NAME

lmdb – LMDB Lightning Memory-Mapped Database binding for gawk

SYNOPSIS

```
@load "lmdb"

handle = mdb_env_create()
```

DESCRIPTION

The *lmdb* extension implements the mdb C API. Please refer to the <lmdb.h> header file for documentation, or view it on the web at **http://www.lmdb.tech/doc**. This document explains only the peculiarities of the *gawk* implementation. Aside from the **mdb_strerror()** function, every function in this library should set the **MDB_ERRNO** variable before returning. So you can always check whether **(MDB_ERRNO == MDB_SUCCESS)** to see if the last call succeeded. The library also sets the standard *gawk* **ERRNO** variable if an error occurs.

Almost all of the mdb_* functions in <lmdb.h> are implemented, as of version 0.9.18.

To achieve optimal performance, please use cursors whenever possible. Under the hood, the mdb library always uses cursors.

Global Variables

MDB_SUCCESS
: 0, as defined in <lmdb.h>.

MDB_ERRNO
: Return status from the last mdb function call.

MDB_KEY, **MDB_DATA**
: Subscripts for use with **mdb_cursor_get()**.

MDB
: An array defining all the MDB_* constants in <lmdb.h> with the "MDB_" prefix removed. For example, MDB["FIRST"] contains the value of MDB_FIRST. The various flags arguments can be constructed by using the *gawk* **or()** function to combine flag bits defined in this array.

Functions

string mdb_strerror(<int errno>)
: This function takes an integer argument and returns the result from calling the *mdb_strerror()* library function. The *gawk* library defines an additional extension-specific error code that is also supported by this function. If the argument is not an integer, it returns an empty string.

string mdb_env_create()
: Call *mdb_env_create()* and return a string handle to use for referencing this environment. On error, an empty string is returned, and **MDB_ERRNO** contains the return code from *mdb_env_create()*.

int mdb_env_open(<env handle>, <path>, <u_int flags>, <u_int mode>)
: Returns the status, which can also be found in **MDB_ERRNO**. You can use the *gawk* **or()** function to construct the flags. For example, the third argument might be **or(MDB["NOSUBDIR"], MDB["NOSYNC"], MDB["NOLOCK"])**.

int mdb_env_close(<env handle>)
: Call *mdb_env_close()* and return status, which can also be found in **MDB_ERRNO**. Although the C function has no return value, this function can return an error if the handle argument is invalid. If the result is **MDB_SUCCESS**, the handle is released and may no longer be used.

int mdb_env_sync(<env handle>, <int force>)
: Returns the status, which can also be found in **MDB_ERRNO**.

int mdb_env_copy(<env handle>, <path string>)
: Returns the status, which can also be found in **MDB_ERRNO**.

int mdb_env_copy2(<env handle>, <path string>, <u_int flags>)
Returns the status, which can also be found in **MDB_ERRNO**.

int mdb_env_get_flags(<env handle>)
On error, returns 0, else the flags value. Please compare **MDB_ERRNO** to **MDB_SUCCESS** to see whether the call succeeded.

int mdb_env_get_maxkeysize(<env handle>)
On error, returns 0, else the maxkeysize value. Please compare **MDB_ERRNO** to **MDB_SUCCESS** to see whether the call succeeded.

int mdb_env_get_maxreaders(<env handle>)
On error, returns 0, else the maxreaders value. Please compare **MDB_ERRNO** to **MDB_SUCCESS** to see whether the call succeeded.

string mdb_env_get_path(<env handle>)
On error, returns "", else the path value. Please compare **MDB_ERRNO** to **MDB_SUCCESS** to see whether the call succeeded.

int mdb_env_set_flags(<env handle>, <u_int flags>, <int onoff>)
Returns the status, which can also be found in **MDB_ERRNO**.

int mdb_env_set_mapsize(<env handle>, <u_int mapsize>)
Returns the status, which can also be found in **MDB_ERRNO**.

int mdb_env_set_maxdbs(<env handle>, <u_int dbs>)
Returns the status, which can also be found in **MDB_ERRNO**.

int mdb_env_set_maxreaders(<env handle>, <u_int readers>)
Returns the status, which can also be found in **MDB_ERRNO**.

string mdb_txn_begin(<env handle>, <parent txn handle or empty string>, <u_int flags>)
Call *mdb_txn_begin()* and return a string handle to use for referencing this transaction. On error, an empty string is returned, and **MDB_ERRNO** will contain an error code.

int mdb_txn_id(<txn handle>)
On error, returns 0, else the transaction id. Please compare **MDB_ERRNO** to **MDB_SUCCESS** to see whether the call succeeded.

int mdb_txn_commit(<txn handle>)
Returns the status, which can also be found in **MDB_ERRNO**. If the result is **MDB_SUCCESS**, the handle is released and may no longer be used.

int mdb_txn_abort(<txn handle>)
Returns the status, which can also be found in **MDB_ERRNO**. If the result is **MDB_SUCCESS**, the handle is released and may no longer be used.

int mdb_txn_reset(<txn handle>)
Returns the status, which can also be found in **MDB_ERRNO**.

int mdb_txn_renew(<txn handle>)
Returns the status, which can also be found in **MDB_ERRNO**.

string mdb_dbi_open(<txn handle>, <database name or empty string>, <u_int flags>)
Call *mdb_dbi_open()* and return a string handle to use for referencing this transaction. On error, an empty string is returned, and **MDB_ERRNO** will contain an error code.

int mdb_dbi_close(<env handle>, <dbi handle>)
Returns the status, which can also be found in **MDB_ERRNO**. If the result is **MDB_SUCCESS**, the handle is released and may no longer be used.

int mdb_dbi_flags(<txn handle>, <dbi handle>)
On error, returns 0, else the flags value. Please compare **MDB_ERRNO** to **MDB_SUCCESS** to see whether the call succeeded.

int mdb_drop(<txn handle>, <dbi handle>, <del: 0 or 1>)
Returns the status, which can also be found in **MDB_ERRNO**. If the call succeeds and <del> is 1, the dbi handle is released and may no longer be used.

string mdb_get(<txn handle>, <dbi handle>, <key>)
On error, returns "", else the data string returned by the function. You must check **MDB_ERRNO** to see whether the call succeeded, since an empty string could be a valid data value.

Please note that starting with version 2 of the API available in *gawk* 4.2, the result will be flagged as user input data eligible for the strnum attribute. So if the value appears to be numeric, *gawk* will treat it as a number in comparisons. This feature was not available prior to version 2 of the *gawk* API.

int mdb_put(<txn handle>, <dbi handle>, <key>, <data>, <u_int flags>)
Returns the status, which can also be found in **MDB_ERRNO**.

int mdb_del(<txn handle>, <dbi handle>, <key>[, <data>])
Returns the status, which can also be found in **MDB_ERRNO.** If the fourth argument is not present, a NULL pointer is passed as the fourth argument to the underlying function. Please be careful: passing an empty "" string as the fourth argument is not the same as omitting the fourth argument!

string mdb_cursor_open(<txn handle>, <dbi handle>)
Call *mdb_cursor_open()* and return a string handle to use for referencing this transaction. On error, an empty string is returned, and **MDB_ERRNO** will contain an error code.

int mdb_cursor_close(<cursor handle>)
Returns the status, which can also be found in **MDB_ERRNO**.

int mdb_cursor_renew(<cursor handle>)
Returns the status, which can also be found in **MDB_ERRNO**.

int mdb_cursor_get(<cursor handle>, <key/data array>, <u_int cursor op>)
Returns the status, which can also be found in **MDB_ERRNO**. The second argument is an array. If the **MDB_KEY** element is populated, its value is passed as the second argument to the underlying function. In its absence, an empty "" string is passed. Similarly, the contents of the **MDB_DATA** element are used to construct the third argument to the underlying function. On exit, if the function call succeeds, these same array elements will be populated with the data returned.

Please note that starting with version 2 of the API available in *gawk* 4.2, the returned array elements will be flagged as user input data eligible for the strnum attribute. So if the key or data appears to be numeric, *gawk* will treat it as a number in comparisons. This feature was not available prior to version 2 of the *gawk* API.

int mdb_cursor_put(<cursor handle>, <key>, <data>, <u_int flags>)
Returns the status, which can also be found in **MDB_ERRNO**.

int mdb_cursor_del(<cursor handle>, <u_int flags>)
Returns the status, which can also be found in **MDB_ERRNO**.

int mdb_cursor_count(<cursor handle>)
On error, returns 0, else the function result. Please compare **MDB_ERRNO** to **MDB_SUCCESS** to see whether the call succeeded.

int mdb_reader_check(<env handle>)
On error, returns 0, else the dead value from the function. Please compare **MDB_ERRNO** to **MDB_SUCCESS** to see whether the call succeeded.

int mdb_cmp(<txn handle>, <dbi handle>, <string a>, <string b>)
On error, returns 0, else the result of the underlying function. **MDB_ERRNO** to **MDB_SUCCESS** to see whether the call succeeded.

int mdb_dcmp(<txn handle>, <dbi handle>, <string a>, <string b>)
On error, returns 0, else the result of the underlying function. **MDB_ERRNO** to **MDB_SUCCESS** to see whether the call succeeded.

int mdb_env_stat(<env handle>, <array>)
Returns the status, which can also be found in **MDB_ERRNO**. If the call succeeds, the array is cleared and populated with the elements of the MDB_stat structure stripped of the "ms_" prefix.

int mdb_stat(<txn handle>, <dbi handle>, <array>)
Returns the status, which can also be found in **MDB_ERRNO**. If the call succeeds, the array is cleared and populated with the elements of the MDB_stat structure stripped of the "ms_" prefix.

int mdb_env_info(<env handle>, <array>)
Returns the status, which can also be found in **MDB_ERRNO**. If the call succeeds, the array is cleared and populated with the elements of the MDB_envinfo structure stripped of the "me_" prefix.

string mdb_txn_env(<txn handle>)
On error, an empty "" string is returned, else the env handle associated with this transaction.

string mdb_cursor_txn(<cursor handle>)
On error, an empty "" string is returned, else the transaction handle associated with this cursor.

string mdb_cursor_dbi(<cursor handle>)
On error, an empty "" string is returned, else the dbi handle associated with this cursor.

string mdb_version([<version array>])
Call *mdb_version()* and return the version string. If the optional array argument is supplied, it will be cleared, and the **major**, **minor**, and **patch** subscripts will be populated with the associated version numbers.

The following functions are not implemented: mdb_env_copyfd, mdb_env_copyfd2, mdb_env_get_fd, mdb_env_get_userctx, mdb_env_set_assert, mdb_env_set_userctx, mdb_reader_list, mdb_set_compare, mdb_set_dupsort, mdb_set_relctx, mdb_set_relfunc.

Of these, mdb_set_compare and mdb_set_dupsort seem most important. To implement these, we need a way for the extension to call into a *gawk* function. I do not think that is currently possible.

EXAMPLE

Please refer to **dict.awk** and **dict_cursor.awk** located in the **test** directory.

SEE ALSO

/usr/include/lmdb.h, **http://www.lmdb.tech/doc**

AUTHOR

Andrew Schorr

COPYING PERMISSIONS