CGDB

for version 0.7.0, 21 March 2017

Bob Rossi (bob@brasko.net)

This manual is for GNU CGDB (version 0.7.0, 21 March 2017), the GNU ncurses based front end to GDB.

Table of Contents

Summary of CGDB

CGDB is a curses-based interface to the GNU Debugger (GDB). The goal of CGDB is to be lightweight and responsive; not encumbered with unnecessary features.

The interface is designed to deliver the familiar GDB text interface, with a split screen showing the source as it executes. The UI is modeled on the classic Unix text editor, vi. Those familiar with vi should feel right at home using CGDB.

The library responsible for communicating with GDB is called gdbwire. Those wanting to develop other interfaces to GDB are welcome to use gdbwire as the basis for their program. Many of the headaches of parsing GDB's output and annotations can be avoided by using it.

Some features offered by CGDB are:

- Syntax-highlighted source window
- Visual breakpoint setting
- Keyboard shortcuts for common functions
- Searching source window (using regexp)

1 Getting In and Out of CGDB

This chapter discusses how to start CGDB, and how to get out of it. The essentials are:

- type '`cgdb`' to start CGDB.
- type *`quit`* or *`C-d`* in the GDB window to exit.
- type *`:quit`* in the source window to exit. This even works if GDB is currently hanging, or operating a long command.

2 Understanding the core concepts of CGDB

The CGDB user interface currently consists of two windows and a status bar. The source window is currently on the top and the GDB window is on the bottom. The *status bar* currently separates the two windows.

The interface has several modes depending on which window is focused. *CGDB mode* is when the source window is focused and *GDB mode* is when the GDB window is focused.

Beginning with CGDB version 1.0, the windows are movable, and the user will be able to create as many or as few that is desired. Currently however, all of my time is spent developing the interface between CGDB and GDB. Once this is complete, the UI of CGDB will become much more polished. If you are a ncurses developer, and have spare time to work on this task, please contact me.

2.1 Understanding the source window.

The *source window* is the window that provides you a view of the source code that the debugged program is made up of. It will display to the user a single source file at a time. While the user is debugging, via `next` and `step`, CGDB will update the source file and line number to keep you informed as to where GDB is debugging.

CGDB has several features that make debugging easier than using plain old GDB. One feature you will notice right away while debugging a C, C++ or ADA program, is that the source files are syntax highlighted. This allows the user to easily navigate through the source file to look for certain places in the source code. If you would like to see another source language highlighted, contact us. To understanding how to navigate through the source window look at the commands in Section 3.1 [CGDB Mode], page 6.

In addition to showing the source code, CGDB also displays to the user the currently executing line. The line number will be highlighted green, to represent that the particular line, is the current line being debugged by GDB. Also, CGDB will display an arrow extending from the line number, to the source line. You can configure what type of display CGDB uses with the `:set executinglinedisplay` configuration option. By default, the `longarrow` display is used.

As you navigate through the source window, the current line the cursor is on will be highlighted with a block. This simply helps you keep track of where you are in the file. You can configure what type of display CGDB uses with the `:set selectedlinedisplay` configuration option. By deafult, the `block` display is used.

Also, you can set or delete breakpoints in CGDB from the source window. Simply navigate to the line that you are interested in setting a breakpoint, and hit the space bar. This will set a breakpoint on the line if one did not already exist. The line number should turn red to indicate that a breakpoint has been set. Hitting the space bar again will delete the breakpoint. If you disable the breakpoint, the line number will turn yellow, to represent the disabled breakpoint.

CGDB also supports regular expression searching within the source window. If you type `/` or `?` you can search in the source window for a string of interest. The C library regular expression functions are used to perform this search, which honors things like '`*`' or '`+`'.

The full list of commands that are available in the source window is in Section 3.1 [CGDB Mode], page 6.

2.2 Understanding the GDB window.

The *GDB window* is how CGDB allows the user to interface with the GNU debugger. If you wish to pass a command to GDB, simply type it into this window and GDB will receive the command. This interface is intended to be 100% identical to using GDB on a terminal.

There is a limited set of keys that can be typed into this window that CGDB interprets and handles, instead of sending to GDB. They are all available in Section 3.2 [GDB Mode], page 7.

CGDB attempts to buffer commands the same way they would be if you typed them at the terminal. So, if you type several commands before a single one finishes, they will each be run in order. There will be no way to stop these commands from being run besides from typing `Ctrl-C`, like you would at any normal terminal when working with GDB.

CGDB also supports regular expression searching within the GDB window.

2.3 Understanding the file dialog window.

The *file dialog window* is available to help the user view and select which file they would like to view. It provides the user with a list of all the files that make up the program being debugged. If there are no files available, because there is no program being debugged or because there is no debug symbols, then the file dialog will not open and a message will be displayed at the status bar.

You can get to the file dialog by hitting `o` when you are at the source window. Once you enter the file dialog, it is possible to leave it by hitting `q`. You can navigate the file dialog using the standard direction keys and you can even use regular expression to find your file. This can save a lot of time as the number of files grow.

The full list of commands that are available in the source window is in Section 3.3 [File Dialog Mode], page 9.

2.4 Understanding the status bar.

The *status bar* is the general purpose way for CGDB to show the user which commands they are currently typing or report errors to the user when they occur. CGDB does not use popup's or other forms of I/O to alert the user of information or problems.

While CGDB is running, you can configure it with any of the commands that are valid in CGDB's configuration file. Simply type `:` in the source window, and you will see the colon, and the rest of the command you type appear in the status bar. When you are finished typing the command that you are interested in, type `enter`. This will alert CGDB to execute the command. If at any point you would like to cancel the current command typed so far, type the cgdb mode key. This will put you back into CGDB mode. For a description of the cgdb mode key, see Section 2.5 [Switching Windows], page 4.

The full list of commands that are available in the source window is in Chapter 4 [Configuring CGDB], page 10.

2.5 Switch between windows

When CGDB is invoked, the interface is in *GDB mode*. A '`*`' at the right of the status bar indicates that input will be passed to GDB. To change the focus to the source window, hit

the `ESC` key. The *cgdb mode key* is the key that is responsible for switching the user into *CGDB mode* from a different mode. The cgdb mode key is defaulted to the `ESC` key. To change this value, look at the configuration options for CGDB. See Chapter 4 [Configuring CGDB], page 10.

The interface is now in *CGDB mode*. To switch back into *GDB mode*, press `i`. This syntax is based on the popular Unix text-editor, vi.

3 CGDB commands

CGDB can be controlled in a variety of different ways. Each mode that CGDB is in acts differently. Currently CGDB implicitly changes modes depending on which window is active. The following information will help you determine what commands are accessible during which modes.

3.1 Commands available during CGDB mode

When you are in the source window, you are implicitly in *CGDB mode*. All of the below commands are available during this mode. This mode is primarily available for the user to view the current source file, search it, or switch to a different mode.

cgdbmodekey
: Puts the user into command mode. However, you are already in this mode. This is defaulted to the `ESC` key.

i
: Puts the user into *GDB mode*.

s
: Puts the user into scroll mode in the *GDB mode*.

Ctrl-T
: Opens a new tty for the debugged program.

Ctrl-W
: Toggle the window orientation (horizontal <-> vertical).

[<number>]k
[<number>]up arrow
: Move up a line or up '`<number>`' lines.

[<number>]j
[<number>]down arrow
: Move down a line or down '`<number>`' lines.

h
left arrow
: Move left a line.

l
right arrow
: Move right a line.

Ctrl-b
page up
: Move up a page.

Ctrl-u
: Move up 1/2 a page.

Ctrl-f
page down
: Move down a page.

Ctrl-d
: Move down 1/2 a page.

gg
: Move to the top of file.

[<number>]G
: Move to the bottom of file or to a line number within the file.

m[a-zA-Z]
: Set a mark at the cursor position. A lower case letter sets a local mark that is valid within one file. An upper case letter sets a global mark valid between files. If there are ten files, each file can have a mark 'a', but only one can have a mark 'A'.

'[a-zA-Z]
: Jump to the corresponding local or global mark.

''
: Jump to the last jump location.

'.
: Jump to the currently executing line.

/
: search from current cursor position.

?
: reverse search from current cursor position.

n
: next forward search.

N
: next reverse search.

o
: open the file dialog.

spacebar
: Sets a breakpoint at the current line number.

t
: Sets a temporary breakpoint at the current line number.

-
: Shrink source window 1 line or column (depending on split orientation).

=
: Grow source window 1 line or column (depending on split orientation).

_
: Shrink source window 25%

+
: Grow source window 25%

Ctrl-l
: Clear and redraw the screen.

F5
: Send a run command to GDB.

F6
: Send a continue command to GDB.

F7
: Send a finish command to GDB.

F8
: Send a next command to GDB.

F10
: Send a step command to GDB.

3.2 Commands available during GDB mode

When in *GDB mode*, the user is in command mode or scroll mode. When in command mode, the user is typing in commands to interact with GDB. When in scroll mode, the user can scroll through the GDB output. You can enter scroll mode by typing *page up* and quit scroll mode by typing *q*, *i* or *enter*.

The commands available in Section 3.2 [GDB Mode], page 7 when in command mode are:

cgdbmodekey
: Switch back to source window. This is defaulted to the ESC key.

page up
: Move up a page and enter scroll mode.

When in command mode, CGDB will pass along any other keys to the readline library. When readline has determined that a command has been received, it alerts CGDB, and a command is then sent to GDB.

The commands available in Section 3.2 [GDB Mode], page 7 when in scroll mode are:

cgdbmodekey
: Switch back to source window. This is defaulted to the `ESC` key.

page up
: Move up a page.

page down
: Move down a page.

Ctrl-u
: Move up 1/2 a page.

Ctrl-d
: Move down 1/2 a page.

F11
home key
gg
: Go to the beginning of the GDB buffer.

F12
end key
G
: Go to the end of the GDB buffer.

k
up arrow
Ctrl-p
: Move up a line.

j
down arrow
Ctrl-n
: Move down a line.

/
: search from current cursor position.

?
: reverse search from current cursor position.

n
: next forward search.

N
: next reverse search.

m[a-z]
: Set a mark at the cursor position.

'[a-z]
: Jump to the corresponding mark.

''
: Jump to the last jump location.

'.
: Jump to the last line.

q
i
enter
: Exit scroll mode.

3.3 Commands available during the file dialog mode

The file dialog is primarily used to allow the user to find and open a source file that the program they are debugging is made up of. The file dialog will be full screen, and will list each file that the debugged program is made up of. A usual instance of the file dialog would be to open it up from the source window using the *o* key, and then to search for the file of interest. If you are looking for foo.c type */foo.c*, press *enter* once to finish the regular expression and again to select the file.

The commands available in the file dialog are:

q Will exit the file dialog, and return to the source window.

k
up arrow Move up a line.

j
down arrow
Move down a line.

h
left arrow
Move left a line.

l
right arrow
Move right a line.

Ctrl-b
page up Move up a page.

Ctrl-u Move up 1/2 a page.

Ctrl-f
page down Move down a page.

Ctrl-d Move down 1/2 a page.

gg Move to the top of the file dialog.

[<number>]G
Move to the bottom of the file dialog or to a line number within the file dialog.

/ search from current cursor position.

? reverse search from current cursor position.

n next forward search.

N next reverse search.

enter Select the current file.

4 CGDB configuration commands

There may be several features that you find useful in CGDB. CGDB is capable of automating any of these commands through the use of the config file called `cgdbrc`. It looks in `$HOME/.cgdb/` for that file. If it exists, CGDB executes each line in the file in order. It is as if the user typed in all the commands into the status bar after the tui was initialized.

The following variables change the behavior of some aspect of CGDB. Many of these commands may be abbreviated in some way, and all boolean commands my be negated by appending '`no`' to the front. For example: *`:set ignorecase`* turns on case-insensitive searching; while *`:set noignorecase`* turns on case-sensitive searching.

`:set asr`
`:set autosourcereload`
: If this is on, CGDB will automatically reload a source file if it has changed since CGDB has opened it. If it is off, the file will never be reloaded, until you start CGDB again. The default is on. This feature is useful when you are debugging a program, then you modify a source file, recompile, and type *`r`* in GDB's CLI window. The file in this case will be updated to show the new version. Note, CGDB only looks at the timestamp of the source file to determine if it has changed. So if you modify the source file, and didn't recompile yet, CGDB will still pick up on the changes.

`:set cgdbmodekey=key`
: This option is used to determine what key puts CGDB into *CGDB Mode*. By default, the *`ESC`* key is used. *key* can be any normal key on the keyboard. It can also be any keycode, as long as the keycode notation is used. This option is especially useful when the user wants to use readline in vi mode. If the user types `set cgdbmodekey=<PageUp>` then the *`Page Up`* key will put CGDB into CGDB mode and the *`ESC`* key will flow through to readline.

`:set color`
: This option is used to enable or disable color support in CGDB. The default is on. When enabled, CGDB can display color when appropriate. This typically occurs when syntax highlighting source files in the source viewer. When disabled, CGDB will not display colors. It may instead use other terminal attributes for syntax highlighting, including bold and reverse attributes. See the `highlight` commamnd and look for the '`cterm`' option.

  It should be noted that even when color is enabled in CGDB, if ncurses declares that the terminal does not support colors, CGDB will not use colors.

`:set dwc`
`:set debugwincolor`
: This option controls if the debug window will display colors or not. The default value is on. When enabled, if GDB or the program being debugged output an ANSI escape code representing color, then CGDB will display the corresponding color instead of the escape code. When disabled, CGDB will display the ANSI escape code in the debug window. See https://en.wikipedia.org/wiki/ANSI_escape_code#Colors to learn more about ANSI escape codes.

:set dis

:set disasm

This option is used to enable or disable showing assembly code in cgdb. The default value is off. When off, CGDB will display the source code to the user. When no source code is available, CGDB will display the assembly code. When this option is enabled, CGDB will display the mixed source and assembly when the source code is enabled and assembly code when the source code is not available. Please note that when assembly mode is displayed, it's displayed per function instead of per file.

:set eld=style
:set executinglinedisplay=style

Set the executing line display to *style*. Possible values for *style* are '`shortarrow`', '`longarrow`', '`highlight`', and '`block`'. Changes the display that is used to indicate the currently executing line in the source viewer. The default value is '`longarrow`'. The '`shortarrow`' option draws an arrow next to the source line number. The '`longarrow`' option also draws an arrow next to the source line number, but extends the arrow until the source code. The '`highlight`' option draws the entire line in inverse video and the '`block`' option draws an inverse block (cursor) next to the source code being executed.

:set hls
:set hlsearch

When enabled, if there is a previous search pattern, highlight all its matches. The default is disabled.

:set ic
:set ignorecase

Sets searching case insensitive. The default is off.

:set sld=style
:set selectedlinedisplay=style

Set the selected line display to *style*. Possible values for *style* are '`shortarrow`', '`longarrow`', '`highlight`', and '`block`'. Changes the display that is used to indicate the currently selected line in the source viewer. The default value is '`block`'. The '`shortarrow`' option draws an arrow next to the source line number. The '`longarrow`' option also draws an arrow next to the source line number, but extends the arrow until the source code. The '`highlight`' option draws the entire line in inverse video and the '`block`' option draws an inverse block (cursor) next to the source code being selected.

:set showmarks

This option controls if the source window will show marks or not. The default value is on. When enabled, CGDB will show the marks that the user has set in the vertical bar in the source window separating the line number from the source code.

:set sdc
:set showdebugcommands
: If this is on, CGDB will show all of the commands that it sends to GDB. If it is off, CGDB will not show the commands that it gives to GDB. The default is off.

:set syn=style
:set syntax=style
: Sets the current highlighting mode of the current file to have the syntax *style*. Possible values for *syntax* are '`c`', '`asm`', '`d`', '`go`', '`ada`', '`rust`', '`off`', '`no`', '`on`' or '`yes`'. Normally, the user will never have to do this, since CGDB automatically detects what syntax a file should be based on its file extension. However, this feature can currently be useful for debugging purposes. '`on`' and '`yes`' enable syntax highlighting based on the source files extension. '`off`' and '`no`' disable syntax highlighting. The remaining values set the specific language highlighters independent of the source files extension.

:set to
:set timeout
: This option is used along with the *ttimeout* option to determine the behavior CGDB should have when it receives part of a mapped key sequence or a keyboard code sequence. If this option is on, CGDB will time out on both user defined mappings and on key codes from the keyboard. If this option is off, user defined mappings will not be timed out on. In this case, CGDB will determine if it should time out on key codes from the keyboard by examining the *ttimeout* option. To determine how CGDB will time out on mappings and key codes, and what time out lengths CGDB will use, please refer to the chart in Chapter 6 [Key User Interface], page 19. The default value for this option is on.

:set tm=delay
:set timeoutlen=delay
: This option is used along with the *ttimeoutlen* option. It represents the number of milliseconds that CGDB should wait for a key code from the keyboard or for a mapped key sequence to complete. If *delay* is 0, CGDB immediately accepts each character it receives. This will prevent any mappings or key codes to complete. *delay* may be any value between 0 and 10000, inclusive. The default setting for the *delay* variable is 1000 (one second).

:set ttimeout
: This option is used along with the *timeout* option to determine the behavior CGDB should have when it receives part of keyboard code sequence. If this option is on, CGDB will time out on key codes from the keyboard. If this option is off, CGDB will determine if it should time out on key codes from the keyboard by examining the *timeout* option. To determine how CGDB will time out on key codes, what what time length it will use, please refer to the chart in Chapter 6 [Key User Interface], page 19. The default value for this option is on.

:set ttm=delay
:set ttimeoutlen=delay
: This option is used along with the *ttimeoutlen* option. It represents the number of milliseconds that CGDB should wait for a key code from the keyboard. If *delay* is 0, CGDB immediately accepts each character it receives. This will prevent any key codes to complete. *delay* may be any value between 0 and 10000, inclusive. The default setting for the *delay* variable is 100 (one tenth of a second).

:set ts=number
:set tabstop=number
: Sets the number of spaces that should be rendered on the screen for `TAB` characters. The default value for *number* is 8.

:set wmh=number
:set winminheight=number
: The minimal height of a window. Windows will never become smaller than this value. The default value for *number* is 0. This is useful when *winsplitorientation* is set to '`horizontal`'.

:set wmw=number
:set winminwidth=number
: The minimal width of a window. Windows will never become smaller than this value. The default value for *number* is 0. This is useful when *winsplitorientation* is set to '`vertical`'.

:set winsplit=style
: Set the split point between source and GDB window. This is especially useful as an init setting in your `cgdbrc` file. See Chapter 4 [Configuring CGDB], page 10. The possible values for *style* are '`src_full`', '`src_big`', '`even`', '`gdb_big`', and '`gdb_full`'.

:set wso=style
:set winsplitorientation=style
: Sets the window split orientation to either '`horizontal`' (which places the source window above and the GDB window below), or '`vertical`' (hich places the source window on the left and the GDB window on the right). See Chapter 4 [Configuring CGDB], page 10.

:set ws
:set wrapscan
: Searches wrap around the end of file. The default is on.

:c
:continue
: Send a continue command to GDB.

:down : Send a down command to GDB.

:e
:edit : reloads the file in the source window. this can be useful if the file has changed since it was opened by cgdb.

:f
:finish Send a finish command to GDB.

:help This will display the current manual in text format, in the *source window*.

:logo This will display one of CGDB's logos in the *source window*.

:hi group cterm=attributes ctermfg=color ctermbg=color term=attributes
:highlight group cterm=attributes ctermfg=color ctermbg=color term=attributes
Set the *color* and *attributes* for a highlighting group. The syntax mimics vim's "highlight" command. Possible values for *group*, *attributes* and *color* are available in Chapter 5 [Highlighting Groups], page 16.

You can give as many or as few of the name=value pairs as you wish, in any order. '`ctermfg`' and '`ctermbg`' set the foreground and background colors. These can be specified by color number or by using the same color names that vim uses. When CGDB is linked with ncurses, the number you use to represent the color can be between -1 and COLORS. When CGDB is linked against curses, it must be between 0 and COLORS.

'`cterm`' sets the video attributes for color terminals. '`term`' sets the video attributes for monochrome terminals. Some examples are,

```
:highlight Logo cterm=bold,underline ctermfg=Red ctermbg=Black
:highlight Normal cterm=reverse ctermfg=White ctermbg=Black
:hi Normal term=bold
```

:insert Move focus to the GDB window.

:n
:next Send a next command to GDB.

:q
:quit Quit CGDB.

:r
:run Send a run command to GDB.

:start Send a start command to GDB.

:k
:kill Send a kill command to GDB.

:s
:step Send a step command to GDB.

:syntax Turn the syntax on or off.

:u
:until Send an until command to GDB.

:up Send an up command to GDB.

:map lhs rhs
Create a new mapping or overwrite an existing mapping in CGDB mode. After the command is run, if *lhs* is typed, CGDB will get *rhs* instead. For more details on how to use the map command look in Section 6.2 [Using Maps], page 20.

:unm lhs

:unmap lhs

Delete an existing mapping from CGDB mode. *lhs* is what was typed in the left hand side when the user created the mapping. For example, if the user typed `:map a<Space>b foo` then the user could delete the existing mapping with `:unmap a<Space>b`.

:im lhs rhs

:imap lhs rhs

Create a new mapping or overwrite an existing mapping in GDB mode. After the command is run, if *lhs* is typed, CGDB will get *rhs* instead. For more details on how to use the map command look in Section 6.2 [Using Maps], page 20.

:iu lhs

:iunmap lhs

Delete an existing mapping from GDB mode. *lhs* is what was typed in the left hand side when the user created the mapping. For example, if the user typed `:imap a<Space>b foo` then the user could delete the existing mapping with `:iunmap a<Space>b`.

5 CGDB highlighting groups

CGDB is capable of using colors if the terminal it is run in supports them. Until version 0.6.1, CGDB did not allow the user to configure these colors in any way. CGDB color use is now fully configurable.

CGDB's modeled its use of color highlighting after vim. Any data that will be colored in the terminal is represented by a highlighting group. A *highlighting group* represents data that should be formatted using foreground colors, background colors and attributes. There are currently several types of highlighting groups in CGDB. There are syntax highlighting groups, which represent syntax highlighting of sources files. There are also User Interface groups, which represent things like CGDB's logo, or the status bar.

Each highlighting group has a default set of attributes and colors associated with it. You can modify a highlighting groups properties by using the highlight command. See Chapter 4 [Configuring CGDB], page 10.

Note that CGDB currently supports using the same background color the terminal was using before CGDB was started. However, this only works when CGDB was linked with ncurses. If you link CGDB with curses, then CGDB will force the background to Black.

5.1 The different highlighting groups

Below is a list of all the highlighting groups that CDGB will use when syntax highlighting source files.

`Statement`
: This represents the keywords a language defines.

`Type`
: This represents the types a language defines.

`Constant`
: This represents either a string or numeric value.

`Comment`
: This represents the comments in a source file.

`PreProc`
: This represents the C/C++ preprocessor commands.

`Normal`
: This represents all normal text.

Below is a list of all the highlighting groups that CGDB will use when it is displaying it's User Interface.

`StatusLine`
: This represents the *status bar* in CGDB. The file dialog's status bar also uses this group.

`Search`
: This represents the group used when displaying the previous search in either the source window, the GDB window in scroll mode, or the *file dialog window*. This is only used when the *hlsearch* option is enabled.

`IncSearch`
: This represents the group used when the user is searching in either the source window, the GDB window in scroll mode, or the *file dialog window*.

`SelectedLineArrow`
: This represents the group used when the user has the `selectedlinedisplay` set to `shortarrow` or `longarrow`.

`ExecutingLineArrow`
: This represents the group used when the user has the `executinglinedisplay` set to `shortarrow` or `longarrow`.

`SelectedLineHighlight`
: This represents the group used when the user has the `selectedlinedisplay` option set to `highlight`.

`ExecutingLineHighlight`
: This represents the group used when the user has the `executinglinedisplay` option set to `highlight`.

`SelectedLineBlock`
: This represents the group used when the user has the `selectedlinedisplay` option set to `block`.

`ExecutingLineBlock`
: This represents the group used when the user has the `executinglinedisplay` option set to `block`.

`Breakpoint`
: This represents the group that is used when CGDB displays a line that has a breakpoint set.

`DisabledBreakpoint`
: This represents the group that is used when CGDB displays a line that has a disabled breakpoint set.

`SelectedLineNr`
: This represents the group that is used when CGDB is displaying the currently selected line. This is the line that the cursor is on.

`ExecutingLineNr`
: This represents the group that is used when CGDB is displaying the currently executing line.

`ScrollModeStatus`
: This represents the group that is used when CGDB is displaying the currently selected line number when in Section 3.2 [GDB Mode], page 7 and the user is scrolling through the GDB buffer.

`Logo` This is the group CGDB uses to display its logo on startup when no source file can be auto detected.

`Mark` This is the group CGDB uses to display a mark in the source window. Marks are displayed when the `showmarks` option is enabled. See the `showmarks` option for more detail.

5.2 The different attributes

CGDB supports many of the attributes that curses provides. It will apply the attributes to the output window, but it is up to the terminal you are using to support such features.

The list of attributes that CGDB currently supports is below.

`normal`
`NONE` This will leave the text normal. Uses A_NORMAL curses attribute.

`bold` This will make the text appear bold. Uses A_BOLD curses attribute.

`underline`
This will underline the text. Uses A_UNDERLINE curses attribute.

`reverse`
`inverse` This will reverse the foreground and background colors. Uses A_REVERSE curses attribute.

`standout` This is the best highlighting mode of the terminal. Uses A_STANDOUT curses attribute.

`blink` This will cause the text to blink. Uses A_BLINK curses attribute.

`dim` This will cause the text to be 1/2 bright. Uses A_DIM curses attribute.

5.3 The different colors

CGDB supports several colors, depending on how many colors your terminal supports. Below is a chart of the colors that CGDB provides. The values for each color represent the defines from the curses header file passed to init_pair() to ask curses to create a new color.

COLOR NAME	Color	Bold attribute
Black	COLOR_BLACK	No
DarkBlue	COLOR_BLUE	No
DarkGreen	COLOR_GREEN	No
DarkCyan	COLOR_CYAN	No
DarkRed	COLOR_RED	No
DarkMagenta	COLOR_MAGENTA	No
Brown, DarkYellow	COLOR_YELLOW	No
LightGray, LightGrey, Gray, Grey	COLOR_WHITE	No
DarkGray, DarkGrey	COLOR_BLACK	Yes
Blue, LightBlue	COLOR_BLUE	Yes
Green, LightGreen	COLOR_GREEN	Yes
Cyan, LightCyan	COLOR_CYAN	Yes
Red, LightRed	COLOR_RED	Yes
Magenta, LightMagenta	COLOR_MAGENTA	Yes
Yellow, LightYellow	COLOR_YELLOW	Yes
White	COLOR_WHITE	Yes

6 CGDB key user interface

The Key User Interface is how CGDB receives input from the user. It is usually referred to as the *KUI*. CGDB simply asks the KUI for the next key the user typed and the KUI will provide it.

The KUI has 2 major responsibilities besides reading normal user input and providing it to CGDB. It needs to detect when the user has typed a user defined map or when the user has hit a special key on the keyboard.

A user defined map, or simply *map*, is used to change the meaning of typed keys. Some users may refer to this type of functionality as a *macro*. An example would be `map a b`. If the user then typed the `a` character, the KUI would detect that it was mapped to `b` and return `b` to CGDB.

When the user types a special key on the keyboard, a *key code* is sent to CGDB. Typically, keys like *HOME*, *DEL*, F1, etc, when pressed will send several characters to the application instead of just one character like a normal key does. These characters combined are called a *key sequence*. The KUI is responsible for assembling the key sequences back together and reporting to CGDB that a particular key was typed by the user. The *ESC* key is special because typically most key codes start with that key. This usually gives all key codes a common first key in its key sequence. The KUI uses the terminfo database to determine what key sequences are sent by which keycodes. There are a few commonly used key sequences that are hard coded into CGDB.

A major challenge the KUI has to overcome is determining when a map or a key sequence is received. The KUI sometimes will need to read more than one character to determine this. For example, if the user has 2 maps, `map abc def` and `map abd def`, the KUI would have to buffer at least the characters `a` and `b` before it could determine if the user was going to type a map. After the next key press, if the user types `c` or `d` then a map was received and the KUI will return *d e f* to CGDB. Otherwise, no map was received and the KUI must return *a b* to CGDB.

The options *timeout*, *ttimeout*, *timeoutlen* and *ttimeoutlen* can be used to tell the KUI if it should timeout on partial mappings or key sequences, and if so, how long it should wait before timing out.

6.1 The KUI's time out options

The KUI may be configured to time out on either maps or key sequences.

When the KUI is matching a partial map or key sequence it is capable of timing out. This means it will simply accepts the keys it has received so far if a certain amount of time elapses between key presses. This is obvious when the user is typing a map because the user must press each key individually. For partial key sequences, this is less obvious. That is because the user only presses a single key, but multiple characters are sent to CGDB. The table below describes how the user can configure the KUI to time out on key codes or maps. The *timeout* and *ttimeout* options control this functionality.

timeout	**ttimeout**	**action**
off	off	do not time out
on	on or off	time out on maps and key codes

off on time out on key codes

It is also possible to tell the KUI how long to wait before timing out on a partial match. If *timeout* is on, then the KUI will wait a certain amount of time for the next character, when matching a map, before it decides a match is no longer possible. If *timeout* or *ttimeout* is on, then the KUI will wait a certain amount of time for the next character, when matching a key sequence, before it decides a match is no longer possible. The *timeoutlen* and *ttimeoutlen* options can be configured by the user to tell the KUI how long to wait before timing out. The table below describes when the KUI uses which option.

timeoutlen	**mapping delay**	**key code delay**
< 0	*timeoutlen*	*timeoutlen*
>= 0	*timeoutlen*	*ttimeoutlen*

A value of 0 means that the KUI will time out right away. It will not be possible to match a map or key code in this circumstance.

A common problem could be that when the user types a special key like the left or right arrows, CGDB will go into the source mode and not perform the action requested by the user. This typically means that the key code delay is to small. If you try setting the option `set ttimeoutlen=1000` CGDB should start acting like the user expects. If not, please report this to the CGDB mailing list.

6.2 Using maps

CGDB fully supports the use of maps. It allows the user to change the meaning of typed keys. For example, you could have the following map `:map <F2> ip<Space>argc<CR>`.

When the user is in CGDB mode and they hit `F2`, the value of the map will be used instead. The `i` key will first be received by CGDB, and it will put the user into insert mode. Next, CGDB will get *`p argc`* followed by the `Enter` key.

CGDB currently supports two mapping lists. Any mapping that was added with the *map* command will be used by CGDB when it is in CGDB mode. You can delete a mapping that you have created with the *map* command with the *unmap* command. If you want to have mappings in GDB mode, you can use the *imap* command. Similarly, *iunmap* will delete a mapping in the *imap* set. Some examples of this would be

```
map a<Space>b foo
unmap a<Space>b

imap a<CR>b foo
iunmap a<CR>b
```

6.3 Understanding keycodes

The above example could use a little more explaining for people unfamiliar with vim maps. The map takes a key and a value. They are separated by a space. Neither the key or value can have a space in them, or it is considered to be the separator between the key and value. If the user desires to have a space in either the key or value part of a map, they can use the keycode notation <Space>. Below is a table of the keycodes in *keycode notation* form. The keycode notation can be used in any mapping command.

notation	**meaning**
<Esc>	escape key
<Up>	cursor up key
<Down>	cursor down key
<Left>	cursor left key
<Right>	cursor right key
<Home>	home key
<End>	end key
<PageUp>	page up key
<PageDown>	page down key
<Del>	delete key
<Insert>	insert key
<Nul>	zero
<Bs>	backspace key
<Tab>	tab key
<NL>	linefeed
<FF>	formfeed
<CR>	carriage return
<Space>	space
<Lt>	less-than
<Bslash>	backslash
<Bar>	vertical bar
<F1> - <F12>	function keys 1 to 12
<C-...>	control keys
<S-...>	shift keys

7 Sending I/O to the program being debugged

If the program being debugged takes input on the terminal it is recommended that the user start the program on one terminal, and attach to it with CGDB from another terminal. This is the easiest way to pass input to the debugged program.

8 Allowing terminal control flow in CGDB

A user can typically set there control flow behavior by using the stty command like so `stty -ixon -ixoff`. This will disable control flow on the terminal where CGDB is started. If you want to turn control flow back on you can type `stty ixon ixoff`. If flow control is on, when the user types *`Ctrl-s`*, the terminal stops. When the user types *`Ctrl-q`*, the terminal restarts. When using readline, the *`Ctrl-s`* character usually does a forward search. So, if you want to get this, or other functionality out of readline, simply turn off control flow and start CGDB.

9 Building CGDB from source

Building CGDB from source requires several packages. First, CGDB is hosted at GitHub. You can determine how to get CGDB from source by looking here: https://cgdb.github.io/

Once you have the source to CGDB, now you can begin to build it. You will of course need many packages to build CGDB. Below is a list of all of them that are required to build CGDB.

`GNU Make` I have successfully used version 3.79.1, however, older versions probably will work.

`GNU GCC` The GNU C compiler. I've compiled CGDB with versions as old as 2.9.5, and as new as 4.0.2.

`GNU Readline`
The GNU readline library version 5.1. CGDB will not work with versions before 5.1. Readline was modified specifically to work with CGDB.

`GNU Ncurses`
I have successfully used libncurses.so.5 successfully. However, older versions probably will work.

Below is a list of optional packages you will need, if modifying certain files in CGDB.

`GNU Flex` If you modify any files with an extension of `.l`, you will have to have flex installed. I have used flex 2.5.4 to build CGDB.

`GNU Texinfo`
If you modify `doc/cgdb.texinfo`, then you will be required to have this package installed. I have used version 4.7 to build the documentation for CGDB.

CGDB uses autoconf/automake to build its configure scripts and makefiles. So, if you change any of the autoconf/automake files, you will need this software installed.

`GNU Automake`
This has the program aclocal, and must be version Version 1.9.5.

`GNU Autoconf`
This has the program autoconf, and must be version 2.59.

`GNU m4` This has the program m4, and must be version 1.4.3.

Appendix A Copying This Manual

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, 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 show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public 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.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software

which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) yyyy  name of author

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

Index