## NAME

dbm – dbminit, fetch, store, delete, firstkey, nextkey

ndbm – dbm_open, dbm_close, dbm_fetch, dbm_store, dbm_delete, dbm_firstkey, dbm_nextkey, dbm_error, dbm_clearerr

## SYNOPSIS

```
#define DB_DBM_HSEARCH    1
#include <db.h>

typedef struct {
        char *dptr;
        int dsize;
} datum;
```

## DBM FUNCTIONS

```
int
dbminit(char *file);

datum
fetch(datum key);

int
store(datum key, datum content);

int
delete(datum key);

datum
firstkey(void);

datum
nextkey(datum key);
```

## NDBM FUNCTIONS

```
DBM *
dbm_open(char *file, int flags, int mode);

void
dbm_close(DBM *db);

datum
dbm_fetch(DBM *db, datum key);

int
dbm_store(DBM *db, datum key, datum content, int flags);

int
dbm_delete(DBM *db, datum key);

datum
dbm_firstkey(DBM *db);

datum
dbm_nextkey(DBM *db);

int
dbm_error(DBM *db);

int
dbm_clearerr(DBM *db);
```

**DESCRIPTION**

The *dbm* and *ndbm* interfaces to the DB library are intended to provide source code compatibility for historic applications. They are not recommended for any other purpose. The historic *dbm* and *ndbm* database format is **not** supported, and databases previously built using the real *dbm* or *ndbm* libraries cannot be read by the DB functions.

To compile *dbm* or *ndbm* applications, replace the application's *#include* of the dbm or ndbm include file (e.g., ‘‘#include <dbm.h>’’ or ‘‘#include <ndbm.h>’’) with the following two lines:

```
#define DB_DBM_HSEARCH    1
#include <db.h>
```

and recompile. If the application attempts to load against a dbm library (e.g., ‘‘-ldbm’’), remove the library from the load line.

*Key*s and *content*s are described by the *datum* typedef. A *datum* specifies a string of *dsize* bytes pointed to by *dptr*. Arbitrary binary data, as well as normal text strings, are allowed.

**DBM FUNCTIONS**

Before a database can be accessed, it must be opened by *dbminit*. This will open and/or create the database *file*.db. If created, the database file is created read/write by owner only (as described in *chmod*(2)) and modified by the process' umask value at the time of creation (see *umask*(2)). The group ownership of created files is based on the system and directory defaults, and is not further specified by DB.

Once open, the data stored under a key is accessed by *fetch* and data is placed under a key by *store*. A key (and its associated contents) is deleted by *delete*. A linear pass through all keys in a database may be made, in an (apparently) random order, by use of *firstkey* and *nextkey*. *Firstkey* will return the first key in the database. With any key *nextkey* will return the next key in the database. This code will traverse the data base:

```
for (key = firstkey();
        key.dptr != NULL; key = nextkey(key))
```

**NDBM FUNCTIONS**

Before a database can be accessed, it must be opened by *dbm_open*. This will open and/or create the database file *file*.db depending on the flags parameter (see *open*(2)). If created, the database file is created with mode *mode* (as described in *chmod*(2)) and modified by the process' umask value at the time of creation (see *umask*(2)). The group ownership of created files is based on the system and directory defaults, and is not further specified by DB.

Once open, the data stored under a key is accessed by *dbm_fetch* and data is placed under a key by *dbm_store*. The *flags* field can be either **DBM_INSERT** or **DBM_REPLACE. DBM_INSERT** will only insert new entries into the database and will not change an existing entry with the same key. **DBM_REPLACE** will replace an existing entry if it has the same key. A key (and its associated contents) is deleted by *dbm_delete*. A linear pass through all keys in a database may be made, in an (apparently) random order, by use of *dbm_firstkey* and *dbm_nextkey*. *Dbm_firstkey* will return the first key in the database. *Dbm_nextkey* will return the next key in the database. This code will traverse the data base:

```
for (key = dbm_firstkey(db);
        key.dptr != NULL; key = dbm_nextkey(db))
```

*Dbm_error* returns non-zero when an error has occurred reading or writing the database. *Dbm_clearerr* resets the error condition on the named database.

**COMPATIBILITY NOTES**

The historic *dbm* and *ndbm* libraries created two underlying database files, traditionally named *file*.dir and *file*.pag. The DB library creates a single database file named *file*.db. Applications that are aware of the underlying database file names may require additional source code modifications.

The historic *dbminit* interface required that the underlying ‘‘.dir’’ and ‘‘.pag’’ files already exist (empty databases were created by creating zero-length ‘‘.dir’’ and ‘‘.pag’’ files). Applications that expect to create databases using this method may require additional source code modifications.

The historic *dbm_dirfno* and *dbm_pagfno* macros are supported, but will return identical file descriptors as there is only a single underlying file used by the DB hashing access method. Applications using both file descriptors for locking may require additional source code modifications.

If an application using the *ndbm* interface exits without closing the database, it may lose updates because the DB library buffers all writes. Such applications will require additional source code modifications to work correctly with the DB library.

**DBM DIAGNOSTICS**

The *dbminit* function returns -1 on failure, setting *errno*, and 0 on success.

The *fetch* function sets the returned *datum*'s *dptr* field to NULL on failure, setting *errno*, and returns a non-NULL *dptr* on success.

The *store* function returns -1 on failure, setting *errno*, and 0 on success.

The *delete* function returns -1 on failure, setting *errno*, and 0 on success.

The *firstkey* function sets the returned *datum*'s *dptr* field to NULL on failure, setting *errno*, and returns a non-NULL *dptr* on success.

The *nextkey* function sets the returned *datum*'s *dptr* field to NULL on failure, setting *errno*, and returns a non-NULL *dptr* on success.

**NDBM DIAGNOSTICS**

The *dbm_open* function returns NULL on failure, setting *errno*, and 0 on success.

The *dbm_fetch* function sets the returned *datum*'s *dptr* field to NULL on failure, setting *errno*, and returns a non-NULL *dptr* on success.

The *dbm_store* function returns -1 on failure, setting *errno*, 0 on success, and 1 if DBM_INSERT was set and the specified key already existed in the database.

The *dbm_delete* function returns -1 on failure, setting *errno*, and 0 on success.

The *dbm_firstkey* function sets the returned *datum*'s *dptr* field to NULL on failure, setting *errno*, and returns a non-NULL *dptr* on success.

The *dbm_nextkey* function sets the returned *datum*'s *dptr* field to NULL on failure, setting *errno*, and returns a non-NULL *dptr* on success.

The *dbm_error* function returns -1 on failure, setting *errno*, and 0 on success.

The *dbm_clearerr* function returns -1 on failure, setting *errno*, and 0 on success.

**ERRORS**

The *dbminit* function may fail and return *errno* for any of the errors specified for the following DB and library functions: dbm_close(3), and dbm_open(3).

The *fetch* function may fail and return *errno* for any of the errors specified for the following DB and library functions: dbm_fetch(3).

The *store* function may fail and return *errno* for any of the errors specified for the following DB and library functions: dbm_store(3).

The *delete* function may fail and return *errno* for any of the errors specified for the following DB and library functions: dbm_delete(3).

The *firstkey* function may fail and return *errno* for any of the errors specified for the following DB and library functions: dbm_firstkey(3).

The *nextkey* function may fail and return *errno* for any of the errors specified for the following DB and library functions: dbm_nextkey(3).

The *dbm_open* function may fail and return *errno* for any of the errors specified for the following DB and library functions: db_open(3), and memset(3).

The *dbm_close* function may fail and return *errno* for any of the errors specified for the following DB and library functions: DB->close(3).

The *dbm_fetch* function may fail and return *errno* for any of the errors specified for the following DB and library functions: DB->get(3), and memset(3).

The *dbm_store* function may fail and return *errno* for any of the errors specified for the following DB and library functions: DB->put(3), and memset(3).

The *dbm_delete* function may fail and return *errno* for any of the errors specified for the following DB and library functions: memset(3).

The *dbm_firstkey* function may fail and return *errno* for any of the errors specified for the following DB and library functions: DB->cursor(3), and memset(3).

The *dbm_nextkey* function may fail and return *errno* for any of the errors specified for the following DB and library functions: DB->cursor(3), and memset(3).

**SEE ALSO**

The DB library is a family of groups of functions that provides a modular programming interface to transactions and record-oriented file access. The library includes support for transactions, locking, logging and file page caching, as well as various indexed access methods. Many of the functional groups (e.g., the file page caching functions) are useful independent of the other DB functions, although some functional groups are explicitly based on other functional groups (e.g., transactions and logging). For a general description of the DB package, see *db_intro*(3).

*db_archive*(1), *db_checkpoint*(1), *db_deadlock*(1), *db_dump*(1), *db_load*(1), *db_recover*(1), *db_stat*(1), *db_intro*(3), *db_appinit*(3), *db_cursor*(3), *db_dbm*(3), *db_internal*(3), *db_lock*(3), *db_log*(3), *db_mpool*(3), *db_open*(3), *db_thread*(3), *db_txn*(3)