GNU dbm

A Database Manager

by Philip A. Nelson, Jason Downs and Sergey Poznyakoff

Manual by Pierre Gaumond, Philip A. Nelson, Jason Downs
and Sergey Poznyakoff

Edition 1.10

for GNU `dbm`, Version 1.10

This is Edition 1.10 of the *GNU* `dbm` *Manual*, for `gdbm` Version 1.10.
Last updated 8 September 2012

Published by the Free Software Foundation
675 Massachusetts Avenue,
Cambridge, MA 02139 USA

Short Contents

Table of Contents

1 Copying Conditions.

This library is *free*; this means that everyone is free to use it and free to redistribute it on a free basis. GNU `dbm` (`gdbm`) is not in the public domain; it is copyrighted and there are restrictions on its distribution, but these restrictions are designed to permit everything that a good cooperating citizen would want to do. What is not allowed is to try to prevent others from further sharing any version of `gdbm` that they might get from you.

Specifically, we want to make sure that you have the right to give away copies `gdbm`, that you receive source code or else can get it if you want it, that you can change these functions or use pieces of them in new free programs, and that you know you can do these things.

To make sure that everyone has such rights, we have to forbid you to deprive anyone else of these rights. For example, if you distribute copies `gdbm`, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must tell them their rights.

Also, for our own protection, we must make certain that everyone finds out that there is no warranty for anything in the `gdbm` distribution. If these functions are modified by someone else and passed on, we want their recipients to know that what they have is not what we distributed, so that any problems introduced by others will not reflect on our reputation.

`Gdbm` is currently distributed under the terms of the GNU General Public License, Version 3. (*NOT* under the GNU General Library Public License.) A copy the GNU General Public License is included with the distribution of `gdbm`.

2 Introduction to GNU `dbm`.

GNU `dbm` (`gdbm`) is a library of database functions that use extensible hashing and works similar to the standard UNIX `dbm` functions. These routines are provided to a programmer needing to create and manipulate a hashed database. (`gdbm` is *NOT* a complete database package for an end user.)

The basic use of `gdbm` is to store key/data pairs in a data file. Each key must be unique and each key is paired with only one data item. The keys can not be directly accessed in sorted order. The basic unit of data in `gdbm` is the structure:

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

This structure allows for arbitrary sized keys and data items.

The key/data pairs are stored in a `gdbm` disk file, called a `gdbm` database. An application must open a `gdbm` database to be able manipulate the keys and data contained in the database. `gdbm` allows an application to have multiple databases open at the same time. When an application opens a `gdbm` database, it is designated as a `reader` or a `writer`. A `gdbm` database can be opened by at most one writer at a time. However, many readers may open the database simultaneously. Readers and writers can not open the `gdbm` database at the same time.

3 List of functions.

The following is a quick list of the functions contained in the `gdbm` library. The include file `gdbm.h`, that can be included by the user, contains a definition of these functions.

```
#include <gdbm.h>

GDBM_FILE gdbm_open(name, block_size, flags, mode, fatal_func);
void gdbm_close(dbf);
int gdbm_store(dbf, key, content, flag);
datum gdbm_fetch(dbf, key);
int gdbm_delete(dbf, key);
datum gdbm_firstkey(dbf);
datum gdbm_nextkey(dbf, key);
int gdbm_reorganize(dbf);
void gdbm_sync(dbf);
int gdbm_exists(dbf, key);
char *gdbm_strerror(errno);
int gdbm_setopt(dbf, option, value, size);
int gdbm_fdesc(dbf);
```

The `gdbm.h` include file is often in the `/usr/local/include` directory. (The actual location of `gdbm.h` depends on your local installation of `gdbm`.)

4 Opening the database.

`GDBM_FILE` **gdbm_open** (*const char* *`name`, *int* `block_size`, *int* `flags`, *int* `mode`, *void* (**fatal_func*)(*const char* *)) [gdbm interface]

Initializes `gdbm` system. If the file has a size of zero bytes, a file initialization procedure is performed, setting up the initial structure in the file.

The arguments are:

name	The name of the file (the complete name, `gdbm` does not append any characters to this name).
block_size	It is used during initialization to determine the size of various constructs. It is the size of a single transfer from disk to memory. This parameter is ignored if the file has been previously initialized. The minimum size is 512. If the value is less than 512, the file system block size is used, otherwise the value of *block_size* is used.
flags	If `flags` is set to '`GDBM_READER`', the user wants to just read the database and any call to `gdbm_store` or `gdbm_delete` will fail. Many readers can access the database at the same time. If `flags` is set to '`GDBM_WRITER`', the user wants both read and write access to the database and requires exclusive access. If `flags` is set to '`GDBM_WRCREAT`', the user wants both read and write access to the database and wants it created if it does not already exist. If `flags` is set to '`GDBM_NEWDB`', the user want a new database created, regardless of whether one existed, and wants read and write access to the new database. The following may also be logically or'd into the database flags: '`GDBM_SYNC`', which causes all database operations to be synchronized to the disk, '`GDBM_NOLOCK`', which prevents the library from performing any locking on the database file, and '`GDBM_NOMMAP`', which disables the memory mapping mechanism. The option '`GDBM_FAST`' is now obsolete, since `gdbm` defaults to no-sync mode. If the host '`open`' call (see Section "open" in *open(2) man page*) supports the '`O_CLOEXEC`' flag, the '`GDBM_CLOEXEC`' can be or'd into the flags, to enable the close-on-exec flag for the database file descriptor.
mode	File mode (see Section "change permissions of a file" in *chmod(2) man page*, and see Section "open a file" in *open(2) man page*), which is used if the file is created).
fatal_func	A function for `gdbm` to call if it detects a fatal error. The only parameter of this function is a string. If the value of '`NULL`' is provided, `gdbm` will use a default function.

The return value, is the pointer needed by all other functions to access that `gdbm` file. If the return is the '`NULL`' pointer, `gdbm_open` was not successful. The errors can be found in `gdbm_errno` variable (see Chapter 18 [Variables], page 25). Available error codes are discussed in Chapter 19 [Error codes], page 27.

In all of the following calls, the parameter *dbf* refers to the pointer returned from `gdbm_open`.

5 Closing the database.

It is important that every file opened is also closed. This is needed to update the reader/writer count on the file:

`void` `gdbm_close` (*GDBM_FILE* ***`dbf`***) [gdbm interface]

This function closes the `gdbm` file and frees all memory associated with it. The parameter is:

dbf	The pointer returned by `gdbm_open`.

6 Inserting and replacing records in the database.

`int gdbm_store` (*GDBM_FILE* `dbf`, *datum* `key`, *datum* `content`, *int* `flag`) [gdbm interface]

The function `gdbm_store` inserts or replaces records in the database.

The parameters are:

dbf	The pointer returned by `gdbm_open`.
key	The search key.
content	The data to be associated with the key.
flag	Defines the action to take when the key is already in the database. The value '`GDBM_REPLACE`' (defined in `gdbm.h`) asks that the old data be replaced by the new *content*. The value '`GDBM_INSERT`' asks that an error be returned and no action taken if the *key* already exists.

This function can return the following values:

-1	The item was not stored in the database because the caller was not an official writer or either *key* or *content* have a '`NULL`' '`dptr`' field. Both *key* and *content* must have the '`dptr`' field be a non-'`NULL`' value. Since a '`NULL`' '`dptr`' field is used by other functions to indicate an error, it cannot be valid data.
+1	The item was not stored because the argument *flag* was '`GDBM_INSERT`' and the *key* was already in the database.
0	No error. The value of *content* is keyed by *key*. The file on disk is updated to reflect the structure of the new database before returning from this function.

If you store data for a *key* that is already in the data base, `gdbm` replaces the old data with the new data if called with '`GDBM_REPLACE`'. You do not get two data items for the same `key` and you do not get an error from `gdbm_store`.

The size in `gdbm` is not restricted like `dbm` or `ndbm`. Your data can be as large as you want.

7 Searching for records in the database.

`datum gdbm_fetch` (*GDBM_FILE* ***dbf***, *datum* ***key***) [gdbm interface]

Looks up a given *key* and returns the information associated with it. The ‘`dptr`’ field in the structure that is returned points to a memory block allocated by `malloc`. It is the caller’s responsibility to free it when no longer needed.

If the ‘`dptr`’ is ‘`NULL`’, no data was found.

The parameters are:

dbf	The pointer returned by `gdbm_open`.
key	The search key.

An example of using this function:

```
content = gdbm_fetch (dbf, key);
if (content.dptr == NULL)
  {
    fprintf(stderr, "key not found\n");
  }
else
  {
    /* do something with content.dptr */
  }
```

You may also search for a particular key without retrieving it:

`int gdbm_exists` (*GDBM_FILE* ***dbf***, *datum* ***key***) [gdbm interface]

Returns ‘`true`’ (‘`1`’) if the *key* exists in *dbf* and ‘`false`’ (‘`0`’) otherwise.

The parameters are:

dbf	The pointer returned by `gdbm_open`.
key	The search key.

8 Removing records from the database.

To remove some data from the database, use the `gdbm_delete` function.

`int gdbm_delete` (*GDBM_FILE* `dbf`, *datum* `key`) [gdbm interface]

Deletes the data associated with the given *key*, if it exists in the database *dbf*. The file on disk is updated to reflect the structure of the new database before returning from this function.

The parameters are:

dbf The pointer returned by `gdbm_open`.

datum key
The search key.

The function returns '`-1`' if the item is not present or the requester is a reader. The return of '`0`' marks a successful delete.

9 Sequential access to records.

The next two functions allow for accessing all items in the database. This access is not `key` sequential, but it is guaranteed to visit every `key` in the database once. The order has to do with the hash values. `gdbm_firstkey` starts the visit of all keys in the database. `gdbm_nextkey` finds and reads the next entry in the hash structure for `dbf`.

`datum gdbm_firstkey` (*GDBM_FILE* ***dbf***) [gdbm interface]

> Initiate sequential access to the database *dbf*. The returned value is the first key accessed in the database. If the '`dptr`' field in the returned datum is '`NULL`', the database contains no data.
>
> Otherwise, '`dptr`' points to a memory block obtained from `malloc`, which holds the key value. The caller is responsible for freeing this memory block when no longer needed.

`datum gdbm_nextkey` (*GDBM_FILE* ***dbf***, *datum* ***prev***) [gdbm interface]

> This function continues the iteration over the keys in *dbf*, initiated by `gdbm_firstkey`. The parameter *prev* holds the value returned from a previous call to `gdbm_nextkey` or `gdbm_firstkey`.
>
> The function returns next key from the database. If the '`dptr`' field in the returned datum is '`NULL`', all keys in the database has been visited.
>
> Otherwise, '`dptr`' points to a memory block obtained from `malloc`, which holds the key value. The caller is responsible for freeing this memory block when no longer needed.

These functions were intended to visit the database in read-only algorithms, for instance, to validate the database or similar operations. The usual algorithm for sequential access is:

```
key = gdbm_firstkey (dbf);
while (key.dptr)
  {
     datum nextkey;

     /* do something with the key */
     ...

     /* Obtain the next key */
     nextkey = gdbm_nextkey (dbf, key);
     /* Reclaim the memory used by the key */
     free (key.dptr);
     /* Use nextkey in the next iteration. */
     key = nextkey;
  }
```

Care should be taken when the `gdbm_delete` function is used in such a loop. File visiting is based on a *hash table*. The `gdbm_delete` function re-arranges the hash table to make sure that any collisions in the table do not leave some item *un-findable*. The original key order is *not* guaranteed to remain unchanged in all instances. So it is possible that some key will not be visited if a loop like the following is executed:

```
key = gdbm_firstkey (dbf);
while (key.dptr)
  {
     datum nextkey;
     if (some condition)
       {
          gdbm_delete (dbf, key);
       }
      nextkey = gdbm_nextkey (dbf, key);
      free (key.dptr);
      key = nextkey;
   }
```

10 Database reorganization.

The following function should be used very seldom.

`int gdbm_reorganize` (*GDBM_FILE* `dbf`) [gdbm interface]

Reorganizes the database.

The parameter is:

dbf	The pointer returned by `gdbm_open`.

If you have had a lot of deletions and would like to shrink the space used by the `gdbm` file, this function will reorganize the database. This results, in particular, in shortening the length of a `gdbm` file by removing the space occupied by deleted records.

This reorganization requires creating a new file and inserting all the elements in the old file *dbf* into the new file. The new file is then renamed to the same name as the old file and *dbf* is updated to contain all the correct information about the new file. If an error is detected, the return value is negative. The value zero is returned after a successful reorganization.

11 Database Synchronization

Unless your database was opened with the '`GDBM_SYNC`' flag, `gdbm` does not wait for writes to be flushed to the disk before continuing. This allows for faster writing of databases at the risk of having a corrupted database if the application terminates in an abnormal fashion. The following function allows the programmer to make sure the disk version of the database has been completely updated with all changes to the current time.

`void gdbm_sync` (*GDBM_FILE* ***dbf***) [gdbm interface]

Synchronizes the changes in *dbf* with its disk file. The parameter is a pointer returned by `gdbm_open`.

This function would usually be called after a complete set of changes have been made to the database and before some long waiting time. The `gdbm_close` function automatically calls the equivalent of `gdbm_sync` so no call is needed if the database is to be closed immediately after the set of changes have been made.

12 Export and Import

`Gdbm` databases can be converted into a portable *flat format*. This format can be used, for example, to migrate between the different versions of `gdbm` databases. Generally speaking, flat files are safe to send over the network, and can be used to recreate the database on another machine. The recreated database is guaranteed to be a byte-to-byte equivalent of the database from which the flat file was created. This does not necessarily mean, however, that this file can be used in the same way as the original one. For example, if the original database contained non-ASCII data (e.g. C structures, integers etc.), the recreated database can be of any use only if the target machine has the same integer size and byte ordering as the source one and if its C compiler uses the same packing conventions as the one which generated C which populated the original database. In general, such binary databases are not portable between machines, unless you follow some stringent rules on what data is written to them and how it is interpreted.

`int gdbm_export` (*GDBM_FILE* **`dbf`**, *const char* *__`exportfile`__, *int* **`flag`**, *int* **`mode`**) [gdbm interface]

Create a flat file from the `gdbm` database. The parameters are:

dbf	A pointer to the source database, returned by a call to `gdbm_open`. The database must be open in '`GDBM_WRITER`' mode.
exportfile	The name of the output file.
flag	How to create the output file. If *flag* is '`GDBM_WRCREAT`', the file will be created if it does not exist already. Otherwise, if it is '`GDBM_NEWDB`', it will be created if it does not exist, and truncated otherwise.
mode	The permissions to use when creating the output file. See Section "open a file" in *open(2) man page*, for a detailed discussion.

`int gdbm_import` (*GDBM_FILE* **`dbf`**, *const char* *__`importfile`__, *int* **`flag`**) [gdbm interface]

Populates the database from an existing flat file.

dbf	A pointer to the source database, returned by a call to `gdbm_open`. The database must be open in '`GDBM_WRITER`' mode.
importfile	The name of the input flat file. The file must exist.
flag	The *flag* argument to be passed to `gdbm_store` function when adding new records. See Chapter 6 [Store], page 6, for a description of its effect.

See also Chapter 17 [gdbmexport], page 24, [testgdbm export], page 20, and [testgdbm import], page 21.

13 Error strings.

To convert a `gdbm` error code into English text, use this routine:

`const char * gdbm_strerror` (*gdbm_error* ***errno***) [gdbm interface]

Converts *errno* (which is an integer value) into a human-readable descriptive text. Returns a pointer to a static string. The caller must not alter or free the returned pointer.

The *errno* argument is usually the value of the global variable `gdbm_errno`. See Chapter 18 [Variables], page 25.

14 Setting options

`Gdbm` supports the ability to set certain options on an already open database.

`int` **`gdbm_setopt`** (*GDBM_FILE* `dbf`, *int* `option`, *void* `*value`, *int* `size`) [gdbm interface]

Sets an option on the database or returns the value of an option.

The parameters are:

dbf	The pointer returned by `gdbm_open`.
option	The option to be set or retreived.
value	A pointer to the value to which *option* will be set or where to place the option value (depending on the option).
size	The length of the data pointed to by *value*.

The valid options are:

GDBM_SETCACHESIZE
GDBM_CACHESIZE

Set the size of the internal bucket cache. This option may only be set once on each GDBM_FILE descriptor, and is set automatically to 100 upon the first access to the database. The *value* should point to a `size_t` holding the desired cache size.

The '`GDBM_CACHESIZE`' option is provided for compatibility with earlier versions.

GDBM_GETCACHESIZE

Return the size of the internal bucket cache. The *value* should point to a `size_t` variable, where the size will be stored.

GDBM_GETFLAGS

Return the flags describing the state of the database. The *value* should point to a `int` variable where to store the flags. The return is the same as the flags used when opening the database (see Chapter 4 [Open], page 4), except that it reflects the current state (which may have been altered by another calls to `gdbm_setopt`.

GDBM_FASTMODE

Enable or disable the *fast writes mode*, i.e. writes without subsequent synchronization. The *value* should point to an integer: '`TRUE`' to enable fast mode, and '`FALSE`' to disable it.

This option is retained for compatibility with previous versions of `gdbm`. Its effect is the reverse of `GDBM_SETSYNCMODE` (see below).

GDBM_SETSYNCMODE
GDBM_SYNCMODE

Turn on or off file system synchronization operations. This setting defaults to off. The *value* should point to an integer: '`TRUE`' to turn synchronization on, and '`FALSE`' to turn it off.

Note, that this option is a reverse of `GDBM_FASTMODE`, i.e. calling `GDBM_SETSYNCMODE` with '`TRUE`' has the same effect as calling `GDBM_FASTMODE` with '`FALSE`'.

The '`GDBM_SYNCMODE`' option is provided for compatibility with earlier versions.

GDBM_GETSYNCMODE
: Return the current synchronization status. The *value* should point to an `int` where the status will be stored.

GDBM_SETCENTFREE
GDBM_CENTFREE
: *NOTICE: This feature is still under study.*

Set central free block pool to either on or off. The default is off, which is how previous versions of `gdbm` handled free blocks. If set, this option causes all subsequent free blocks to be placed in the *global* pool, allowing (in theory) more file space to be reused more quickly. The *value* should point to an integer: '`TRUE`' to turn central block pool on, and '`FALSE`' to turn it off.

The '`GDBM_CENTFREE`' option is provided for compatibility with earlier versions.

GDBM_SETCOALESCEBLKS
GDBM_COALESCEBLKS
: *NOTICE: This feature is still under study.*

Set free block merging to either on or off. The default is off, which is how previous versions of `gdbm` handled free blocks. If set, this option causes adjacent free blocks to be merged. This can become a CPU expensive process with time, though, especially if used in conjunction with GDBM_CENTFREE. The *value* should point to an integer: '`TRUE`' to turn free block merging on, and '`FALSE`' to turn it off.

GDBM_GETCOALESCEBLKS
: Return the current status of free block merging. The *value* should point to an `int` where the status will be stored.

GDBM_SETMAXMAPSIZE
: Sets maximum size of a memory mapped region. The *value* should point to a value of type `size_t`, `unsigned long` or `unsigned`. The actual value is rounded to the nearest page boundary (the page size is obtained from `sysconf(_SC_PAGESIZE)`).

GDBM_GETMAXMAPSIZE
: Return the maximum size of a memory mapped region. The *value* should point to a value of type `size_t` where to return the data.

GDBM_SETMMAP
: Enable or disable memory mapping mode. The *value* should point to an integer: '`TRUE`' to enable memory mapping or '`FALSE`' to disable it.

GDBM_GETMMAP
: Check whether memory mapping is enabled. The *value* should point to an integer where to return the status.

GDBM_GETDBNAME

Return the name of the database disk file. The *value* should point to a variable of type `char**`. A pointer to the newly allocated copy of the file name will be placed there. The caller is responsible for freeing this memory when no longer needed. For example:

```
char *name;

if (gdbm_setopt (dbf, GDBM_GETDBNAME, &name, sizeof (name)))
  {
     fprintf (stderr, "gdbm_setopt failed: %s\n",
      gdbm_strerror (gdbm_errno));
  }
else
  {
    printf ("database name: %s\n", name);
    free (name);
  }
```

The return value will be '`-1`' upon failure, or '`0`' upon success. The global variable `gdbm_errno` will be set upon failure.

For instance, to set a database to use a cache of 10, after opening it with `gdbm_open`, but prior to accessing it in any way, the following code could be used:

```
int value = 10;
ret = gdbm_setopt (dbf, GDBM_CACHESIZE, &value, sizeof (int));
```

15 File Locking.

With locking disabled (if `gdbm_open` was called with ‘`GDBM_NOLOCK`’), the user may want to perform their own file locking on the database file in order to prevent multiple writers operating on the same file simultaneously.

In order to support this, the `gdbm_fdesc` routine is provided.

`int` **`gdbm_fdesc`** (*GDBM_FILE* `dbf`) [gdbm interface]

Returns the file descriptor of the database *dbf*. This value can be used as an argument to `flock`, `lockf` or similar calls.

16 Test and modify a GDBM database.

The `testgdbm` utility allows you to view and modify an existing GDBM database or to create a new one.

When invoked without options, it tries to open a database file called `junk.gdbm`, located in the current working directory. You can change this default using the `-g` command line option. This option takes a single argument, specifying the file name to open, e.g.:

```
$ testgdbm -g file.db
```

The database will be opened in read-write mode, unless the `-r` option is specified, in which case it will be opened only for reading.

If the database does not exist, `testgdbm` will create it. There is a special option `-n`, which instructs the utility to create a new database. If it is used and if the database already exists, it will be deleted, so use it sparingly.

16.1 testgdbm invocation

The following table summarizes all `testgdbm` command line options:

`-b` *`size`*	Set block size.
`-c` *`size`*	Set cache size.
`-g` *`file`*	Operate on *file* instead of the default `junk.gdbm`.
`-h`	Print a concise help summary.
`-n`	Create the database.
`-r`	Open the database in read-only mode.
`-s`	Synchronize to the disk after each write.
`-v`	Print program version and licensing information and exit.

16.2 testgdbm interactive mode

After successful startup, `testgdbm` starts a loop, in which it reads commands from the user, executes them and prints the results on the standard output. If the standard input is attached to a console, `testgdbm` runs in interactive mode, which is indicated by its *prompt*:

```
testgdbm> _
```

The utility finishes when it reads the '`quit`' command (see below) or detects end-of-file on its standard input, whichever occurs first.

A `testgdbm` command consists of a *command verb*, optionally followed by one or two *arguments*, separated by any amount of white space. A command verb can be entered either in full or in an abbreviated form, as long as that abbreviation does not match any other verb. For example, '`co`' can be used instead of '`count`' and '`ca`' instead of '`cache`'. Furthermore, many command verbs also have single-letter forms, called *command letters*.

An argument is any sequence of non-whitespace characters. Notice, that currently there is no way to enter arguments containing white space. This limitation will be removed in future releases.

Each command takes at most two *formal parameters*, which can be optional or mandatory. If the number of actual arguments is less than the number of mandatory parameters, `testgdbm` will prompt you to supply missing arguments. For example, the '`store`' command takes two mandatory parameters, so if you invoked it with no arguments, you would be prompted twice to supply the necessary data, as shown in example below:

```
testgdbm> store
key> three
data> 3
```

However, such prompting is possible only in interactive mode. In non-interactive mode (e.g. when running a script), all arguments must be supplied with each command, otherwise `testgdbm` will report an error and exit immediately.

Some commands produce excessive amounts of output. To help you follow it, `testgdbm` uses a pager utility to display such output. The name of the pager utility is taken from the environment variable `PAGER`. The pager is invoked only in interactive mode and only if the estimated number of output lines is greater then the number of lines on your screen.

Many of the `testgdbm` commands operate on database key and data values. The utility assumes that both keys and data are ASCII strings, either nul-terminated or not. By default, it is assumed that strings are nul-terminated. You can change this by using `z` (`key-zero`, for keys) and `Z` (`data-zero`, for data) commands.

The following table summarizes all available commands:

`count` [command verb]
`co` [command abbrev]
`c` [command letter]

Print the number of entries in the database.

`delete` *`key`* [command verb]
`de` *`key`* [command abbrev]
`d` *`key`* [command letter]

Delete entry with a given *key*

`export` *`file-name`* [*truncate*] [command verb]
`e` *`file-name`* [*truncate*] [command abbrev]

Export the database to the flat file *file-name*. See Chapter 12 [Flat files], page 13, for a description of the flat file format and its purposes. This command will not overwrite an existing file, unless the word '`truncate`' is given as its second argument.

See also Chapter 17 [gdbmexport], page 24.

`fetch` *`key`* [command verb]
`fe` *`key`* [command abbrev]
`f` *`key`* [command letter]

Fetch and display a record with the given *key*.

`import` *`file-name`* [*replace*] [command verb]
`i` *`file-name`* [*replace*] [command abbrev]

Import data from a flat dump file *file-name* (see Chapter 12 [Flat files], page 13). If the word '`replace`' is given as the second argument, any records with the same keys as the already existing ones will replace them.

`list` [command verb]
`l` [command abbrev]

List the contents of the database (see [pager], page 20).

`next` [*key*] [command verb]
`n` [*key*] [command abbrev]

Sequential access: fetch and display a next record. If *key* is given, a record following one with this key will be fetched. Otherwise, the key supplied by the latest `1`, `2` or *n* command will be used.

See also `first`, below.

See Chapter 9 [Sequential], page 9, for more information on sequential access.

`quit` [command verb]
`q` [command abbrev]

Close the database and quit the utility.

`store` *key data* [command verb]
`sto` *key data* [command abbrev]
`s` *key data* [command letter]

Store the *data* with *key* in the database. If *key* already exists, its data will be replaced.

`first` [command verb]
`fi` [command abbrev]
`1` [command letter]

Fetch and display the first record in the database. Subsequent records can be fetched using `next` command (see above). See Chapter 9 [Sequential], page 9, for more information on sequential access.

`read` *file* [*replace*] [command verb]
`rea` *file* [*replace*] [command abbrev]
`<` *file* [*replace*] [command letter]

Read entries from *file* and store them in the database. If the word '`replace`' is given as the second argument, any existing records with matching keys will be replaced.

`reorganize` [command verb]
`reo` [command abbrev]
`r` [command letter]

Reorganize the database (see Chapter 10 [Reorganization], page 11).

`key-zero` [command verb]
`k` [command abbrev]
`z` [command letter]

Toggle key nul-termination. Use `status` to inspect the current state. See [nul-termination], page 20.

`avail` [command verb]
`a` [command abbrev]
`A` [command letter]

Print the *avail list*.

`bucket` [command verb]
`b` [command abbrev]
`B` [command letter]

Print the bucket number *num*.

`current` [command verb]
`cu` [command abbrev]
`C` [command letter]

Print the current bucket.

`dir` [command verb]
`di` [command abbrev]
`D` [command letter]

Print hash directory.

`header` [command verb]
`hea` [command abbrev]
`F` [command letter]

Print file header.

`hash` *key* [command verb]
`ha` *key* [command abbrev]
`H` *key* [command letter]

Compute and display the hash value for the given *key*.

`cache` [command verb]
`ca` [command abbrev]
`K` [command letter]

Print the bucket cache.

`status` [command verb]
`sta` [command abbrev]
`S` [command letter]

Print current program status. The following example shows the information displayed:

```
Database file: junk.gdbm
Zero terminated keys: yes
Zero terminated data: yes
```

`version` [command verb]
`v` [command abbrev]

Print the version of `gdbm`.

`data-zero` [command verb]
`da` [command abbrev]
`Z` [command letter]

Toggle data nul-termination. Use `status` to examine the current status.

See [nul-termination], page 20.

help [command verb]
hel [command abbrev]
? [command letter]

Print a concise command summary, showing each command letter and verb with its parameters and a short description of what it does. Optional arguments are enclosed in square brackets.

17 Export a database into a portable format.

The `gdbmexport` utility converts the database into a portable *flat format*. Files in this format can be used to populate databases using the `gdbm_import` function (see Chapter 12 [Flat files], page 13) or the `i` command of `testgdbm` utility (see [testgdbm import], page 21). In many cases files in this format are suitable for sending over the network to populate the database on another machine. The only exception to this are databases whose records contain non-ASCII data (e.g. C structures, integers etc.). For such databases you will be better off by writing a specialized utility to convert them to an architecture-independent format.

If `gdbmexport` is linked with `libgdbm` version 1.8.3, it can be used to convert databases from old to new format.

The utility takes two mandatory arguments: the name of the database file to convert and the output file name, e.g.:

```
$ gdbmexport junk.gdbm junk.flat
```

In addition two options are understood:

`-h`	Display short usage summary and exit.
`-v`	Display program version and licensing information, and exit.

18 Useful global variables.

The following global variables and constants are available:

`gdbm_error gdbm_errno` [Variable]

This variable contains error code from the last failed `gdbm` call. See Chapter 19 [Error codes], page 27, for a list of available error codes and their descriptions.

Use `gdbm_strerror` (see Chapter 13 [Errors], page 14) to convert it to a descriptive text.

`const char * gdbm_errlist[]` [Variable]

This variable is an array of error descriptions, which is used by `gdbm_strerror` to convert error codes to human-readable text (see Chapter 13 [Errors], page 14). You can access it directly, if you wish so. It contains `_GDBM_MAX_ERRNO + 1` elements and can be directly indexed by the error code to obtain a corresponding descriptive text.

`_GDBM_MIN_ERRNO` [Constant]

The minimum error code used by `gdbm`.

`_GDBM_MAX_ERRNO` [Constant]

The maximum error code used by `gdbm`.

`const char * gdbm_version` [Variable]

A string containing the version information.

`int const gdbm_version_number[3]` [Variable]

This variable contains the `gdbm` version numbers:

Index	**Meaning**
0	Major number
1	Minor number
2	Patchlevel number

Additionally, the following constants are defined in the `gdbm.h` file:

GDBM_VERSION_MAJOR
  Major number.

GDBM_VERSION_MINOR
  Minor number.

GDBM_VERSION_PATCH
  Patchlevel number.

These can be used to verify whether the header file matches the library.

To compare two split-out version numbers, use the following function:

`int gdbm_version_cmp` (*int const* `a`[*3*], *int const* `b`[*3*]) [gdbm interface]

Compare two version numbers. Return ‘`-1`’ if *a* is less than *b*, ‘`1`’ if *a* is greater than *b* and ‘`0`’ if they are equal.

Comparison is done from left to right, so that:

```
a = { 1, 8, 3 };
b = { 1, 8, 3 };
gdbm_version_cmp (a, b) ⇒ 0

a = { 1, 8, 3 };
b = { 1, 8, 2 };
gdbm_version_cmp (a, b) ⇒ 1

a = { 1, 8, 3 };
b = { 1, 9. 0 };
gdbm_version_cmp (a, b) ⇒ -1
```

19 Error codes

This chapter summarizes the error codes which can be set by the functions in `gdbm` library.

GDBM_NO_ERROR
: No error occurred.

GDBM_MALLOC_ERROR
: Memory allocation failed. Not enough memory.

GDBM_BLOCK_SIZE_ERROR
: This error is set by the `gdbm_open` function (see Chapter 4 [Open], page 4), if the value of its *block_size* argument is incorrect.

GDBM_FILE_OPEN_ERROR
: The library was not able to open a disk file. This can be set by `gdbm_open` (see Chapter 4 [Open], page 4), `gdbm_export` and `gdbm_import` functions (see Chapter 12 [Flat files], page 13).

Inspect the value of the system `errno` variable to get more detailed diagnostics.

GDBM_FILE_WRITE_ERROR
: Writing to a disk file failed. This can be set by `gdbm_open` (see Chapter 4 [Open], page 4), `gdbm_export` and `gdbm_import` functions.

Inspect the value of the system `errno` variable to get more detailed diagnostics.

GDBM_FILE_SEEK_ERROR
: Positioning in a disk file failed. This can be set by `gdbm_open` (see Chapter 4 [Open], page 4) function.

Inspect the value of the system `errno` variable to get a more detailed diagnostics.

GDBM_FILE_READ_ERROR
: Reading from a disk file failed. This can be set by `gdbm_open` (see Chapter 4 [Open], page 4), `gdbm_export` and `gdbm_import` functions.

Inspect the value of the system `errno` variable to get a more detailed diagnostics.

GDBM_BAD_MAGIC_NUMBER
: The file given as argument to `gdbm_open` function is not a valid `gdbm` file: it has a wrong magic number.

GDBM_EMPTY_DATABASE
: The file given as argument to `gdbm_open` function is not a valid `gdbm` file: it has zero length.

GDBM_CANT_BE_READER
: This error code is set by the `gdbm_open` function if it is not able to lock file when called in '`GDBM_READER`' mode (see Chapter 4 [Open], page 4).

GDBM_CANT_BE_WRITER
: This error code is set by the `gdbm_open` function if it is not able to lock file when called in writer mode (see Chapter 4 [Open], page 4).

GDBM_READER_CANT_DELETE

Set by the `gdbm_delete` (see Chapter 8 [Delete], page 8) if it attempted to operate on a database that is open in read-only mode (see Chapter 4 [Open], page 4).

GDBM_READER_CANT_STORE

Set by the `gdbm_store` (see Chapter 6 [Store], page 6) if it attempted to operate on a database that is open in read-only mode (see Chapter 4 [Open], page 4).

GDBM_READER_CANT_REORGANIZE

Set by the `gdbm_reorganize` (see Chapter 10 [Reorganization], page 11) if it attempted to operate on a database that is open in read-only mode (see Chapter 4 [Open], page 4).

GDBM_UNKNOWN_UPDATE

Currently unused. Reserved for future uses.

GDBM_ITEM_NOT_FOUND

Requested item was not found. This error is set by `gdbm_delete` (see Chapter 8 [Delete], page 8) and `gdbm_fetch` (see Chapter 7 [Fetch], page 7) when the requested *key* value is not found in the database.

GDBM_REORGANIZE_FAILED

The `gdbm_reorganize` function is not able to create a temporary database. See Chapter 10 [Reorganization], page 11.

GDBM_CANNOT_REPLACE

Cannot replace existing item. This error is set by the `gdbm_store` if the requested *key* value is found in the database and the *flag* parameter is not '`GDBM_REPLACE`'. See Chapter 6 [Store], page 6, for a detailed discussion.

GDBM_ILLEGAL_DATA

Either *key* or *content* parameter was wrong in a call to to `gdbm_store` (see Chapter 6 [Store], page 6).

GDBM_OPT_ALREADY_SET

Requested option can be set only once and was already set. This error is returned by the `gdbm_setopt` function. See Chapter 14 [Options], page 15.

GDBM_OPT_ILLEGAL

The *option* argument is not valid or the *value* argument points to an invalid value in a call to `gdbm_setopt` function. See Chapter 14 [Options], page 15.

GDBM_BYTE_SWAPPED

The `gdbm_open` function (see Chapter 4 [Open], page 4) attempts to open a database which is created on a machine with different byte ordering.

GDBM_BAD_FILE_OFFSET

The `gdbm_open` function (see Chapter 4 [Open], page 4) sets this error code if the file it tries to open has a wrong magic number.

GDBM_BAD_OPEN_FLAGS

Set by the `gdbm_export` function if supplied an invalid *flags* argument. See Chapter 12 [Flat files], page 13.

GDBM_FILE_STAT_ERROR

Getting information about a disk file failed. The system `errno` will give more details about the error.

This error can be set by the following functions: `gdbm_open`, `gdbm_reorganize`.

GDBM_FILE_EOF

End of file was encountered where more data was expected to be present. This error can occur when fetching data from the database and usually means that the database is truncated or otherwise corrupted.

This error can be set by any GDBM function that does I/O. Some of these functions are: `gdbm_delete`, `gdbm_exists`, `gdbm_fetch`, `gdbm_export`, `gdbm_import`, `gdbm_reorganize`, `gdbm_firstkey`, `gdbm_nextkey`, `gdbm_store`.

20 Compatibility with standard `dbm` and `ndbm`.

`Gdbm` includes a compatibility layer, which provides traditional '`ndbm`' and older '`dbm`' functions. The layer is compiled and installed if the `--enable-libgdbm-compat` option is used when configuring the package.

The compatibility layer consists of two header files: `ndbm.h` and `dbm.h` and the `libgdbm_compat` library.

Older programs using `ndbm` or `dbm` interfaces can use `libgdbm_compat` without any changes. To link a program with the compatibility library, add the following two options to the `cc` invocation: `-lgdbm_compat-lgdbm`. The `-L` option may also be required, depending on where `gdbm` is installed, e.g.:

```
cc ... -L/usr/local/lib -lgdbm_compat -lgdbm
```

Please note that the compatibility library contains references to gdbm routines so the order in which the libraries are linked is essential. This means that the library linking order given in the above example must be respected.

Databases created and manipulated by the compatibility interfaces consist of two different files: *`file`*`.dir` and *`file`*`.pag`. This is required by the POSIX specification and corresponds to the traditional usage. Note, however, that despite the similarity of the naming convention, actual data stored in these files has not the same format as in the databases created by other `dbm` or `ndbm` libraries. In other words, you cannot access a standard UNIX `dbm` file with GNU `dbm`!

GNU `dbm` files are not `sparse`. You can copy them with the usual `cp` command and they will not expand in the copying process.

20.1 NDBM interface functions.

The functions below implement the POSIX '`ndbm`' interface:

`DBM * dbm_open` (*char* **`file`*, *int* *`flags`*, *int* *`mode`*) [ndbm]

Opens a database. The *file* argument is the full name of the database file to be opened. The function opens two files: *`file`*`.pag` and *`file`*`.dir`. The *flags* and *mode* arguments have the same meaning as the second and third arguments of `open` (see Section "open a file" in *open(2) man page*), except that a database opened for write-only access opens the files for read and write access and the behavior of the `O_APPEND` flag is unspecified.

The function returns a pointer to the `DBM` structure describing the database. This pointer is used to refer to this database in all operations described below.

Any error detected will cause a return value of '`NULL`' and an appropriate value will be stored in `gdbm_errno` (see Chapter 18 [Variables], page 25).

`void dbm_close` (*DBM* **`dbf`*) [ndbm]

Closes the database. The *dbf* argument must be a pointer returned by an earlier call to `dbm_open`.

`datum dbm_fetch` (*DBM* **`dbf`*, *datum* *`key`*) [ndbm]

Reads a record from the database with the matching key. The *key* argument supplies the key that is being looked for.

If no matching record is found, the `dptr` member of the returned datum is '`NULL`'. Otherwise, the `dptr` member of the returned datum points to the memory managed by the compatibility library. The application should never free it.

`int dbm_store` (*DBM *`dbf`, datum `key`, datum `content`, int `mode`*) [ndbm]

Writes a key/value pair to the database. The argument *dbf* is a pointer to the `DBM` structure returned from a call to `dbm_open`. The *key* and *content* provide the values for the record key and content. The *mode* argument controls the behavior of `dbm_store` in case a matching record already exists in the database. It can have one of the following two values:

`DBM_REPLACE`
Replace existing record with the new one.

`DBM_INSERT`
The existing record is left unchanged, and the function returns '`1`'.

If no matching record exists in the database, new record will be inserted no matter what the value of the *mode* is.

`int dbm_delete` (*DBM *`dbf`, datum `key`*) [ndbm]

Deletes the record with the matching key from the database. If the function succeeds, '`0`' is returned. Otherwise, if no matching record is found or if an error occurs, '`-1`' is returned.

`datum dbm_firstkey` (*DBM *`dbf`*) [ndbm]

Initializes iteration over the keys from the database and returns the first key. Note, that the word '`first`' does not imply any specific ordering of the keys.

If there are no records in the database, the `dptr` member of the returned datum is '`NULL`'. Otherwise, the `dptr` member of the returned datum points to the memory managed by the compatibility library. The application should never free it.

`datum dbm_nextkey` (*DBM *`dbf`*) [ndbm]

Continues the iteration started by `dbm_firstkey`. Returns the next key in the database. If the iteration covered all keys in the database, the `dptr` member of the returned datum is '`NULL`'. Otherwise, the `dptr` member of the returned datum points to the memory managed by the compatibility library. The application should never free it.

The usual way of iterating over all the records in the database is:

```
for (key = dbm_firstkey (dbf);
     key.ptr;
     key = dbm_nextkey (dbf))
  {
    /* do something with the key */
  }
```

The loop above should not try to delete any records from the database, otherwise the iteration is not guaranteed to cover all the keys. See Chapter 9 [Sequential], page 9, for a detailed discussion of this.

`int dbm_error` (*DBM* *`*dbf`*) [ndbm]

Returns the error condition of the database: '`0`' if no errors occurred so far while manipulating the database, and a non-zero value otherwise.

`void dbm_clearerr` (*DBM* *`*dbf`*) [ndbm]

Clears the error condition of the database.

`int dbm_dirfno` (*DBM* *`*dbf`*) [ndbm]

Returns the file descriptor of the '`dir`' file of the database. It is guaranteed to be different from the descriptor returned by the `dbm_pagfno` function (see below).

The application can lock this descriptor to serialize accesses to the database.

`int dbm_pagfno` (*DBM* *`*dbf`*) [ndbm]

Returns the file descriptor of the '`pag`' file of the database. See also `dbm_dirfno`.

`int dbm_rdonly` (*DBM* *`*dbf`*) [ndbm]

Returns '`1`' if the database *dbf* is open in a read-only mode and '`0`' otherwise.

20.2 DBM interface functions.

The functions below are provided for compatibility with the old UNIX '`DBM`' interface. Only one database at a time can be manipulated using them.

`int dbminit` (*char* *`*file`*) [dbm]

Opens a database. The *file* argument is the full name of the database file to be opened. The function opens two files: *`file`*.pag and *`file`*.dir. If any of them does not exist, the function fails. It never attempts to create the files.

The database is opened in the read-write mode, if its disk permissions permit.

The application must ensure that the functions described below in this section are called only after a successful call to `dbminit`.

`int dbmclose` (*void*) [dbm]

Closes the database opened by an earlier call to `dbminit`.

`datum fetch` (*datum* *`key`*) [dbm]

Reads a record from the database with the matching key. The *key* argument supplies the key that is being looked for.

If no matching record is found, the `dptr` member of the returned datum is '`NULL`'. Otherwise, the `dptr` member of the returned datum points to the memory managed by the compatibility library. The application should never free it.

`int store` (*datum* *`key`*, *datum* *`content`*) [dbm]

Stores the key/value pair in the database. If a record with the matching key already exists, its content will be replaced with the new one.

Returns '`0`' on success and '`-1`' on error.

`int delete` (*datum* *`key`*) [dbm]

Deletes a record with the matching key.

If the function succeeds, '`0`' is returned. Otherwise, if no matching record is found or if an error occurs, '`-1`' is returned.

`datum firstkey` (*void*) [dbm]

Initializes iteration over the keys from the database and returns the first key. Note, that the word '`first`' does not imply any specific ordering of the keys.

If there are no records in the database, the `dptr` member of the returned datum is '`NULL`'. Otherwise, the `dptr` member of the returned datum points to the memory managed by the compatibility library. The application should never free it.

`datum nextkey` (*datum* `key`) [dbm]

Continues the iteration started by a call to `firstkey`. Returns the next key in the database. If the iteration covered all keys in the database, the `dptr` member of the returned datum is '`NULL`'. Otherwise, the `dptr` member of the returned datum points to the memory managed by the compatibility library. The application should never free it.

21 Problems and bugs.

If you have problems with GNU `dbm` or think you've found a bug, please report it. Before reporting a bug, make sure you've actually found a real bug. Carefully reread the documentation and see if it really says you can do what you're trying to do. If it's not clear whether you should be able to do something or not, report that too; it's a bug in the documentation!

Before reporting a bug or trying to fix it yourself, try to isolate it to the smallest possible input file that reproduces the problem. Then send us the input file and the exact results `gdbm` gave you. Also say what you expected to occur; this will help us decide whether the problem was really in the documentation.

Once you've got a precise problem, send e-mail to `bug-gdbm@gnu.org`.

Please include the version number of GNU `dbm` you are using. You can get this information by printing the variable `gdbm_version` (see Chapter 18 [Variables], page 25).

Non-bug suggestions are always welcome as well. If you have questions about things that are unclear in the documentation or are just obscure features, please report them too.

You may contact the authors and maintainers by e-mail:

`phil@cs.wwu.edu`, `downsj@downsj.com`, `gray@gnu.org.ua`

22 Additional resources

For the latest updates and pointers to additional resources, visit `http://www.gnu.org/software/gdbm`.

In particular, a copy of `gdbm` documentation in various formats is available online at `http://www.gnu.org/software/gdbm/manual`.

Latest versions of `gdbm` can be downloaded from anonymous FTP: `ftp://ftp.gnu.org/gnu/gdbm`, or via HTTP from `http://ftp.gnu.org/gnu/gdbm`, or from any GNU mirror worldwide. See `http://www.gnu.org/order/ftp.html`, for a list of mirrors.

To track `gdbm` development, visit `http://puszcza.gnu.org.ua/projects/gdbm`.

Appendix A GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

(Index is nonexistent)