NAME

hash - hash database access method

SYNOPSIS

#include <sys/types.h>
#include <db.h>

DESCRIPTION

The routine **dbopen**() is the library interface to database files. One of the supported file formats is **hash** files. The general description of the database access methods is in dbopen(3), this manual page describes only the **hash** specific information.

The **hash** data structure is an extensible, dynamic hashing scheme.

The access method specific data structure provided to **dbopen**() is defined in the <*db.h*> include file as follows:

```
typedef struct {
        u_int bsize;
        u_int ffactor;
        u_int nelem;
        u_int cachesize;
        uint32_t (*hash)(const void *, size_t);
        int lorder;
} HASHINFO;
```

The elements of this structure are as follows:

bsize The *bsize* element defines the **hash** table bucket size, and is, by default, 4096 bytes. It may be preferable to increase the page size for disk-resident tables and tables with large data items.

ffactor
The *ffactor* element indicates a desired density within the **hash** table. It is an approximation of the number of keys allowed to accumulate in any one bucket, determining when the **hash** table grows or shrinks. The default value is 8.

nelem The *nelem* element is an estimate of the final size of the **hash** table. If not set or set too low, **hash** tables will expand gracefully as keys are entered, although a slight performance degradation may be noticed. The default value is 1.

cachesize
A suggested maximum size, in bytes, of the memory cache. This value is *only* advisory, and the access method will allocate more memory rather than fail.

hash The *hash* element is a user defined **hash** function. Since no **hash** function performs equally well on all possible data, the user may find that the built-in **hash** function does poorly on a particular data set. User specified **hash** functions must take two arguments (a pointer to a byte string and a length) and return a 32-bit quantity to be used as the **hash** value.

lorder The byte order for integers in the stored database metadata. The number should represent the order as an integer; for example, big endian order would be the number 4,321. If *lorder* is 0 (no order is specified) the current host order is used. If the file already exists, the specified value is ignored and the value specified when the tree was created is used.

If the file already exists (and the O_TRUNC flag is not specified), the values specified for the *bsize*, *ffactor*, *lorder* and *nelem* arguments are ignored and the values specified when the tree was created are used.

If a **hash** function is specified, **hash_open**() will attempt to determine if the **hash** function specified is the same as the one with which the database was created, and will fail if it is not.

Backward compatible interfaces to the older *dbm* and *ndbm* routines are provided, however these interfaces are not compatible with previous file formats.

ERRORS

The **hash** access method routines may fail and set *errno* for any of the errors specified for the library routine dbopen(3).

SEE ALSO

btree(3), dbopen(3), mpool(3), recno(3)

Per-Ake Larson, *Dynamic Hash Tables*, Communications of the ACM, April 1988.

Margo Seltzer, *A New Hash Package for UNIX*, USENIX Proceedings, Winter 1991.

BUGS

Only big and little endian byte order is supported.