NAME

gdbmtool - examine and modify a GDBM database

SYNOPSIS

gdbmtool [**-lmNnqrs**] [**-b** *SIZE*] [**-c** *SIZE*] [**-f** *FILE*] [**--block-size**=*SIZE*] [**--cache-size**=*SIZE*] [**--file** *FILE*] [**--newdb**] [**--no-lock**] [**--no-mmap**] [**--norc**] [**--quiet**] [**--read-only**] [**--synchronize**] [*DBFILE*] [*COMMAND* [; *COMMAND*...]]

gdbmtool [**-Vh**] [**--help**] [**--usage**] [**--version**]

DESCRIPTION

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

The *DBFILE* argument supplies the name of the database to open. If not supplied, the default name **junk.gdbm** is used instead. If the named database does not exist, it will be created. An existing database can be cleared (i.e. all records removed from it) using the **--newdb** option (see below).

Unless the **-N** (**--norc**) option is given, after startup **gdbmtool** looks for file named **.gdbmtoolrc** first in the current working directory, and, if not found there, in the home directory of the user who started the program. If found, this file is read and interpreted as a list of **gdbmtool** commands.

Then **gdbmtool** starts a loop, in which it reads commands from the standard input, executes them and prints the results on the standard output. If the standard input is attached to a console, the program runs in interactive mode.

The program terminates when the **quit** command is given, or end-of-file is detected on its standard input.

Commands can also be specified in the command line, after the *DBFILE* argument. In this case, they will be interpreted without attempting to read more commands from the standard input.

If several commands are supplied, they must be separated by semicolons (properly escaped or quoted, in order to prevent them from being interpreted by the shell).

A **gdbmtool** command consists of a command verb, optionally followed by one or more 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.

Any sequence of non-whitespace characters appearing after the command verb forms an argument. If

the argument contains whitespace or unprintable characters it must be enclosed in double quotes. Within double quotes the usual escape sequences are understood, as shown in the table below:

Escape	Expansion
\a	Audible bell character (ASCII 7)
\b	Backspace character (ASCII 8)
\f	Form-feed character (ASCII 12)
\n	Newline character (ASCII 10)
\r	Carriage return character (ASCII 13)
\t	Horizontal tabulation character (ASCII 9)
\v	Vertical tabulation character (ASCII 11)
\\	Single slash

In addition, a backslash immediately followed by the end-of-line character effectively removes that character, allowing to split long arguments over several input lines.

OPTIONS

-b, **--block-size**=*SIZE*
Set block size.

-c, **--cache-size**=*SIZE*
Set cache size.

-f, **--file**=*FILE*
Read commands from *FILE*, instead of from the standard input.

-l, **--no-lock**
Disable file locking.

-m, **--no-mmap**
Do not use **mmap**(2).

-n, **--newdb**
Create the database, truncating it if it already exists.

-q, **--quiet**
Don’t print initial banner.

-r, **--read-only**

Open database in read-only mode.

-s, **--synchronize**
Synchronize to disk after each write.

-h, **--help**
Print a short usage summary.

--usage
Print a list of available options.

-V, **--version**
Print program version

SHELL COMMANDS

avail
Print the **avail list**.

bucket *NUM*
Print the bucket number *NUM* and set is as the current one.

cache
Print the bucket cache.

close
Close the currently open database.

count
Print the number of entries in the database.

current
Print the current bucket.

delete *KEY*
Delete record with the given *KEY*.

dir Print hash directory.

downgrade
Downgrade the database from the extended *numsync* format to the standard format.

export *FILE-NAME* [**truncate**] [**binary**|**ascii**]
Export the database to the flat file *FILE-NAME*. This is equivalent to **gdbm_dump**(1).

This command will not overwrite an existing file, unless the **truncate** parameter is also given. Another optional parameter determines the type of the dump (*note Flat files::). By default, ASCII dump will be created.

fetch *KEY*
Fetch and display the record with the given *KEY*.

first
Fetch and display the first record in the database. Subsequent records can be fetched using the **next** command (see below).

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

header
Print file header.

help or **?**
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.

history
Shows the command history list with line numbers. This command is available only if the program was compiled with GNU Readline.

history *COUNT*.
Shows *COUNT* latest commands from the command history.

history *N COUNT*.
Shows *COUNT* commands from the command history starting with *N*th command.

import *FILE-NAME* [**replace**] [**nometa**]
Import data from a flat dump file *FILE-NAME*. If the **replace** argument is given, any records with the same keys as the already existing ones will replace them. The **nometa** argument turns off restoring meta-information from the dump file.

list List the contents of the database.

next [*KEY*]

Sequential access: fetch and display the next record. If the *KEY* is given, the record following the one with this key will be fetched.

open *FILE*

Open the database file *FILE*. If successful, any previously open database is closed. Otherwise, if the operation fails, the currently opened database remains unchanged.

This command takes additional information from the variables **open**, **lock**, **mmap**, and **sync**. See the section **VARIABLES**, for a detailed description of these.

quit

Close the database and quit the utility.

reorganize

Reorganize the database.

set [*VAR=VALUE*...]

Without arguments, lists variables and their values. If arguments are specified, sets variables. Boolean variables can be set by specifying variable name, optionally prefixed with **no**, to set it to **false**.

snapshot *FILE FILE*

Analyzes two database snapshots and selects the most recent of them. In case of error, prints a detailed diagnostics. Use this command to manually recover from a crash. For details, please refer to the chapter **Crash Tolerance** in the **GDBM manual**.

source *FILE*

Read commands from the given *FILE*.

status

Print current program status.

store *KEY DATA*

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

sync

Synchronize the database file with the disk storage.

upgrade
Upgrade the database from the standard to the extended *numsync* format.

unset *VARIABLE*...
Unsets listed variables.

version
Print the version of **gdbm**.

DATA DEFINITIONS

The **define** statement provides a mechanism for defining key or content structures. It is similar to the **C struct** declaration:

define key|**content {** *defnlist* **}**

The *defnlist* is a comma-separated list of member declarations. Within *defnlist* the newline character looses its special meaning as the command terminator, so each declaration can appear on a separate line and arbitrary number of comments can be inserted to document the definition.

Each declaration has one of the following formats

type name
type name **[***N***]**

where *type* is a data type and *name* is the member name. The second format defines the member *name* as an array of *N* elements of *type*.

The supported types are:

type	meaning
char	single byte (signed)
short	signed short integer
ushort	unsigned short integer
int	signed integer
unsigned	unsigned integer
uint	ditto
long	signed long integer
ulong	unsigned long integer
llong	signed long long integer
ullong	unsigned long long integer

float	a floating point number
double	double-precision floating point number
string	array of characters (see the **NOTE** below)
stringz	null-terminated string of characters

The following alignment declarations can be used within *defnlist*:

offset *N*
The next member begins at offset *N*.

pad *N*
Add *N* bytes of padding to the previous member.

For example:

```
define content {
    int status,
    pad 8,
    char id[3],
    stringz name
}
```

To define data consisting of a single data member, the following simplified construct can be used:

define key|**content** *type*

where *type* is one of the types discussed above.

NOTE: The **string** type can reasonably be used only if it is the last or the only member of the data structure. That's because it provides no information about the number of elements in the array, so it is interpreted to contain all bytes up to the end of the datum.

VARIABLES

confirm, boolean
Whether to ask for confirmation before certain destructive operations, such as truncating the existing database. Default is **true**.

ps1, string
Primary prompt string. Its value can contain *conversion specifiers*, consisting of the **%** character followed by another character. These specifiers are expanded in the resulting prompt as follows:

Sequence	Expansion
%f	name of the db file
%p	program name
%P	package name (**gdbm**)
%_	horizontal space (**ASCII** 32)
%v	program version
%%	**%**

The default prompt is **%p>%_**.

ps2, string
Secondary prompt. See **ps1** for a description of its value. This prompt is displayed before reading the second and subsequent lines of a multi-line command.

The default value is **%_>%_**.

delim1, string
A string used to delimit fields of a structured datum on output (see the section **DATA DEFINITIONS**).

Default is **,** (a comma). This variable cannot be unset.

delim2, string
A string used to delimit array items when printing a structured datum.

Default is **,** (a comma). This variable cannot be unset.

pager, string
The name and command line of the pager program to pipe output to. This program is used in interactive mode when the estimated number of output lines is greater then the number of lines on your screen.

The default value is inherited from the environment variable **PAGER**. Unsetting this variable disables paging.

quiet, boolean
Whether to display welcome banner at startup. This variable should be set in a startup script file.

The following variables control how the database is opened:

cachesize, numeric
Sets the cache size. By default this variable is not set.

blocksize, numeric
Sets the block size. Unset by default.

open, string
Open mode. The following values are allowed:

newdb
Truncate the database if it exists or create a new one. Open it in read-write mode.

wrcreat or **rw**
Open the database in read-write mode. Create it if it does not exist. This is the default.

reader or **readonly**
Open the database in read-only mode. Signal an error if it does not exist.

filemode, octal
Sets the file mode for newly created database files. Default is 0644.

lock, boolean
Lock the database. This is the default.

mmap, boolean
Use memory mapping. This is the default.

coalesce, boolean
When set, this option causes adjacent free blocks to be merged which allows for more efficient memory management at the expense of a certain increase in CPU usage.

centfree, boolean
Enables central free block pool. This causes all free blocks of space to be placed in the global pool, thereby speeding up the allocation of data space.

SEE ALSO

gdbm_dump(1), **gdbm_load**(1), **gdbm**(3).

REPORTING BUGS

Report bugs to <bug-gdbm@gnu.org>.

COPYRIGHT