Graphical User Interfaces for gdb

Summary (no quick start this time)

This document explains how to use Emacs as a graphical front end for gdb to debug the ManalinkEh.dll. More on gdb and how to use it is discussed in a separate paper.

As gdb comes without a graphical user interface (so-called GUI; there is a built-in one but it doesn't work under Windows) this paper shows a few alternatives and the one that works given out not-so-standard setting consisting of ManalinkEh.dll ("the dll") and Magic.exe ("the exe").

I assume you have gdb (as part of MinGW64) installed and that you can call it from the command prompt (PATH environment variable is set accordingly).

Why do I need a graphical user interface?

Actually you don't need one as gdb can be used with the command line interface. Occasionally it is helpful to see local variables or a longer piece of code at the same time without switching windows etc. so having a GUI makes sense. Besides, it allows you to use the mouse to some extent.

Which GUI should I use?

As with the debuggers I tried a couple (if you consult the Internet you find several recommendations) and stuck with what worked. Which was not that much unfortunately. Most of the time I couldn't load the dll separately and set breakpoints as most GUIs seem to assume that you always debug an exe and take it from there.
As last time let me outline what didn't work or what I didn't try to save you the effort to try it yourself. Feel free to skip this section and forward to „Emacs"if you are not interested. If you want to try out the alternatives and make progress I would welcome your feedback.

gdbtui

The good news is that gdb has a built-in GUI based on ncurses. The bad news is that it is not available under Windows and you must jump through several hoops to get something that barely works. I want something easier to set up.

Eclipse CDT

Eclipse is an Integrated Development Environment (IDE) similar to Visual Studio. Its stand-alone debugger CDT can be downloaded here: https://www.eclipse.org/cdt/downloads.php. Unpack it, put it into your Program Files (x86) or Program Files folder (the former for the 32-bit version, the latter for the 64-bit version). Before running it the first time you must install the latest version of the Java Development Kit (JDK), the usual Java Runtime Environment (JRE) won't do. Make sure you use a 32-bit version of Eclipse with a 32-bit version of JDK or 64-bit for both. In case you get an error message when starting Eclipse (double-click on cdtdebug.exe) you probably must adjust the ini file as shown here: https://javahungry.blogspot.com/2018/10/solved-java-started-returned-exit-code-13-error-eclipse.html. Sounds like a ton of work but is actually quite easy. The issue is that after starting CDT it asks which executable you want to debug… So no way to break into the dll and when an error occurs it shows you the right function but without source code. Nice, but doesn't help.

gdbgui

A stand-alone GUI for gdb that uses your browser as a front-end. You can download an executable here: https://github.com/cs01/gdbgui/releases (I assume you don't want to set up Python to run it from there). Make sure to adjust the firewall settings under Control Panel-Windows Defender Firewall and add gdbgui to the list of permitted programs (otherwise you can start it but your browser won't show anything as the communication gets blocked). The GUI looks nice, but alas, I didn't find a way to debug the dll separately from the exe. Besides that I miss the auto-complete feature when pressing TAB (TAB is used differently in a browser).

Emacs

Emacs is an Operating System that doubles as an editor. No, I'm kidding but Emacs can do a lot of things beyond what a common editor can. Among other things it can run gdb and do basically everything I want. You can download it from https://www.gnu.org/software/emacs/download.html or directly from the FTP server (https://ftp.gnu.org/gnu/emacs/windows/). Take the latest version (emacs-26 at the time of writing) and pick either the 32-bit version emacs-26.3-i686.zip or the 64-bit version emacs-26.3-x86_64.zip. If you are in doubt use the 32-bit version. Unpack it, put it into your Program Files (x86) or Program Files folder (the former for the 32-bit version, the latter for the 64-bit version). Double-click on bin/runemacs.exe to start it.

Like gdb Emacs has a ton of functions and you might easily get lost. A cheat sheet to get you started is here: http://www.rgrjr.com/emacs/emacs_cheat.html.

By convention *C-<key>* denotes the Control (Ctrl) key plus <key>. *M-<key>* denotes the Alt key plus <key>. So *C-x C-c* means press Control and keep it, then press x and c. This ends Emacs. The Escape key serves as alternative M (Alt) key. If you need to quit a command press *C-g* instead.

Once you started it go to the Options menu and check "Use CUA keys". This ensures you can use C-x, C-c, C-v and C-z as usual (Emacs has another convention for cut, copy, paste and undo which might otherwise drive you crazy). Afterwards click Options-Save Options. This creates a configuration file with name .emacs in the folder %USERNAME%\AppData\Roaming (under Windows 10, in case you still have an older version see https://www.gnu.org/software/emacs/manual/html_node/emacs/Windows-HOME.html).

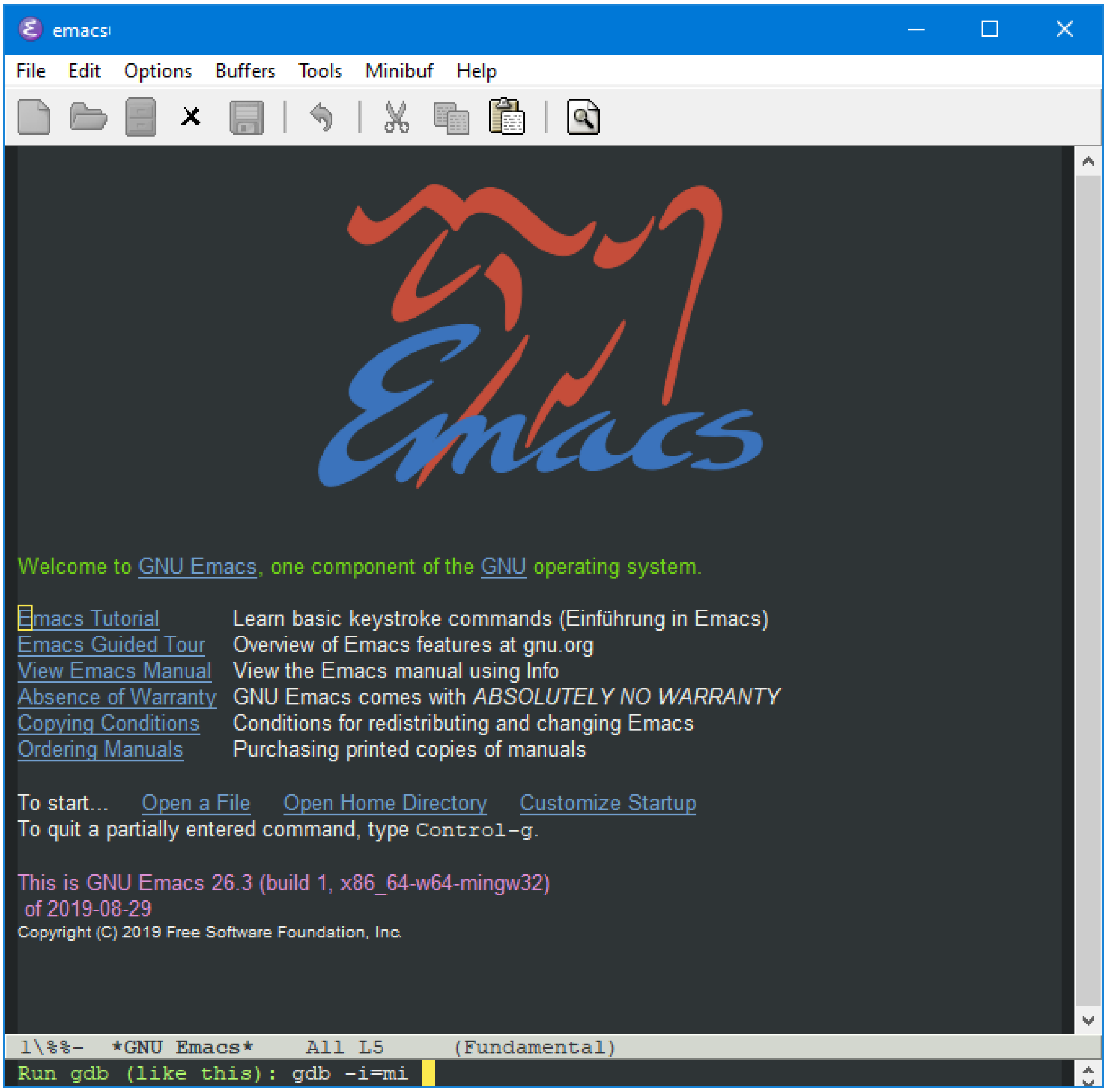
Under Options-Customize Emacs-Custom Themes you find some colour themes in case you don't like the current theme. Mine is tango-dark.

Before continuing look for the .emacs file and open it with an editor (you can use Emacs if you want) or take mine and copy it inside the folder (mine contains the code below plus the Annex). The contents should look like this (note the cua-base and tango-dark settings):

```
(custom-set-variables
 ;; custom-set-variables was added by Custom.
 ;; If you edit it by hand, you could mess it up, so be careful.
 ;; Your init file should contain only one such instance.
 ;; If there is more than one, they won't work right.
 '(cua-mode t nil (cua-base))
 '(custom-enabled-themes (quote (tango-dark))))
(custom-set-faces
 ;; custom-set-faces was added by Custom.
 ;; If you edit it by hand, you could mess it up, so be careful.
 ;; Your init file should contain only one such instance.
 ;; If there is more than one, they won't work right.
 )
```

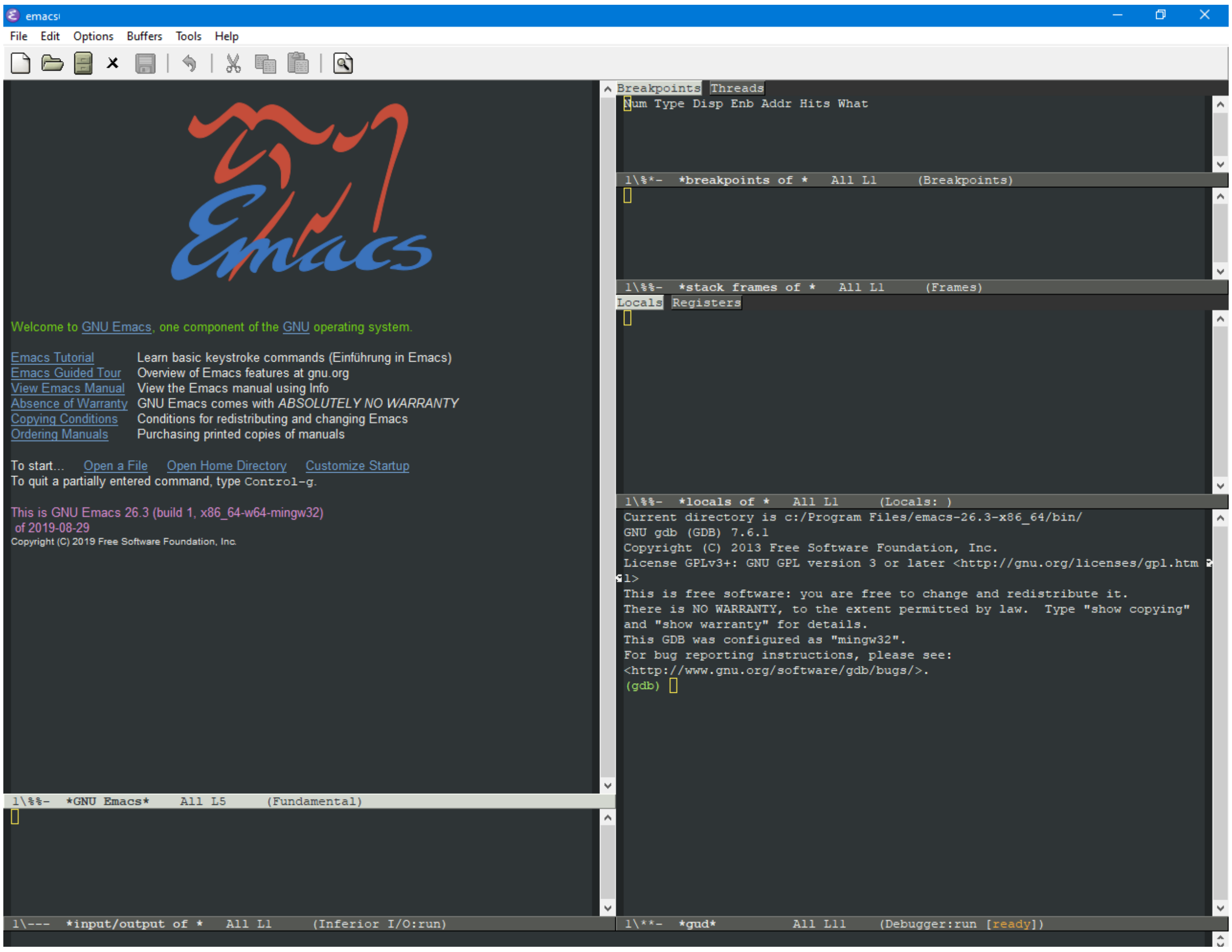
Add the code in Annex 1 at the end of the file and save it. This makes sure you have a nicer look very soon. Alternatively you can skip that and use gdb-many-windows instead (will be explained in a second). I liked the option in Annex 1 better.

Now restart Emacs. You can start gdb (and generally execute a bunch of commands) with *M-x gdb*. Delete the wish.exe. Your Emacs window should look like this:



```
Welcome to GNU Emacs, one component of the GNU operating system.

Emacs Tutorial       Learn basic keystroke commands (Einführung in Emacs)
Emacs Guided Tour    Overview of Emacs features at gnu.org
View Emacs Manual    View the Emacs manual using Info
Absence of Warranty  GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions   Conditions for redistributing and changing Emacs
Ordering Manuals     Purchasing printed copies of manuals

To start...   Open a File   Open Home Directory   Customize Startup
To quit a partially entered command, type Control-g.

This is GNU Emacs 26.3 (build 1, x86_64-w64-mingw32)
 of 2019-08-29
Copyright (C) 2019 Free Software Foundation, Inc.

 1\%%-  *GNU Emacs*    All L5      (Fundamental)
Run gdb (like this): gdb -i=mi
```

Press Enter and you get something like this:



Welcome to GNU Emacs, one component of the GNU operating system.

Emacs Tutorial	Learn basic keystroke commands (Einführung in Emacs)
Emacs Guided Tour	Overview of Emacs features at gnu.org
View Emacs Manual	View the Emacs manual using Info
Absence of Warranty	GNU Emacs comes with *ABSOLUTELY NO WARRANTY*
Copying Conditions	Conditions for redistributing and changing Emacs
Ordering Manuals	Purchasing printed copies of manuals

To start... Open a File Open Home Directory Customize Startup
To quit a partially entered command, type `Control-g`.

This is GNU Emacs 26.3 (build 1, x86_64-w64-mingw32)
 of 2019-08-29
Copyright (C) 2019 Free Software Foundation, Inc.

```
Num Type Disp Enb Addr Hits What
1\%*-  *breakpoints of *   All L1     (Breakpoints)
1\%%-  *stack frames of *   All L1     (Frames)
1\%%-  *locals of *   All L1     (Locals: )
Current directory is c:/Program Files/emacs-26.3-x86_64/bin/
GNU gdb (GDB) 7.6.1
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.htm
l>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "mingw32".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
(gdb) 
1\%%-  *GNU Emacs*    All L5     (Fundamental)
1\---  *input/output of *   All L1     (Inferior I/O:run)
1\**-  *gud*          All L11    (Debugger:run [ready])
```

Alternatively you can use *M-x gdb-many-windows* for an alternative split of windows. The functions will be the same, I liked the proportions of this one better. Note that this script deactivates *gdb-many-windows*. If you want it back just delete the section in your .emacs file. If you want to make *gdb-many-windows* persistent use *M-x customize*. This opens up the Emacs config options which you can search for GDB.

The left upper frame shows the Welcome screen. You can see the buffer name "*GNU Emacs*" in the status line at the bottom. Program code will be shown in the frame.

The left lower frame shows input and output of the debugged program as the buffer name *input/output of* indicates. In our case there won't be much to see here as the exe usually doesn't print anything.

Going to the right side and starting at the top you see the buffer *breakpoints of* with an overview of all breakpoints currently set.

Next is the buffer "stack frames of*. This shows the order of function calls with the current function at the top.

Next come the *locals of*, i.e. the local variables of the current scope.

Last is the *gud*, the so-called Grand Unified Debugger. The in- and output here is precisely what you get when starting gdb from the command prompt but you can use the editor's capabilities like scrolling back, copying, re-using a previous command and much more.

As you have to go through the frames once in a while always using the mouse and clicking can be tedious. Add the code in Annex 2 to your .emacs file to use *C-<left arrow>* and *C-<right arrow>* to cycle through the frames.

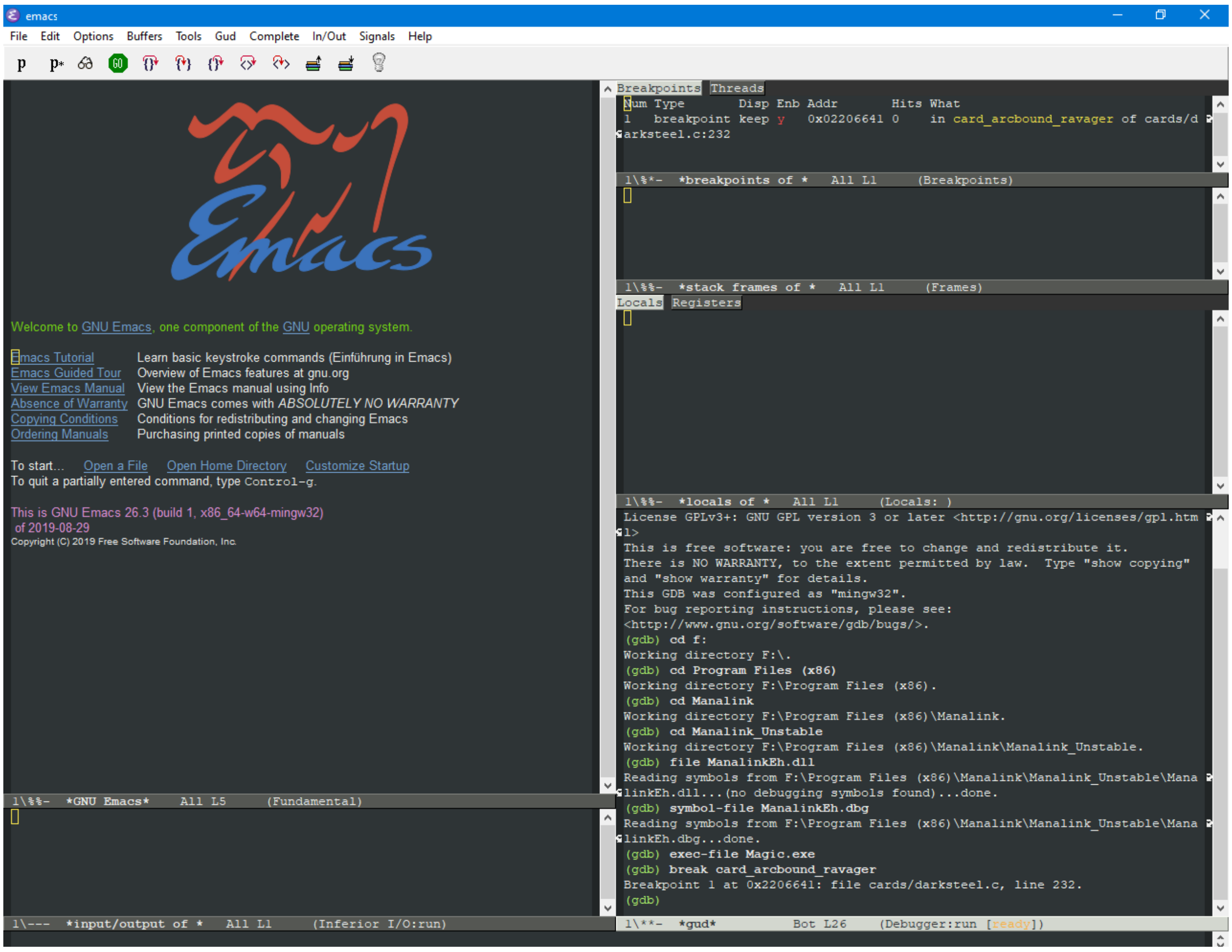
A sample debug session with Emacs

Let’s run the same example as in “Debugging the DLL with gdb”. Say your exe resides in

F:\Program files (x86)\Manalink\Manalink_Unstable

Then enter the following commands the gud frame:

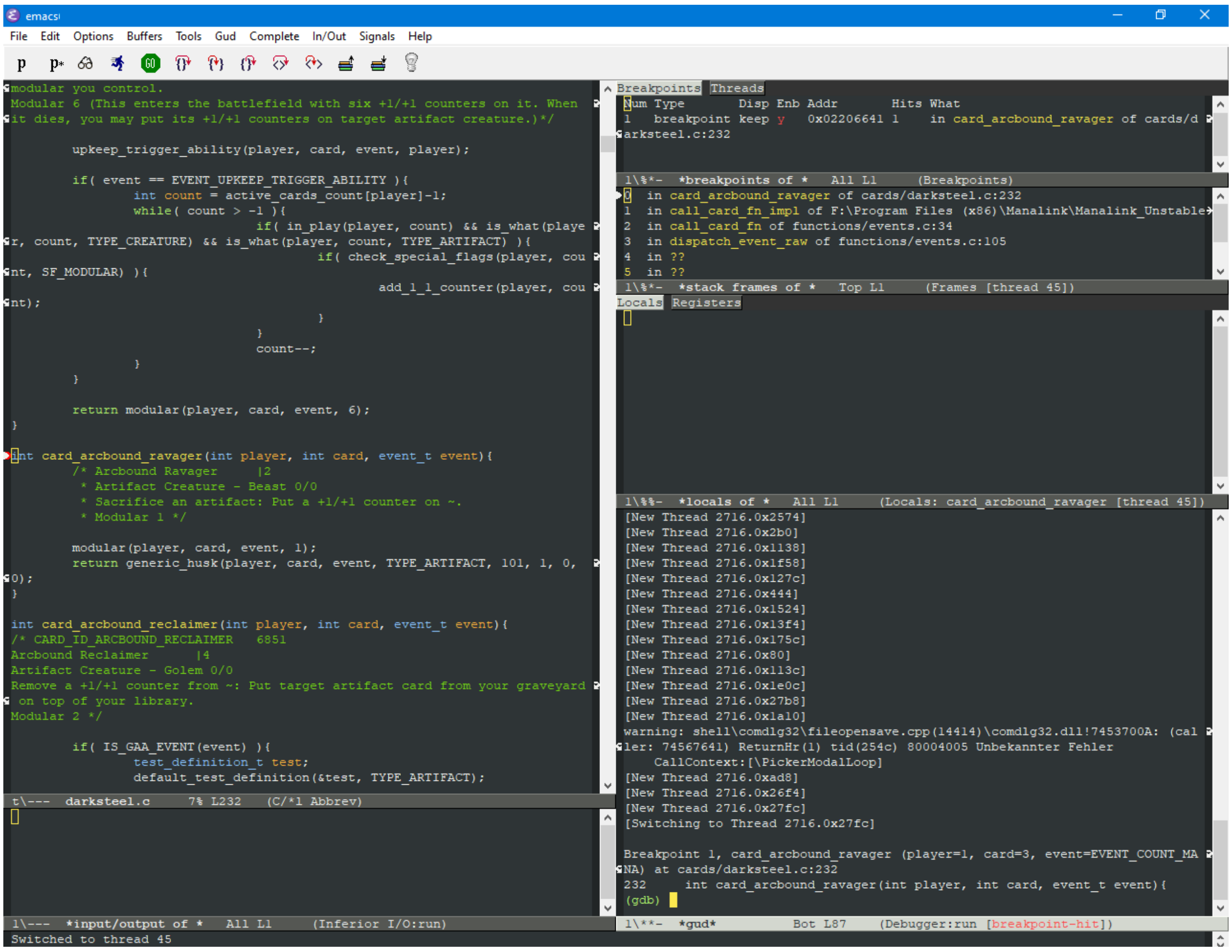
- *cd f:*
- *cd Program files (x86)*
- *cd Manalink*
- *cd Manalink_Unstable* or alternatively *cd Manalink /Manalink_Unstable* for the last two (note the use of „/“ to separate folders as per Unix convention)
- *file ManalinkEh.dll*
- *symbol-file ManalinkEh.dbg*
- *exec-file Magic.exe*
- *break card_arcbound_ravager*
- Your output should look like this:


emacs
File Edit Options Buffers Tools Gud Complete In/Out Signals Help

```
Welcome to GNU Emacs, one component of the GNU operating system.

Emacs Tutorial       Learn basic keystroke commands (Einführung in Emacs)
Emacs Guided Tour    Overview of Emacs features at gnu.org
View Emacs Manual    View the Emacs manual using Info
Absence of Warranty  GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions   Conditions for redistributing and changing Emacs
Ordering Manuals     Purchasing printed copies of manuals

To start...   Open a File   Open Home Directory   Customize Startup
To quit a partially entered command, type Control-g.

This is GNU Emacs 26.3 (build 1, x86_64-w64-mingw32)
 of 2019-08-29
Copyright (C) 2019 Free Software Foundation, Inc.

1\%%-  *GNU Emacs*    All L5     (Fundamental)

1\---  *input/output of *   All L1     (Inferior I/O:run)
```

```
Breakpoints Threads
Num Type        Disp Enb Addr       Hits What
1   breakpoint keep y   0x02206641 0    in card_arcbound_ravager of cards/d
arksteel.c:232

1\%*-  *breakpoints of *   All L1     (Breakpoints)

1\%%-  *stack frames of *   All L1     (Frames)
Locals Registers

1\%%-  *locals of *   All L1     (Locals: )
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.htm
l>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "mingw32".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
(gdb) cd f:
Working directory F:\.
(gdb) cd Program Files (x86)
Working directory F:\Program Files (x86).
(gdb) cd Manalink
Working directory F:\Program Files (x86)\Manalink.
(gdb) cd Manalink_Unstable
Working directory F:\Program Files (x86)\Manalink\Manalink_Unstable.
(gdb) file ManalinkEh.dll
Reading symbols from F:\Program Files (x86)\Manalink\Manalink_Unstable\Mana
linkEh.dll...(no debugging symbols found)...done.
(gdb) symbol-file ManalinkEh.dbg
Reading symbols from F:\Program Files (x86)\Manalink\Manalink_Unstable\Mana
linkEh.dbg...done.
(gdb) exec-file Magic.exe
(gdb) break card_arcbound_ravager
Breakpoint 1 at 0x2206641: file cards/darksteel.c, line 232.
(gdb)
1\**-  *gud*          Bot L26     (Debugger:run [ready])
```

The breakpoint we just set is shown in the upper left frame. Now we can run the exe and wait until an Arcbound Ravager shows up somewhere.

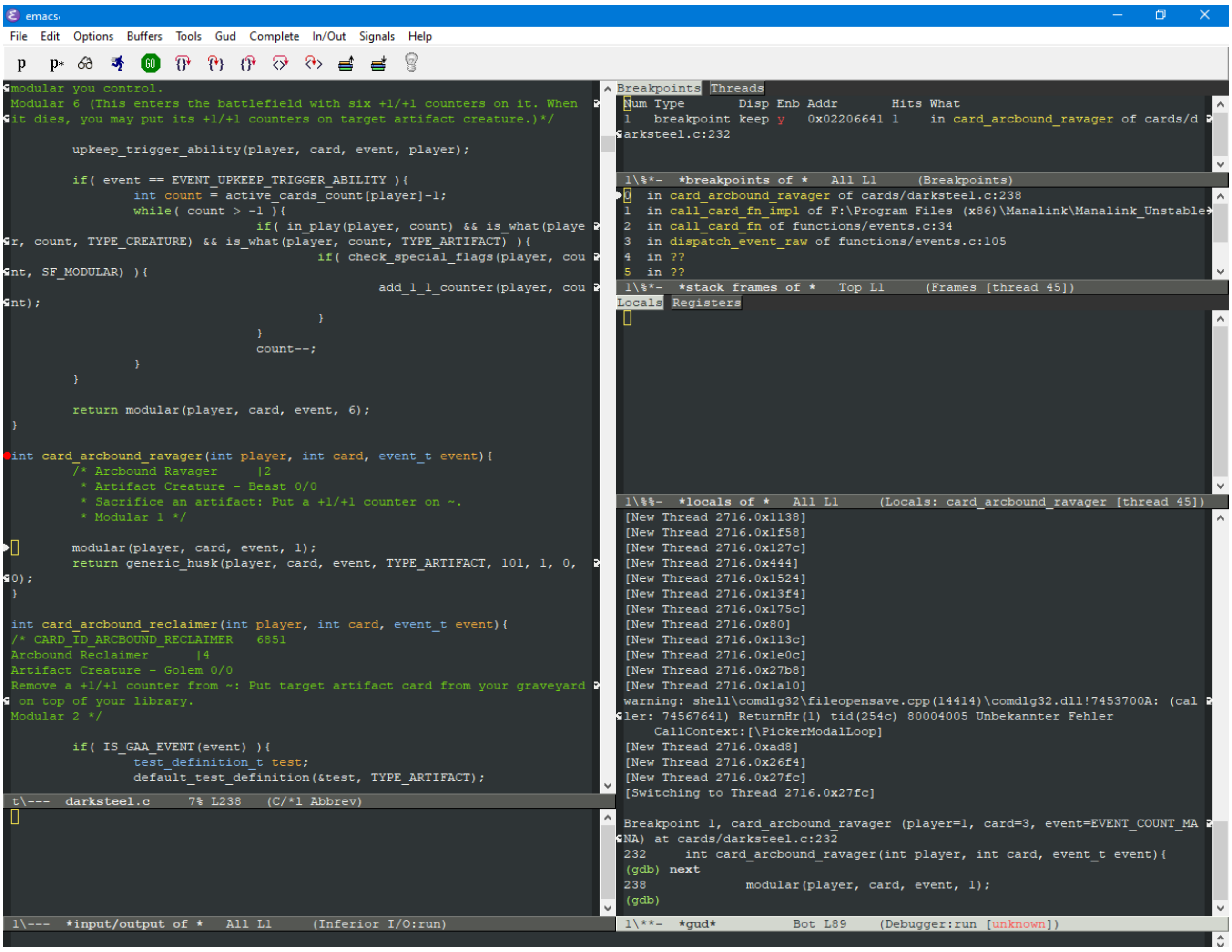

emacs
File Edit Options Buffers Tools Gud Complete In/Out Signals Help

```
modular you control.
Modular 6 (This enters the battlefield with six +1/+1 counters on it. When
it dies, you may put its +1/+1 counters on target artifact creature.)*/

        upkeep_trigger_ability(player, card, event, player);

        if( event == EVENT_UPKEEP_TRIGGER_ABILITY ){
                int count = active_cards_count[player]-1;
                while( count > -1 ){
                                if( in_play(player, count) && is_what(playe
r, count, TYPE_CREATURE) && is_what(player, count, TYPE_ARTIFACT) ){
                                        if( check_special_flags(player, cou
nt, SF_MODULAR) ){
                                                add_1_1_counter(player, cou
nt);
                                        }
                                }
                                count--;
                }
        }

        return modular(player, card, event, 6);
}

int card_arcbound_ravager(int player, int card, event_t event){
        /* Arcbound Ravager      |2
         * Artifact Creature - Beast 0/0
         * Sacrifice an artifact: Put a +1/+1 counter on ~.
         * Modular 1 */

        modular(player, card, event, 1);
        return generic_husk(player, card, event, TYPE_ARTIFACT, 101, 1, 0,
0);
}

int card_arcbound_reclaimer(int player, int card, event_t event){
/* CARD_ID_ARCBOUND_RECLAIMER   6851
Arcbound Reclaimer       |4
Artifact Creature - Golem 0/0
Remove a +1/+1 counter from ~: Put target artifact card from your graveyard
 on top of your library.
Modular 2 */

        if( IS_GAA_EVENT(event) ){
                test_definition_t test;
                default_test_definition(&test, TYPE_ARTIFACT);
t\---  darksteel.c      7% L232    (C/*l Abbrev)

1\---  *input/output of *   All L1      (Inferior I/O:run)
Switched to thread 45
```

```
Breakpoints  Threads
Num Type         Disp Enb Addr         Hits What
1   breakpoint keep y   0x02206641 1     in card_arcbound_ravager of cards/d
arksteel.c:232

1\%*-  *breakpoints of *   All L1      (Breakpoints)
0  in card_arcbound_ravager of cards/darksteel.c:232
1  in call_card_fn_impl of F:\Program Files (x86)\Manalink\Manalink_Unstable
2  in call_card_fn of functions/events.c:34
3  in dispatch_event_raw of functions/events.c:105
4  in ??
5  in ??
1\%*-  *stack frames of *   Top L1      (Frames [thread 45])
Locals  Registers

1\%%-  *locals of *   All L1      (Locals: card_arcbound_ravager [thread 45])
[New Thread 2716.0x2574]
[New Thread 2716.0x2b0]
[New Thread 2716.0x1138]
[New Thread 2716.0x1f58]
[New Thread 2716.0x127c]
[New Thread 2716.0x444]
[New Thread 2716.0x1524]
[New Thread 2716.0x13f4]
[New Thread 2716.0x175c]
[New Thread 2716.0x80]
[New Thread 2716.0x113c]
[New Thread 2716.0x1e0c]
[New Thread 2716.0x27b8]
[New Thread 2716.0x1a10]
warning: shell\comdlg32\fileopensave.cpp(14414)\comdlg32.dll!7453700A: (cal
ler: 74567641) ReturnHr(1) tid(254c) 80004005 Unbekannter Fehler
    CallContext:[\PickerModalLoop]
[New Thread 2716.0xad8]
[New Thread 2716.0x26f4]
[New Thread 2716.0x27fc]
[Switching to Thread 2716.0x27fc]

Breakpoint 1, card_arcbound_ravager (player=1, card=3, event=EVENT_COUNT_MA
NA) at cards/darksteel.c:232
232     int card_arcbound_ravager(int player, int card, event_t event){
(gdb) 
1\**-  *gud*          Bot L87     (Debugger:run [breakpoint-hit])
```

In the upper left frame you see the code which was loaded automatically. There is a little red dot indicating the breakpoint which is a bit hard to see right now. The stack frame buffer tells us that this function has been called by call_card_fn_impl, which in turn has been called by call_card_fn and so on until we end up in unknown territory, probably the exe.

Lets move to the next line using the *next* command inside the gud or by clicking on the appropriate symbol under the menu bar.


emacs
File Edit Options Buffers Tools Gud Complete In/Out Signals Help

```
modular you control.
Modular 6 (This enters the battlefield with six +1/+1 counters on it. When
it dies, you may put its +1/+1 counters on target artifact creature.)*/

        upkeep_trigger_ability(player, card, event, player);

        if( event == EVENT_UPKEEP_TRIGGER_ABILITY ){
                int count = active_cards_count[player]-1;
                while( count > -1 ){
                                if( in_play(player, count) && is_what(playe
r, count, TYPE_CREATURE) && is_what(player, count, TYPE_ARTIFACT) ){
                                        if( check_special_flags(player, cou
nt, SF_MODULAR) ){
                                                add_1_1_counter(player, cou
nt);
                                        }
                                }
                                count--;
                }
        }

        return modular(player, card, event, 6);
}

int card_arcbound_ravager(int player, int card, event_t event){
        /* Arcbound Ravager     |2
         * Artifact Creature - Beast 0/0
         * Sacrifice an artifact: Put a +1/+1 counter on ~.
         * Modular 1 */

        modular(player, card, event, 1);
        return generic_husk(player, card, event, TYPE_ARTIFACT, 101, 1, 0,
0);
}

int card_arcbound_reclaimer(int player, int card, event_t event){
/* CARD_ID_ARCBOUND_RECLAIMER   6851
Arcbound Reclaimer      |4
Artifact Creature - Golem 0/0
Remove a +1/+1 counter from ~: Put target artifact card from your graveyard
 on top of your library.
Modular 2 */

        if( IS_GAA_EVENT(event) ){
                test_definition_t test;
                default_test_definition(&test, TYPE_ARTIFACT);
t\---  darksteel.c     7% L238    (C/*l Abbrev)

1\---  *input/output of *   All L1      (Inferior I/O:run)
```

```
Breakpoints Threads
Num Type       Disp Enb Addr       Hits What
1   breakpoint keep y   0x02206641 1    in card_arcbound_ravager of cards/d
arksteel.c:232

1\%*-  *breakpoints of *   All L1      (Breakpoints)
0  in card_arcbound_ravager of cards/darksteel.c:238
1  in call_card_fn_impl of F:\Program Files (x86)\Manalink\Manalink_Unstable
2  in call_card_fn of functions/events.c:34
3  in dispatch_event_raw of functions/events.c:105
4  in ??
5  in ??
1\%*-  *stack frames of *   Top L1      (Frames [thread 45])
Locals Registers

1\%%-  *locals of *   All L1      (Locals: card_arcbound_ravager [thread 45])
[New Thread 2716.0x1138]
[New Thread 2716.0x1f58]
[New Thread 2716.0x127c]
[New Thread 2716.0x444]
[New Thread 2716.0x1524]
[New Thread 2716.0x13f4]
[New Thread 2716.0x175c]
[New Thread 2716.0x80]
[New Thread 2716.0x113c]
[New Thread 2716.0x1e0c]
[New Thread 2716.0x27b8]
[New Thread 2716.0x1a10]
warning: shell\comdlg32\fileopensave.cpp(14414)\comdlg32.dll!7453700A: (cal
ler: 74567641) ReturnHr(1) tid(254c) 80004005 Unbekannter Fehler
    CallContext:[\PickerModalLoop]
[New Thread 2716.0xad8]
[New Thread 2716.0x26f4]
[New Thread 2716.0x27fc]
[Switching to Thread 2716.0x27fc]

Breakpoint 1, card_arcbound_ravager (player=1, card=3, event=EVENT_COUNT_MA
NA) at cards/darksteel.c:232
232     int card_arcbound_ravager(int player, int card, event_t event){
(gdb) next
238             modular(player, card, event, 1);
(gdb)
1\**-  *gud*          Bot L89    (Debugger:run [unknown])
```

Much better. The little arrow indicates which line of code will be executed next. From here on you can do the same things as already outlined in “Debugging the DLL with gdb”. Let’s stop here with the *kill* command and either quit gdb with the *quit* command or Emacs either through File-Quit or *C-x C-c* (in this case Emacs recognizes that the gdb process is still running and asks you whether you really want to exit).

Some tips

Look at the menu bar and the icon bar. Depending on the context both might change and provide you commands that fit the current frame.

Use *C-h C-h* for the built-in help function.

If you want to quit a command press *C-g*.

To create a new file use find or shorter *C-x C-f*.

To save a file use save or shorter *C-x C-s*, to save as *C-x C-w*.

Searching works easiest through incremental search (search as you type) with *C-s* for forward incremental search and *C-r* for backward incremental search. If you found the expression, but not at the right place (e.g. when searching for “card”) press *C-s* or *C-r* again to move to the next occurrence. Pressing the Enter key after *C-s* or *C-r* brings you to the non-incremental search where you enter the search string, then press Enter to search the next occurrence.

Call Dired (C-x d) for an explorer like interface to browse files, directories and much more like finding files, comparing directories and much more.

If you mark something with the mouse (press left button and move the mouse pointer) the marked section gets copied into a buffer. Use the middle mouse button for pasting (as an alternative to *C-c C-v* as this uses a different buffer)

Need a break? Try *M-x tetris*.

Keyboard shortcuts for the debugger

Here you find some frequently used keyboard shortcuts for quick reference.

These commands are available both in the GUD interaction buffer (bottom of the right window half) and globally, but with different key bindings. The keys starting with *C-c* are available only in the GUD interaction buffer, while those starting with *C-x C-a* are available globally.

- *C-x C-a C-b*: Set a breakpoint on the source line that point is on.
- *C-c C-s* or *C-x C-a C-s*: Execute the next single line of code (step). If the line contains a function call, execution stops after entering the called function.
- *C-c C-n* or *C-x C-a C-n*: Execute the next single line of code, stepping across function calls without stopping inside the functions (next).
- *C-c C-p* or *C-x C-a C-p*: Evaluate the expression at point (print). If Emacs does not print the exact expression that you want, mark it as a region first.
- *C-c C-r* or *C-x C-a C-r*: Continue execution without specifying any stopping point. The program will run until it hits a breakpoint, terminates, or gets a signal that the debugger is checking for (cont).
- *C-c C-d* or *C-x C-a C-d*: Delete the breakpoint(s) on the current source line. If you use this command in the GUD interaction buffer, it applies to the line where the program last stopped.

Finally, some documentation...

Emacs is one hell of an editor and it is easy to get lost. The commands above should get you started, but to use it to the full extent possible you need to look at the documentation. Here are some cheat sheets, a shorter tutorial and the full-blown documentation:

- Emacs cheat sheet: http://www.rgrjr.com/emacs/emacs_cheat.html
- Emacs tutorial: https://www.emacswiki.org/emacs/EmacsTutorial
- Emacs manual: https://www.gnu.org/software/emacs/manual/html_node/emacs/index.html
- Gdb graphical interface (within the Emacs manual): https://www.gnu.org/software/emacs/manual/html_node/emacs/GDB-Graphical-Interface.html#GDB-Graphical-Interface
- Using GDB under GNU Emacs: https://sourceware.org/gdb/onlinedocs/gdb/Emacs.html#Emacs
- GUD commands (within the Emacs manual): https://www.gnu.org/software/emacs/manual/html_node/emacs/Commands-of-GUD.html
- Emacs amusements(within the Emacs manual): https://www.gnu.org/software/emacs/manual/html_node/emacs/Amusements.html

Open issues:

I don’t know how to remove “wish.exe” from the command line when calling gdb the first time.

Annex 1 (a nice debugging layout):

Code for the .emacs file:

```
;; Fancy gdb layout
;; from https://stackoverflow.com/questions/3860028/customizing-emacs-gdb

(setq gdb-many-windows nil)

(defun set-gdb-layout(&optional c-buffer)
  (if (not c-buffer)
      (setq c-buffer (window-buffer (selected-window)))) ;; save current buffer

  ;; from http://stackoverflow.com/q/39762833/846686
  (set-window-dedicated-p (selected-window) nil) ;; unset dedicate state if needed
  (switch-to-buffer gud-comint-buffer)
  (delete-other-windows) ;; clean all

  (let* (
        (w-source (selected-window)) ;; left top
        (w-gdb (split-window w-source nil 'right)) ;; right bottom
        (w-locals (split-window w-gdb nil 'above)) ;; right middle bottom
        (w-stack (split-window w-locals nil 'above)) ;; right middle top
        (w-breakpoints (split-window w-stack nil 'above)) ;; right top
        (w-io (split-window w-source (floor(* 0.9 (window-body-height)))
                    'below)) ;; left bottom
        )
    (set-window-buffer w-io (gdb-get-buffer-create 'gdb-inferior-io))
    (set-window-dedicated-p w-io t)
    (set-window-buffer w-breakpoints (gdb-get-buffer-create 'gdb-breakpoints-buffer))
    (set-window-dedicated-p w-breakpoints t)
    (set-window-buffer w-locals (gdb-get-buffer-create 'gdb-locals-buffer))
    (set-window-dedicated-p w-locals t)
    (set-window-buffer w-stack (gdb-get-buffer-create 'gdb-stack-buffer))
    (set-window-dedicated-p w-stack t)

    (set-window-buffer w-gdb gud-comint-buffer)

    (select-window w-source)
    (set-window-buffer w-source c-buffer)
    ))
(defadvice gdb (around args activate)
  "Change the way to gdb works."
  (setq global-config-editing (current-window-configuration)) ;; to restore: (set-window-
configuration c-editing)
```

```
  (let (
        (c-buffer (window-buffer (selected-window))) ;; save current buffer
        )
    ad-do-it
    (set-gdb-layout c-buffer))
  )
(defadvice gdb-reset (around args activate)
  "Change the way to gdb exit."
  ad-do-it
  (set-window-configuration global-config-editing))
```

Annex 2 (cycling through the frames):

Code for the .emacs file:

```
;; Set c-<left> and c-<right> to cycle through gud frames
(global-set-key (kbd "C-<right>") 'next-multiframe-window)
(global-set-key (kbd "C-<left>") 'previous-multiframe-window)
```