NAME

dbm, dbminit, dbmclose, fetch, store, delete, firstkey, nextkey – data base subroutines

SYNOPSIS

/usr/ucb/cc [*flag* ...] *file* ... **–ldbm**

```
#include <dbm.h>

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

int dbminit(file)
char *file;

int dbmclose()

datum fetch( key)
datum key;

int store(key, dat)
datum key, dat;

int delete(key)
datum key;

datum firstkey()

datum nextkey(key)
datum key;
```

DESCRIPTION

The **dbm()** library has been superseded by **ndbm**(3).

These functions maintain key/content pairs in a data base. The functions will handle very large (a billion blocks) databases and will access a keyed item in one or two file system accesses.

key/dat and their content 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 ASCII strings, are allowed. The data base is stored in two files. One file is a directory containing a bit map and has **.dir** as its suffix. The second file contains all data and has **.pag** as its suffix.

Before a database can be accessed, it must be opened by **dbminit()**. At the time of this call, the files *file***.dir** and *file***.pag** must exist. An empty database is created by creating zero-length **.dir** and **.pag** files.

A database may be closed by calling **dbmclose()**. You must close a database before opening a new one.

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))

RETURN VALUES

All functions that return an **int** indicate errors with negative values. A zero return indicates no error. Routines that return a **datum** indicate errors with a NULL (0) *dptr*.

SEE ALSO

ar(1), **cat**(1), **cp**(1), **tar**(1), **ndbm**(3)

NOTES

Use of these interfaces should be restricted to only applications written on BSD platforms. Use of these interfaces with any of the system libraries or in multi-thread applications is unsupported.

The **.pag** file will contain holes so that its apparent size may be larger than its actual content. Older versions of the UNIX operating system may create real file blocks for these holes when touched. These files cannot be copied by normal means (**cp**(1), **cat**(1), **tar**(1), **ar**(1)) without filling in the holes.

dptr pointers returned by these subroutines point into static storage that is changed by subsequent calls.

The sum of the sizes of a key/content pair must not exceed the internal block size (currently 1024 bytes). Moreover all key/content pairs that hash together must fit on a single block. **store** will return an error in the event that a disk block fills with inseparable data.

delete( ) does not physically reclaim file space, although it does make it available for reuse.

The order of keys presented by **firstkey( )** and **nextkey( )** depends on a hashing function, not on anything interesting.

There are no interlocks and no reliable cache flushing; thus concurrent updating and reading is risky.

The database files (*file***.dir** and *file***.pag**) are binary and are architecture-specific (for example, they depend on the architecture's byte order.) These files are not guaranteed to be portable across architectures.