time-of-creation The `st_mtime` field.

date-and-time-of-last-modification The `st_mtime` field.

date-and-time-of-last-read-access The `st_atime` field.

date-and-time-of-last-attribute-modification
The `st_ctime` field.

identity-of-creator The `st_uid` field according to *passwd*(5).

identity-of-last-modifier The `st_uid` field according to *passwd*(5) (if the value of the `st_mode` field guarantees uniqueness).

identity-of-last-reader The `st_uid` field according to *passwd*(5) (if the value of the `st_mode` field guarantees uniqueness).

identity-of-last-attribute-modifier The `st_uid` field according to *passwd*(5) (if the value of the `st_mode` field guarantees uniqueness).

file-availability Immediate.

permitted-actions Depends on the `st_mode` the as interpreted by *access*(2): `R_OK` for permission to read; `W_OK` for permission to write; permission is always granted to read attributes; permission is granted to change attributes if the initiator has uid equal to the `st_uid` field; and, permission to delete is based on writability of parent directory.

filesize The `st_size` field.

future-filesize Not available.

access-control Not available.

encryption-name Not available.

legal-qualifications Not available.

private-use Not available.

The activity attribute mappings are straight-forward. The read action corresponds to reading UNIX files. The insert, replace, extend, and erase actions correspond to writing UNIX files. Concurrency control is supported for reading and writing, but not for reading or changing attributes, or for deleting files.

Chapter 5

FTAM-FTP gateway

The FTAM/FTP gateway is an application-gateway for file service. The gateway is actually two programs: one which acts as an FTAM responder and an FTP client, and the other which acts as an FTP server and an FTAM initiator. Note that the gateway currently resides on a different location than the standard FTAM responder and FTP server.

The implementation included runs only on Berkeley UNIX.

5.1 Implementation

If you have access to the source tree for this release, the directories *ftam-ftp/* and *ftp-ftam/* contains the code for the two programs.

5.1.1 The FTAM/FTP side

The FTAM/FTP side of the gateway appears to implement the responder side of the FTAM service, but actually acts as an FTP client in order to provide this service.

The true destination is encoded in the user name (i.e., `user@tcphost`).

Note that the FTAM/FTP side is available on a different presentation address than the FTAM service on the gateway host. To select the FTAM/FTP side, tell your FTAM initiator to associate with the service having "qualifier" `ftpstore` on the gateway host. For example, using *ftam*(1c):

```
% ftam
```