NAME

snobol4dbm − SNOBOL4 NDBM interface

SYNOPSIS

-INCLUDE 'ndbm.sno'

dbhandle = DBM_OPEN(file,flags,mode)

DBM_CLOSE(dbhandle)

DBM_STORE(dbhandle,key,datum,flags)

datum = DBM_FETCH(dbhandle,key)

DBM_DELETE(dbhandle,key)

key = DBM_FIRSTKEY(dbhandle)

key = DBM_NEXTKEY(dbhandle)

DBM_ERROR(dbhandle)

DBM_CLEARERR(dbhandle)

DESCRIPTION

"NDBM" (for New Data Base Managemnt) is an industry standard fast hashed storage API first created in 4.3BSD, and included in the Unix 98 (SUSv2) standard. The original DBM API appeared in AT&T Research Unix Version 7, and only allowed access to a single file at a time.

There are many different implementations of this API, including:

- The original BSD 4.3 ndbm, based on AT&T dbm. Found in commercial Un*x offerings.
- Berkeley DB v1 compatibility interface. Supplied with 4.4BSD based systems: (Free|Open|Net)BSD, MacOS X.
- GNU DBM (GDBM) found in Linux distributions.
- SDBM, Ozan Yigit's Public Domain implementation of NDBM. Supplied with this distribution, and used as a last resort on Un*x systems (and by default on non Un*x systems).

DBM_OPEN takes a filename (STRING), flags (either "R" for read-only, "W" for write access, or "CW" to create and write a new file), and a "mode" string, which defaults to "0660" (octal) and returns a database handle which can be passed to the remaining functions.

DBM_CLOSE closes the database file. **DBM_CLOSE** *MUST* be called to ensure that all your data is written.

DBM_STORE takes a database handle, key and datum strings, and a flag (either **DBM_INSERT** to insert a new pair, or fail if the key already exists, or **DBM_REPLACE** to insert or replace existing data). The key and datum strings may contain an arbitrary series of characters.

DBM_FETCH returns the stored datum (if any) for the supplied key, or fails.

DBM_DELETE deletes the stored datum (if any) for the supplied key, or fails.

DBM_FIRSTKEY and subsequent calls to **DBM_NEXTKEY** allow you to traverse all stored keys. The keys will be returned in arbitrary order, and the routines will fail at the end of the traversal.

DBM_ERROR is a predicate which succeeds if the database handle is valid and an I/O error has occurred on the file. **DBM_CLEARERR** is a predicate which succeeds if if the database handle is valid, and has the side effect of clearing the I/O error flag.

FILES

NDBM, GDBM, and SDBM create two files: *filename*.dir, *filename*.pag. Berkeley DB creates a single *filename*.db file.

SEE ALSO

ndbm(3), **dbopen**(3), **gdbm**(3).

AUTHOR

Philip L. Budne

BUGS

None of the implementations allow concurrent read and write.

Some implementations (classic NDBM and SDBM) place limits on the total size of key plus datum (typically slightly less than 1KB).

NOTE: Some implementations (classic NDBM and SDBM) create sparse file which appear (to "ls -l") to be larger than they are (see "ls -s"). Copying such files may cause the non-allocated blocks to be "filled" with zeroed disk blocks, and then the files really will be large!

Only GDBM provides locking to eliminate the possibility of file corruption, or reading of incomplete data.

GDBM locking sometimes fails on NFS mounted partitions (particularly on Linux systems). GDBM's NDBM interface does not provide a way to disable locking.

DBM databases accessed concurrently by multiple processes are traditionally (re)created from text files and used for fast disk-based read-only table lookups. Programs which need to update the file generate a new temporary copy using a different name, and then rename the new file(s), so that the next reader gets the new copies (existing readers continue to see old data).