NAME

Source − browsing the Exodus Storage Manager source

DESCRIPTION

This document briefly describes the arrangement of the Storage Manager source files and directories.

INTRODUCTION

The Exodus Storage Manager (SM) comprises applications, clients, and a server. An application is code written by a person who wishes to use the SM. A client is a portion of the SM code that is linked with the application. The client and application make all decisions regarding the management of data and the organization of data into **objects,** and the layout of objects on **pages** and in **files.** Clients request locks and pages, create indices, initiate transactions, and log activity, among other things. The server manages pages, locks, transactions, disk I/O, concurrency, and data recovery. The server performs disk I/O in parallel by delegating the task to subordinate processes.

The source code is divided into **modules** that roughly correspond to the divisions of labor described in the previous paragraph. First, the source is divided into the following directories:

client
: Source code and object files that are only in the client.

common
: Source code and object files that are linked into both the client and the server.

config
: Configuration files for the client and server, containing compiled-in configuration information.

include
: Source code header files that are common to the client and server.

server
: Main() for the server process. The server process is linked in this directory.

serverlib
: The majority of the code for the server is here. These objects are also linked into the volume formatter.

diskproc
: The source and objects for the disk I/O processes.

formatvolume
: Source and objects for the volume formatter, **formatvol.** Formatvol links in the server library also.

admin
: Miscellaneous administrative utility programs.

MODULES

The client and server are further divided into modules. The modules correspond to directories in client/, in serverlib/, and in many cases, in both. In most cases, the client and server perform different tasks under the aegis of the same module. For example, in the **lm** (lock manager) module, the client requests and caches locks, while the server performs the lock management, including deadlock detection, and it denies or grants the client's request.

bf/
: Buffers. (Both.) Both the client and server contain buffer pools. These modules manage page caching in the buffer pools.

bm/, bitmap/
: Bitmaps. (Server only.) The server has bitmaps to determine what disk pages are in use. The module **bitmap** contains low-level routines used by **bm.**

client/sm/
: Storage Manager interface. (Client only.) This module implements the application's interface to the SM.

client/smint/
: (Client only.) This module contains functions called from functions in ../sm.

cl/ Client interface. (Server only.) This module implements the server's interface to the client.

clog/, log/
: Logging. Both the client and server log their activities so that recovery can be performed. **clog/** is the client's logging module. **log/** is the server's logging module.

fi/ File operations. (Both.) This module contains routines for creating and manipulating files (collections of objects).

lg/ Large objects. (Both.) In the client, this module contains code that pertains to large objects (objects larger than 4K bytes). In the server, this module is code that is needed to recover (redo or undo transactions on) large objects.

lm/ Lock manager. (Both.) In the client, this includes requesting and freeing locks. In the server, this includes creating and destroying locks, converting locks, and detecting deadlocks due to granting of locks.

trans/
: Transaction manager. (Both.) The client initiates transaction operations (begin, commit, abort) by passing the requests along to the server. The **server/trans** module is the entire transaction manager.

msg/ Communication between client and server. (Both.)

io/ Page I/O.

btree/
: Indices. (Both.) There is one copy of the source (kept in client/btree), but the objects differ between the server and the client, having been compiled with different C preprocessor flags.

lhash/
: Linear Hashing-based indices. (Both.) There is one copy of the source (kept in client/lhash), but the objects differ between the server and the client, having been compiled with different C preprocessor flags.

recover/, analyze/, redo/, undo/
: Recovery. (Server only.) The modules **analyze/, redo/,** and **undo/** are passes 1, 2, and 3 of recovery, which is performed upon initialization of the server process.

common/, util/, resource/
: Glue. (Server only.) These modules contain code for miscellaneous routines needed in the server.

thread/, sync/
: Coroutine support. (Server only.) The server performs many tasks in parallel. **threads/** implements lightweight threads, and manages synchronization of the activities of the various threads. Threads are implemented with setjmp() and longjmp(), and do not offer preemptive scheduling.

disk/ Communication between the server process and the disk I/O processes. (Server only.)

mr/ Terminal interface and media recovery (media recovery is not implemented). (Server only.)

MAKEFILES

Myriad makefiles are used to make the storage manager. In every directory is a file called **Makefile,** which includes various files, **makefile.*** as need be.

SEE ALSO

Makefile – making an Exodus Storage Manager