Liquid War 6

A unique multiplayer wargame
Edition 0.0.6beta, for Liquid War 6 Version 0.0.6beta
10 January 2009

by Christian Mauduit <ufoot@ufoot.org>

Liquid War 6, a unique multiplayer wargame.

Table of Contents

1 Introduction

Read this chapter to discover Liquid War 6.

1.1 In a nutshell

Liquid War 6 is a unique multiplayer wargame. Your army is a blob of liquid and you have to try and eat your opponents. Rules are very simple yet original, they have been invented by Thomas Colcombet. It is possible to play alone against the computer but the game is really designed to be played with friends, on a single computer, on a LAN, or on Internet.

An older version, Liquid War 5, is available, but is not part of the GNU Project. Only Liquid War 6 is part of the GNU Project, it is a complete rewrite.

Yor more information, you can read the Wikipedia article about Liquid War.

1.2 Project status

As of today, the game is in beta state. It can be installed, and you can toy arround with, but it's far from being complete.

What works:

- The whole framework is here, some functions are not implemented yet, but the bases are set up, and they are believed solid. The game is very modular, it is designed so that graphics, sound, network and bot backends can be hacked at will. It has a complete self-test suite, and many debugging built-in tools. This is not a quick hack.
- Documentation. Yes, you're reading it.
- Version 0.0.6beta is playable. Local game between humans (up to 4 players) is possible. Two bots are implemented, named random and stupid. No great players but well, they move the cursor.
- Liquid War 6 already has some features which are nowhere to be found in Liquid War 5, such as multiple layers. It can be worth the upgrade.
- Maps. A number of interesting maps have already been designed (thanks to Kasper Hviid).
- The game has been ported to Microsoft Windows 32-bit proprietary platform. It still runs fine under GNU/Linux, of course.

In the near future:

- Network play. Top-level priority.
- Fix bugs ;) The current engine is somewhat buggy, fighters might loose the cursor, it clearly needs polishing.

In the long run:

- Write new graphical backends so that the game does not require Mesa or any OpenGL-like subsystem. The idea is to get rid of the 3D-accelerator dependency.
- Implement all the fancy 3D features, make it possible to play Liquid War 6 on a Moebius ring.
- Use the cool features of CSound to provide dynamic, contextualized sounds & musics.
- Optimize the bot algorithm, which is probably a complex AI problem.

1.3 How you can help

1.3.1 Help GNU

Please remember that development of Liquid War 6 is a volunteer effort, and you can also contribute to its development. For information about contributing to the GNU Project, please read How to help GNU.

1.3.2 Todo list

Here's a short list of todo items. It is probably too early to start hacking the core engine itself, for it is still under heavy development, might undergo major rewrites, and it's hard for documentation to keep up with the reality of the code. However, there are still many things to do.

- Try the game. Play. Test. Send bug reports. Without bug reports, bugs don't get fixed.
- Write maps. Obviously, this is something which can perfectly be delegated. Experience shows user-contributed maps are, on average, better than maps conceived by the author...
- Port the game to other platforms, for instance Mac OS/X.
- Translate texts. Liquid War 6 uses GNU gettext, so all messages can be translated.
- ...any help is welcome.

Feel free to join the mailing-lists or contact Christian Mauduit if you are interested.

2 User's manual

The Liquid War 6 user's manual hopefully contains any usefull information to install the program and play the game. If you just want to enjoy Liquid War 6 without diving into map creation and programming, this is just for you.

2.1 Mailing lists

2.1.1 General discussion

The main discussion list is <help-liquidwar6@gnu.org>, and is used to discuss all aspects of Liquid War 6, including installation, development, game strategies, and whatever subject players and hackers might want to talk about, provided it is Liquid War 6 related. If you don't know on which list to subscribe, this is the one.

To subscribe to it, please send an empty mail with a Subject: header line of just "subscribe" to the `-request` list, that is <help-liquidwar6-request@gnu.org>.

You can also subscribe to the list using the Mailman web interface for help-liquidwar6 and consult help-liquidwar6 archives.

2.1.2 Announcements

Announcements about LiquidWar 6 are made on <info-liquidwar6@gnu.org>. Subscribe to it to be informed of major releases, and other significant news.

To subscribe to it, please send an empty mail with a Subject: header line of just "subscribe" to the `-request` list, that is <info-liquidwar6-request@gnu.org>.

You can also subscribe to the list using the Mailman web interface for info-liquidwar6 and consult info-liquidwar6 archives.

Please also consider reading the latest news on Savannah.

2.1.3 Bugs

There is also a special list used for reporting bugs, <bug-liquidwar6@gnu.org>. Please try and describe the bug as precisely as possible. The more accurate the description, the more chances it will get to be fixed.

While this is the standard GNU way of reporting bugs, modern SPAM standards make it very hard to filter real bug reports from junk on this list. It is more convenient to report bugs on Savannah using a web interface.

Please take a look at the bug list before submitting new bugs.

2.2 Getting the game

2.2.1 Download source

Liquid War 6 can be found on `http://download.savannah.gnu.org/releases/liquidwar6/` and `http://www.ufoot.org/download/liquidwar/v6/`.

Downloading the latest file from this place, and compile it yourself on your computer with a classical `./configure && make && make install` is the recommended way to install Liquid War 6.

2.2.2 Download binaries

Some binary packages are available. As of today, only GNU/Linux based systems are supported, through Debian `.deb` and Red Hat `RPM` packages.

Using these files might save you time installing the game, but installing from source is still the safest and best supported way to install the game, as it is still in beta stage. Binary are also not necessarily available for the latest, most up to date versions of the game.

The list of all the available downloads is accessible on `http://www.ufoot.org/liquidwar/v6/download`.

Check out the MD5 checksums and GnuPG signatures to verify the integrity and authenticity of the files you download.

2.2.3 Work in progress

Latest work in progress versions can be obtained with GNU Arch. Here's a typical set of commands which will fetch the latest version:

```
tla register-archive http://arch.sv.gnu.org/archives/liquidwar6
tla get -A liquidwar6@sv.gnu.org liquidwar6--beta
```

Alternatively, you can directly download GNU Arch patches, and an Archzoom server allows you to browse the source interactively.

2.3 Installation

This section covers installation from source. Other ways of installing the program are not described here.

2.3.1 Requirements

All these libraries are mandatory to compile the game. Liquid War 6 won't compile, let alone run, without them. Some of them could probably be replaced by equivalent tools, but this would certainly require a programming effort and some changes in Liquid War 6 source code.

- GNU Make. Liquid War 6 might and certainly does use GNU Make extensions.
- GNU C library. Sounds obvious, but you need a standard C library. It happens that glibc has some rather usefull extensions (yes, as of 2006, some vendors continue to offer C libraries without `snprintf`...) and Liquid War 6 might use them. In a general manner, Liquid War 6 is part of and designed for GNU. You might however manage to compile it with limited libc support, this is the case with mingw32 for instance but, do it at your own risk.
- Perl. Some Makefile commands require Perl. You don't need any Perl devel packages, and you can probably use any Perl 5.x version, since no fancy recent feature of Perl is used. Just plain Perl.
- Guile. Possibly the most required library, since Liquid War 6 is a scheme program which uses a set of functions coded in standard C. You need at least Guile 1.8.
- GNU MP. GMP is a free library for arbitrary precision arithmetic, required by Guile.
- ltdl. This library, which comes with libtool, provides a portable alternative to `dlopen` and `dlclose`. Check that you have a `/usr/include/ltdl.h` file, or install the corresponding package.

- zlib. Required by other libraries, but can also be used directly by Liquid War 6 to compress network messages for instance.
- expat. Used to read and write XML files, which contain constants and configuration data.
- libpng. Liquid War 6 uses libpng to read levels (maps), not to speak of other optional libraries (SDL and the rest) who need it themselves.
- libjpeg. Maps can also be provided as jpeg files, so libjpeg is required as well.
- SQLite 3. Used to handle the list of available servers.

2.3.2 Optional libraries

While all these libraries are theorically optional (the game will successfully compile without them), you'll obviously need, for instance, one graphics backend. Otherwise, you'll simply have no display. This is not acceptable. As of today, one can reasonnably consider all SDL-related libraries are required. The rest is truely optional.

- ncurses. Required by readline, needs to be there otherwise readline might not be detected properly on some systems.
- GNU readline. Used to handle input on the console. Console is not absolutely mandatory, but it's a must-have if you want to hack the game. Console unavailable does not mean you won't get anything on stdout but, the interactive script shell just won't work.
- Mesa. This library provides an API similar to OpenGL and enables 2-D and 3-D drawing.
- SDL. SDL is used to set up a working OpenGL environnement, and handle input (mouse and keyboard).
- SDL_image. This SDL extension is used to read textures and other graphics from disk.
- FreeType 2. This library is required by SDL_ttf, to draw fonts.
- SDL_ttf. This SDL extension is used to draw fonts. It is UTF-8 enabled.
- libcsound. While this tool is not used yet, it is meant to be the final sound backend, as CSounds offers great power to the composer, enabling truely dynamically generated sound & music.
- SDL_mixer. This SDL extension is used to allow dynamic mixing of sounds, and it also provides a builtin `OGG/Vorbis` file renderer.
- libcURL. Used to handle HTTP requests, the idea being not to re-invent the wheel but use a robust standards-compliant generic library.

2.3.3 Compiling

Liquid War 6 uses GNU Automake, Autoconf and GNU Libtool. Once the requirements and optional libraries are installed on your system, run:

```
./configure
make
make install
```

Liquid War 6 supports the standard `./configure --prefix=/my/path` option (in fact, it supports much more than that) so you can install the game in any directory. You do not need to be `root` to install Liquid War 6.

2.4 Extra maps

2.4.1 The extra maps package

The main package contains some maps so that you can try out the game. Still, an additionnal package, called `extra-maps` or `liquidwar6-extra-maps` is available, containing more maps. It really does contain many of them, including most Liquid War 3 and Liquid War 5 legacy maps, plus new Liquid War 6 maps.

2.4.2 Install extra maps on GNU/Linux and POSIX systems

On GNU/Linux systems (and possibly any POSIX unixish system) running:

```
./configure
make
make install
```

will install the extra maps on your system automatically, they will then be available in the `extra/` sub-directory when browsing maps.

The `./configure` script has a `--enable-liquidwar6` switch which will try and find automatically if there's an existing `liquidwar6` binary in the path. If there's such a binary, it will run it and ask for its `map-path` and use this value automatically.

2.4.3 Raw install of extra maps (all-platforms)

Another solution, which works on all platforms including MS-Windows but also works on GNU/Linux, is to simply unpack the `extra-maps` package (unzip or untar) in your custom map directory, or in the system map directory. There's nothing else to do to install these maps but simply put them on your hard drive in the right directory.

Typically on an MS-Windows system, you would unpack the extra maps in `C:\Program Files\Liquid War 6\map\` (system directory) and on a GNU/Linux or POSIX system you would unpack them in `$HOME/.liquidwar6/map/` (user directory).

Next time you run the game, the maps should be browsable.

If you can't see them, run `liquidwar6 --audit` and check that the place where you unpacked the files is actually searched by the binary.

2.5 Troubleshooting

2.5.1 Compilation problems

A quick survival guide:

- Check that you have all dependencies installed. Also check their version number. Double-check that you have devel packages installed, not only run-time binaries.
- Read carefully the output of `./configure`. Running `./configure > configure.log 2> configure.err` does help.
- Editing `/etc/ld.so.conf` and running `ldconfig` as `root` can help if some dependencies are installed in exotic places.
- Check the values of the environment variables `CFLAGS`, `LDFLAGS` and `LD_LIBRARY_PATH`.

- Try `./configure --enable-allinone`, this will disable some fancy but somewhat complicated dynamic `.so` file support, it can help if shared libraries are handled differently on your system than on a plain GNU/Linux box.

If none of these help, consider reporting a bug, or search the mailing-lists for help.

2.5.2 Check installation

Here's a check-list to ensure that your installation is correct:

- What was the output of `make install`? `make check`?
- Is the `liquidwar6` binary in your `PATH` environment variable? It might be in `/usr/games`.
- Run `liquidwar6 --pedigree`. Look at the output. Check the compilation date & time, the version number.
- Run `liquidwar6 --audit`. What do these paths look like? Are they absolute paths? Do they exist? What's there? Normally, once the game is installed, all of them should exist, and be populated with sub-directories and files.
- Run `liquidwar6 --modules`, to know which modules where compiled. You need at least one graphical module, for instance `mod-gl`, else the game won't run.
- Run `liquidwar6 --target`, this displays informations about the target system the binary has been built for.

2.5.3 Problems running the game

Now, game looks correctly installed, but you have problems running it.

- Run the game from a terminal, not from a Gnome or KDE launcher, you need to see the console output.
- In the `$HOME/.liquidwar6/` directory, you'll find two files, `log.csv` and `dump.txt`. They might contain valuable information, read them.
- Run `liquidwar6 --defaults`. This will reset all options to defaults. You might need to run this when upgrading from a version to another, since some options might appear, disappear, or defaults values can change.
- Run `liquidwar6 --test`. This should run a complete test suite, many functions in the game will be tested automatically, and errors reported.
- Run `liquidwar6 --show-script-file`. Are you really running the right code?
- Game segfaults: try `make uninstall && make clean && make && make install`. Many problems can come from using a wrong shared module, especially with beta versions.
- Game (still) segfaults: try `gdb liquidwar6`. Type `run` and watch output.
- The dynamic library loader can sometimes have problemes, and does not always report explicit messages on `stdout` or `stderr`. You can change this by modifying some environment variables: `export LD_DEBUG=all`. This is very verbose but does help finding bugs.
- Consider compiling the game using `./configure --enable-valgrind` and then run it using Valgrind.
- Try `find / -type d -a -name "liquidwar6*" 2> /dev/null` to ensure you don't have an old version of Liquid War 6 somewhere else...

2.6 Quick start

2.6.1 Quick start

Once the game is installed, run it, click on `Quick start` with the mouse, and control the red `'a'` cursor with the mouse, or keyboard, both work. Try and surround the green team, it's a stupid bot, you should win ;)

You army is formed by all the red pixels on the screen, they should try and rejoin the cursor (the blinking `'a'` letter) using the shortest path. When red and green meet, they fight. Try it, toy arround.

The `Quick start` button will always make you play red against a green stupid bot, whatever other options you have set up.

Todo...

2.7 Strategy tips

2.8 User interface

Todo...

2.9 Network games

Not implemented yet.

2.10 Graphics

Todo...

2.11 Sound & music

Todo...

2.12 Config file

The config file is a simple XML file. It uses XML only to benefit standard parsing tools, but it's not a structured XML file, in the sense that the tree is so simple that all items are at the same level. It is just a simple key-value binding.

This file is in `$HOME/.liquidwar6/config.xml`, you're free to edit it manually, but all parameters are changeable with command line options. The program will overwrite this file each time it exits, so if you put comments in it, they will disappear. The advantage of this is that if you mispell something, or if for some reason the game does not understand a value, then when rewriting the file, it will show you it just did not get it.

The file embeds the documentation for all its entries, it is therefore rather verbose. The documentation is the same you will find online or by quering the game with the `--about` option, also the same you would get reading this manual.

2.13 Logs

Liquid War 6 uses `stdout` to output important messages, and `stderr` to log warnings and errors. It will also use syslog if available.

Additionnally, a verbose log is available in `$HOME/.liquidwar6/log.csv`. You can read this using any spreadsheet software capable of reading csv file. It uses the tab (`\t`) character as a separator. It contains valuable informations including version and most default values for the game, and for each line logged, it says where in the code the log function was called. A must-have for debugging.

2.14 Report bugs

There are two ways to report bugs:

- send a mail to <bug-liquidwar6@gnu.org>;
- use the web-based Savannah bug tracker.

The latter (Savannah) is much preferred, because the mailing-list is bloated with spam... It also offers a list of bugs which you should read before submitting a new one.

3 Hacker's guide

This hacker's guide is for anyone who is curious about the game, and wants to know how it works. It covers many aspects from simple map creation to technical program internals. A great effort has been done in Liquid War 6 so that it should be much more hackable than previous versions. Any feedback is welcome.

3.1 Designing levels

3.1.1 Why is level design so important?

As of Liquid War 5, most levels have been contributed by players. While the maintainer of Liquid War 6 has technical knowledge to develop the game, artistic talent and taste might not be his domain of excellence 8-)

Therefore contribution are truely welcomed when they take the form of a new, original, fun and good looking level. It's believed the levels often make the game much more than its engine. This is true for any type of game, and Liquid War is no exception.

So this section is here to help players understand how to hack existing levels, and create new ones, in the hope that 1) they can enjoy their own creations and 2) possibly share their work with others.

Note that this manual might refer to levels and maps: they are just two different names to describe the very same thing. It's an alias.

3.1.2 Format overview

Liquid War 6 stores level information in a plain directory.

There is no such thing as an opaque `.dat` binary file. The name of the level is the name of the directory itself, and its elements are the files contained in it.

Files must follow a precise naming scheme. For instance Liquid War 6 expects a `map.png` file to be present in each map directory.

All image files in a level use the Portable Network Graphics or JPEG format. It is possible that in the long term, Liquid War 6 will be able to handle levels as `.tar.gz` or `.zip` files. In that case these files will only be a compressed image of the actual level directory.

See the `./map/` directory of the source Liquid War 6 distribution to see example of maps.

3.1.3 map.png

This is the only required file in a level.

In fact, the existence of `map.png` makes a directory a level. When checking wether a directory is a correct level, Liquid War 6 simply tests the existence and validity of `map.png`.

This image is a simple black & white area, where white zones are the background, the sea, the places where fighters can move, and black zones are the foreground, the walls, the places where fighters can't go.

This informations can be stored in a 2-color indexed file, or in a grayscaled or even truecolor RGB file, but color information won't be used. Internally, Liquid War 6 will read the color of every point. If it is over 127 on a 0 to 255 scale, it will be considered as background, if it is below 127, it will be considered as foreground.

3.1.4 layer2.png ... layer7.png

Liquid War 6 can handle mutiple layer maps. Think of a pile of maps, one being on top of the other. This allows you to create a volume, the game considers every layer has two axis x and y, and the z axis is to travel through layers. First layer corresponds to z=0, second layer to z=1, and so on.

Here are the files you can use to define layers:

- `map.png` this one is on top, it's always defined (z=0)
- `layer2.png` (z=1)
- `layer3.png` (z=2)
- `layer4.png` (z=3)
- `layer5.png` (z=4)
- `layer6.png` (z=5)
- `layer7.png` (z=6)

A `layerX.png` file should be designed exactly like `map.png`. In fact, `map.png` could simply have been called `layer1.png`.

Up to 6 extra layers can be defined (from `layer2.png` to `layer7.png`). This is a hard-coded limit. It allows you to define 7 different layers, including the top `map.png` layer. Keep in mind this layer system is not real 3D, it's more a "2D and a half" model. Adding layers can considerably slow down the game, so it's wise to try and use as few layers as possible. Technically, 3 layers will allow you to build bridges and tunnels, which is probably the most usefull construction using layers. Fighters can also have difficulties navigating through layers so piling up layers in narrow "vertical" z-axis based tunnels is probably not a great idea.

The `ufoot/concept/pass` map of the `liquidwar6-extra-maps` demonstrates basic layer usage.

3.1.5 texture.png, texture.jpeg and texture-alpha.jpeg

It is possible to define a texture for the map by putting a `texture.png` or `texture.jpeg` file. It does not need to have the same dimensions as the map itself. Indeed, textures can be much more precise than the actual logical map.

There's no theorical limit on how big a texture can be, more precisely, it can be much bigger than any hardware/driver maximum texture size. In practice, a too big texture will waste your video card RAM, and slow everything down. Sizes ranging from 640x480 to 1600x1200 are reasonable texture sizes.

If you don't define this, the `map.png` file will be used as the texture, and also import colors from `style.xml` if defined.

Note that the shape of the texture defines the shape of the map, that is, the ratio with which it will appear on the screen.

The PNG alpha layer will be used for transparency. But to save disk space, it can be convienient to prefer the JPEG format, use `texture.jpeg` instead of `texture.png` and store the alpha layer in a separated file, called `texture-alpha.jpeg`. This avoids handling heavy PNG files, PNG compression not being performant on most textures.

In `texture-alpha.jpeg`, white is considered opaque, black is transparent. Different levels of gray correspond to different levels of opacity.

3.1.6 rules.xml

Whereas `style.xml` is only about the appearance of the map, `options.xml` allows the map designer to change pretty much any parameter.

Ultimately, the player can still ignore these settings and overide them with its own values, but the idea is: most game options are only pertinent in a given context. For instance, on some maps it's interesting to move slowly, on some other it's interesting to move fast. Some maps might be playable packed with fighters everywhere, some other might be much more fun with almost nobody on them.

The approach in Liquid War 5 was to make the options available, but let the player himself find the right settings for the right map. The consequence is that no one ever used all those cryptic options in the advanced options menu, and probably 99% of the players ended up playing with default settings. This is not that bad, but given the fact that changing a few parameters one can totally transform the gameplay, it has decided been that in Liquid War 6, the map designer suggests the right options that matches his map.

This does not prevent the player from toying with options himself, he can still do it.

There's also one important point to note: all these options are technically implemented as integer parameters. We certainly do not want any float here, since, and it is a Liquid War specific behavior, the game must be 100,00% predictable and behave the same on every platform. As there is nothing like exactness when speaking of floats, those are forbidden here. As for strings, we are dealing here with low-level internals, and this section is not about telling a story. They are technical options only. Booleans are implemented with the usual `false = 0` and `true = 1` convention. Note that other config files in Liquid War 6 might rely on floats, strings, and booleans with conventionnal `true` and `false` values, but not this one. `rules.xml` is special.

See Section 4.9 [Map rules.xml], page 40.

3.1.7 hints.xml

This XML file gives hints to the map loader. It will for instance allow the user to modify the resolution on the fly, force a minimum, a maximum surface (resolution) for the map. It can seriously change gameplay, but parameters set here never appear directly in the loaded map, changing them afterwards makes no sense.

See Section 4.10 [Map hints.xml], page 52.

3.1.8 style.xml

This is a simple XML file defining various appearance parameters. It has absolutely no effect on gameplay. These settings can ultimately be overriden by the player, but the idea is that if the map designer thinks this level looks better with this or that option, let him say it in this file.

See Section 4.11 [Map style.xml], page 56.

3.2 Architecture

3.2.1 C + Guile

Technically, Liquid War 6 is a collection of C functions which are exported to Guile. The main binary embeds a Guile interpreter, which will run a Guile script. This script calls the exported C functions, and glues them together.

3.2.2 Internal libraries

The C code is splitted into several internal libraries. This allow independant testing of various game modules.

3.3 About mod-gl

Todo...

3.4 Compilation tips

3.4.1 Advanced ./configure options

In addition to all the common Autoconf switches such as `--prefix`, Liquid War 6 has some custom switches:

- `--enable-optimize`: will turn on optimizations. This will turn on compiler options such as `-fomit-frame-pointer` but also disable some code in the program. Indeed, most of the advanced memory checking in the game - which ensures it does not leak - will be turned of. This will certainly speed up things, however, it's not recommended to turn this on until program is not stable enough so that memory leaks and other problems can be declared 'impossible'. Turn this on if you really have some speed problem, otherwise it's safer to use the full-featured 'slow' version of the game.
- `--enable-allinone`: will stuff all the internal libraries into one big executable. Very convenient for profiling. The major drawback is that you need to have all the optional libraries installed to compile all the optional modules. Another side effect is that with this option there's no more dynamic loading of binary modules, so if your platform has a strange or buggy support for `.so` files, this option can help.
- `--enable-fullstatic`: will build a totally static binary, that is using the `--static` option for `gcc` and the `-all-static` option for `libtool`. Currently broken, this option could in the future allow for building binaries that run pretty much everywhere, without requiring any dependency but a Kernel.
- `--enable-gprof`: will enable profiling informations. This will activate `--enable-allinone`, else you would only track the time spent in functions in the main `liquidwar6` executable, and exclude lots of interesting code contained in dynamic libraries.
- `--enable-gcov`: will enable coverage informations, to use with gcov and lcov. This is for developpers only. It will activate `--enable-allinone`, else there would be some link errors when opening dynamic libraries.
- `--enable-valgrind`: will enable some `CFLAGS` options which are suitable for the use of Valgrind, to track down memory leaks and other common programming errors. Use for debugging only, usually together with `--enable-allinone`.

3.4.2 Microsoft Windows msys/mingw32 port

This section describes how to compile the game from source under Microsoft Windows. Note that players are encouraged to use a free system such as GNU/Linux, which is the platform Liquid War 6 is being hacked on by default. If you encounter problems with this port, you'll probably save time by installing a double-boot with GNU/Linux coexisting with your previous Microsoft Windows install.

Basically, Liquid War 6 requires MinGW. More precisely, it requires MSYS. A standard Cygwin installation won't work, because it is too UNIXish to allow third party libraries like SDL to compile natively. You might argue that SDL is available for Cygwin, but in reality, the Cygwin port of SDL is a MinGW port. Indeed, Cygwin brings all standard POSIX functions including the use of `main` instead of `WinMain` and I suspect this is a problem for graphical libraries like SDL which do require some sort of direct access to the OS low-level functions. Therefore, MinGW is more adapted for it does not define all these functions, and allows any library to hook on Microsoft Windows internals directly. Point is then, you also loose the cool effect of Cygwin which is to have a complete glibc available, including network functions like `select` defined the POSIX way, and not the WinSock way. If you ever ported code from POSIX sockets to WinSock 2, you know what I mean. Using MinGW is also embarassing for some libraries won't compile easily, and for instance programs which heavily rely on a real `TTY` interface to work are usually hard to port. This includes ncurses and GNU readline. Liquid War 6 tries to have workarrounds for all this, and in some cases the workarround is simply that embarassing code is not compiled on Microsoft Windows. For this reason, some features are not available on this platform. Period.

Now the reason you need MSYS and not only MinGW is that MSYS will allow `./configure` scripts to run, and this eases up the porting process a lot. MinGW and MSYS packages are downloadable on the SourceForge MinGW download page. Alternatively, there is a mirror on ufoot.org, but files might be outdated.

To compile Liquid War 6, first download and unzip all the following files in the same directory, for instance `C:\MSYS`. If you do not have any tool to handle `.tar.gz` and `.tar.bz2` files under Microsoft Windows, which is likely to be the case when MSYS is not installed yet, you can untar these on any GNU/Linux box, then upload the whole directory to the target Windows host.

- autoconf2.5-2.61-1-bin.tar.bz2
- autoconf-4-1-bin.tar.bz2
- autogen-5.9.2-MSYS-1.0.11-1-bin.tar.gz
- autogen-5.9.2-MSYS-1.0.11-1-dev.tar.gz
- autogen-5.9.2-MSYS-1.0.11-1-dll25.tar.gz
- automake1.10-1.10-1-bin.tar.bz2
- automake-3-1-bin.tar.bz2
- bash-3.1-MSYS-1.0.11-1.tar.bz2
- binutils-2.18.50-20080109-2.tar.gz
- bison-2.3-MSYS-1.0.11-1.tar.bz2
- coreutils-5.97-MSYS-1.0.11-snapshot.tar.bz2
- crypt-1.1-1-MSYS-1.0.11-1.tar.bz2

- csmake-3.81-MSYS-1.0.11-2.tar.bz2
- cvs-1.11.22-MSYS-1.0.11-1-bin.tar.gz
- diffutils-2.8.7-MSYS-1.0.11-1.tar.bz2
- findutils-4.3-MSYS-1.0.11-1.tar.bz2
- flex-2.5.33-MSYS-1.0.11-1.tar.bz2
- gawk-3.1.5-MSYS-1.0.11-1.tar.bz2
- gcc-core-3.4.5-20060117-3.tar.gz
- gcc-g++-3.4.5-20060117-3.tar.gz
- gcc-g77-3.4.5-20060117-3.tar.gz
- gcc-objc-3.4.5-20060117-3.tar.gz
- gdb-6.8-mingw-3.tar.bz2
- gdbm-1.8.3-MSYS-1.0.11-1.tar.bz2
- gettext-0.16.1-1-bin.tar.bz2
- gettext-0.16.1-1-dll.tar.bz2
- gettext-0.16.1-MSYS-1.0.11-1.tar.bz2
- gettext-devel-0.16.1-MSYS-1.0.11-1.tar.bz2
- inetutils-1.3.2-40-MSYS-1.0.11-2-bin.tar.gz
- libiconv-1.11-1-bin.tar.bz2
- libiconv-1.11-1-dll.tar.bz2
- libiconv-1.11-MSYS-1.0.11-1.tar.bz2
- libtool1.5-1.5.25a-1-bin.tar.bz2
- libtool1.5-1.5.25a-1-dll.tar.bz2
- libtool1.5-1.5.25a-20070701-MSYS-1.0.11-1.tar.bz2
- lndir-6.8.1.0-MSYS-1.0.11-1.tar.bz2
- lpr-1.0.1-MSYS.tar.gz
- lzma-4.43-MSYS-1.0.11-1-bin.tar.gz
- make-3.81-MSYS-1.0.11-2.tar.bz2
- mingw-runtime-3.14.tar.gz
- mingw-utils-0.3.tar.gz
- minires-1.01-1-MSYS-1.0.11-1.tar.bz2
- MSYS-1.0.11-20071204.tar.bz2
- msysCORE-1.0.11-2007.01.19-1.tar.bz2
- openssh-4.7p1-MSYS-1.0.11-1-bin.tar.gz
- openssl-0.9.8g-1-MSYS-1.0.11-2-bin.tar.gz
- openssl-0.9.8g-1-MSYS-1.0.11-2-dev.tar.gz
- openssl-0.9.8g-1-MSYS-1.0.11-2-dll098.tar.gz
- perl-5.6.1-MSYS-1.0.11-1.tar.bz2
- perl-man-5.6.1-MSYS-1.0.11-1.tar.bz2
- regex-0.12-MSYS-1.0.11-1.tar.bz2

- tar-1.19.90-MSYS-1.0.11-1-bin.tar.gz
- texinfo-4.11-MSYS-1.0.11-1.tar.bz2
- vim-7.1-MSYS-1.0.11-1-bin.tar.gz
- w32api-3.11.tar.gz
- zlib-1.2.3-MSYS-1.0.11-1.tar.bz2

This file list might contain file which are not absolutely mandatory for Liquid War 6, for instance the Fortran 77 compiler is absolutely useless, but installing it won't harm either. Some packages might unzip things the right way, but some do it in a subfolder. You might need to run commands like:

```
cp -r coreutils*/* .
rm -rf coreutils*
```

Get rid of useless files:

```
rm ._.DS_Store .DS_Store
```

It's also mandatory to move everything that has been installed in `/usr` or `/usr/local` to `/` since MSYS has some builtin wizardry which maps `/usr` on `/`. You need to do this if you don't unzip files from a MinGW shell, which is obviously the case when you first install it. Usefull command can be:

```
mv usr/* .
rmdir usr
```

Next, `libintl` is not correctly handled/detected by LW6, and can raise an error like `"gcc.exe: C:/msys/local/lib/.libs/libintl.dll.a: No such file or directory"` so one needs to copy some libraries in `/usr/local/lib/.libs/`:

```
mkdir local/lib/.libs
cp local/lib/libintl.* local/lib/.libs/
```

Another step is to edit `/etc/profile` and add lines like:

```
export CFLAGS="-g -I/usr/local/include"
export LDFLAGS="-L/usr/local/lib"
export GUILE_LOAD_PATH="C:\\MSYS\\local\\share\\guile\\1.8\\"
```

Close and re-launch your msys shell (rxvt) so that these changes take effect. Check that those values are correctly set:

```
env | grep FLAGS
env | grep GUILE
```

Finally, your MSYS environment is (hopefully...) working.

Now you need to compile the following programs, from source. Files are mirrored on ufoot.org for your convenience, however these might be outdated. Still, there are known to work. Proceed like if you were under a POSIX system. Some packages use the `--disable-rpath` swith, there are various reasons for which rpath is an issue. In the same manner, `--disable-nls` when linking against `libintl` or `libiconv` was painful.

- pthreads-win32, untar pthreads-w32-2-8-0-release.tar.gz then `make clean GC; cp pthread.h sched.h /usr/local/include/; cp pthreadGC2.dll /usr/local/bin/; cp libpthreadGC2.a /usr/local/lib/`
- GNU MP, untar `gmp-4.2.2.tar.gz` then `./configure && make && make install`

- Guile, untar `guile-1.8.5.tar.gz`. Edit `libguile/guile.c` and insert `#undef SCM_IMPORT` just before `#include <libguile.h>`. Edit `./libguile/threads.c` and place `struct timespec { long tv_sec; long tv_nsec; };` just before `#include "libguile/_scm.h"`. Then `./configure --disable-nls --disable-rpath --disable-error-on-warning --without-threads && make && make install`. The `GUILE_LOAD_PATH` value must be correctly set for `guile-config` to work. For unknown reasons, running `guile` can throw a stack overflow error. Don't panic. See bug 2007506 on SourceForge.net for an explanation on why the Guile binary shipped with MSYS is not suitable for Liquid War 6.
- expat, untar `expat-2.0.1.tar.gz` then `./configure && make && make install`
- SQLite, untar `sqlite-amalgamation-3.5.9.tar.gz` then `./configure && make && make install`
- libpng, untar `libpng-1.2.29.tar.gz` then `./configure && make && make install`
- libjpeg, untar `jpegsrc.v6b.tar.gz` then `./configure && make && make install && make install-lib`
- libcURL, untar `curl-7.18.1.tar.gz` then `./configure --without-ssl && make && make install`
- FreeType 2, untar `freetype-2.3.5.tar.gz` then `./configure && make && make install`
- libogg, untar `libogg-1.1.3.tar.gz` then `./configure && make && make install`
- libvorbis, untar `libvorbis-1.2.0.tar.gz` then `LDFLAGS="$LDFLAGS -logg" && ./configure && make && make install`
- SDL, untar `SDL-1.2.13.tar.gz` then `./configure && make && make install`
- SDL_image, untar `SDL_image-1.2.6.tar.gz` then `./configure && make && make install`
- SDL_mixer, untar `SDL_mixer-1.2.8.tar.gz` then `./configure && make && make install`
- SDL_ttf, untar `SDL_ttf-2.0.9.tar.gz` then `./configure && make && make install`

For your convenience, a zip file containing a complete MSYS "Liquid War 6 ready" environment is available. It is simply the result of all the operations described above. Simply unzip msys-for-liquidwar6-20080819.zip (about 240 megs) in `C:\MSYS\`. All dependencies compiled in `/local` have been generated using the command:

```
cd /usr/local/src
./msys-for-liquidwar6-build.sh > ./msys-for-liquidwar6-build.log 2>&1
```

Note that this script does't do everything, you'll still need to edit Guile source code and patch it manually.

It might even be possible to use this MSYS environment under Wine. Simply unzip it under `$HOME/.wine/drive_c`, and run `wine "$HOME/.wine/drive_c/windows/system32/cmd.exe" /c "c:\\msys\\msys.bat"` and with luck, you'll get a working shell. Note that this might allow you to compile the game, but running it is another story. Consider this MSYS over Wine trick as a hack enabling the use of free software only when compiling for Microsoft proprietary platform. It is not a reasonnable way to run the game. If running under a UNIXish platform, or better, GNU, simply run native code. Use the Windows 32-bit port only if you are jailed on a Microsoft system.

Now, let's come to the real meat, untar the Liquid War 6 source tarball, launch your MSYS shell, and:

```
./configure
make
make install
```

Now the binary is in `src/.libs/liquidwar6.exe` (beware, `src/liquidwar6.exe` is only a wrapper). This binary is an MSYS/MinGW binary, so it read paths " la" Microsoft, that is, it has no knowledge of what `/usr` is, for instance. It requires paths starting by `C:\`.

3.5 Coding guidelines

3.5.1 Project goals reminder

One of the purposes of Liquid War 6 is to make a cleaner implementation of Liquid War than the previous one, namely Liquid War 5. While the latter has achieved the practical goal of providing a playable implementation of the game, it failed at providing an evolutive platform. Network capabilities where finally added to Liquid War 5, but anyone who played on Internet with someone a few hundreds of milliseconds away would agree that it's far from being perfect. The main reason for this is that it is really had to hack on Liquid War 5, especially when you are not the core developper. The core developper himself, even knowing all the various hacks in the game, is very quickly lost when trying to implement major changes.

To put it short, Liquid War 5 is a global variable hell, a pile of hacks on top of a quick and dirty implementation. Still, it works.

With Liquid War 6, the idea is to take the time to make something stable, something nice which will enable developpers to implement the cool features, and have fun along the way. Of course, this is only a dream, and in the (hopefully `"very"`) long run, Liquid War 6 will also end up as a big unmaintainable mess, like any real-life program, until then, it should remain hackable.

3.5.2 Common sense

Here are a few guidelines which I think are common sense advice, but they are still worth mentionning:

- try and respect the GNU coding standards;
- absolutely no `strcpy` or `sprintf` anywhere in the code. Nowhere. Use their equivalent `strncpy` and `snprintf` systematically, as they are part of the glibc and are an order of magnitude safer. Moreover, Liquid War 6 provides wrappers, such as `lw6sys_new_sprintf` which handles all the nasty dirty memory allocation stuff for you;
- keep global variables for when there is something truely global, and even in that case try to fit them in clearly identified structures.

3.5.3 Unitary tests

Each of the internal libraries in Liquid War has a "test" program associated with it. For instance `liquidwar6sys-test` is associated to `libliquidwar6sys`, and its purpose is to test the features of this library.

While it is fairly easy to test out unitary functions which require no peculiar context, testing high-level functions which requires files, graphical and possibly network contexts to exist is obviously harder to achieve. There's no easy way to draw the line, but the idea is to put in these test executables as many features as possible, to be sure that what is tested in them is rock solid, bullet proof, and that one can safely rely on it and trust that code when running it in a more complex environnement.

These test executables are also very good places to see a library API in action, find code fragments, and make experiments.

3.5.4 Memory allocation

Liquid War 6 provides macros to allocate and free memory. One should use them systematically, except when trying to free something allocated by another library, and in very special cases, mostly concerning low-low level operations which are seldom hacked on.

Usage of macros `LW6SYS_MALLOC`, `LW6SYS_CALLOC` and `LW6SYS_FREE` is straightforward, read any random chunk of code, for instance `./src/lib/sys/sys-test.c` to see them in action. They are defined in `sys/sys.h`.

3.5.5 Private and public interfaces

Each library exports a public interface and hides its internal. Since Liquid War 6 uses standard C and no C++, there's no real standard way to handle public/private features. The convention used in Liquid War 6 is to show internal structures as opaque pointers (`void *`) whenever some function needs to operate on a structure which has possibly private fields. This way the caller function has no way to access the internals, and we are sure that no reference to any internal implementation specific feature will appear.

Here's a code excerpt from `src/gfx/setup.c`:

```
void _lw6gfx_quit(_LW6GFX_CONTEXT *context) {
  /*
   * Implementation here.
   */
[...]
}

void lw6gfx_quit(void *context) {
  _lw6gfx_quit((_LW6GFX_CONTEXT *) context);
}
```

The function `_lw6gfx_quit` (note the "_") is internal, declared in `internal.h` whereas the function `lw6gfx_quit` is public, and is therefore exported in `gfx.h`.

This way, functions in the program using `lw6gfx_quit` do not know what is in the `_LW6GFX_CONTEXT` structure, and they need not know it.

This does not mean it is not possible to have public structures, only these structures must reflect some truely public, accessible and safe to access structures.

3.6 Using the console

The console can be activated by passing `--display-console` when starting the game or by using the system options menu.

When the console is activated, a `lw6>` prompt should appear in the terminal which launched the program. If you started Liquid War 6 by clicking on an icon, console probably won't work at all since `stdout` and `stdin` won't be attached to anything.

The console allows you to type arbitray Scheme/Guile code.

Try, for instance:

```
(+ 1 2)
(display "foo\n")
```

You can really break things with this console, it gives you a direct access to all the program internals. At least, all the values which are accessible through the script interface, that is, many of them.

You can call any internal C function which has been exported to Guile, here are some examples:

```
(c-lw6gfx-get-ticks)
(c-lw6bot-get-backends)
(c-lw6sys-sleep 2.0)
(lw6-config-get-number "zoom")
(lw6-config-set-number! "zoom" 0.9)
(lw6-config-get-number "zoom")
```

While syntax (and possibly other) errors will be trapped by the interpreter, note that if you break things inside the game by, say, changing some global value, or in a general manner cause an error elsewhere in the code, the game will really raise a fatal error and stop. That's how you can "break things".

Still, this console is a very powerfull tool, very usefull for debugging but also for tweaking the game without restarting it and/or navigating through the menu interface.

3.7 Advanced tweaking

Todo...

3.8 Writing modules

Todo...

3.9 Use as a library

Todo...

3.10 Network protocol

This section describes how Liquid War 6 handles network messages. Note that for now this is purely theorical, more of a draft, a plan, it might change before being implemented.

Bare technical stuff.

Out of band messages:

- UPTIME -> uptime in seconds
- ROUND -> number of rounds (session related)
- BPS -> number of net messages send per second

- LIST -> list all known servers
- PEERS -> list servers with active connections
- VERSION -> version of the program
- ID -> the id of this server
- SESSION -> the id of the session
- PING -> expects PONG
- URL -> an alternate URL, possibly proxied
- MAX -> max number of teams on this server/session
- PLAYERS -> players playing
- COLORS -> colors playing
- LOCKED -> expects "YES" or "NO"
- MOTD -> message of the day

TCP messages:

```
LW6 <passwd> <client-id>
MSG1
MSG2
```

UDP messages:

```
LW6 <passwd> <client-id> MSG1
LW6 <passwd> <client-id> MSG2
```

HTTP messages:

```
/lw6/<passwd>/<client-id>/MSG1
/lw6/<passwd>/<client-id>/MSG2
```

HTTP public URLs:

```
error 404 -> /lw6/
/lw6/ -> HTML human readbable page
/favicon.ico
/lw6/favicon.ico
/lw6/screenshot.jpg
/lw6/<oob>
```

MSG syntax:

```
<serial>-<i>-<n> COMMAND
```

COMMAND format:

```
<round> <server-id> <command> <arg1> ... <argN>
```

COMMAND examples:

```
2 1234abcd1234abcd REGISTER
3 1234abcd1234abcd ADD 5678 YELLOW
4 1234abcd1234abcd SET 5678 20 5
10 1234abcd1234abcd NOP
400 1234abcd1234abcd REMOVE 5678
1000 1234abcd1234abcd UNREGISTER
```

3.11 Using GNU Arch

3.11.1 About GNU Arch

There is no CVS or SVN repository for Liquid War 6. Instead, a GNU Arch repository is used to follow the different versions. Read the GNU Arch tutorial to learn how Arch works. Note that there are many other source control managers available, some of which provide functionnalities similar to GNU Arch / tla. GNU Arch has been chosen for Liquid War 6 because:

- it is Free Software,
- it is not limited to per-file commits like CVS, and supports atomic commits involving several files,
- it is distributed,
- it enables developpers to sign each of their contributions,
- it was already available back in 2005.

3.11.2 Getting the latest version from the repository

The repository for Liquid War 6 is accessible on `http://arch.savannah.gnu.org/archives/liquidwar6`. This is a read-only access, but with the distributed nature of GNU Arch, it still allows you to keep track of your own changes, and submit patches. Accessing it in read/write mode with sftp requires a Savannah account and special rights on the Liquid War 6 project.

Here are typicall commands one can use to get Liquid War 6 source from the GNU Arch repository:

```
tla register-archive http://arch.savannah.gnu.org/archives/liquidwar6
tla get -A liquidwar6@sv.gnu.org liquidwar6--beta
```

All the patches in the archive are signed with GnuPG, so you can check their authenticity with my public key.

You might need to edit your `$HOME/.arch-params/signing/=default.check` file and put the following text in it:

```
tla-gpg-check gpg_command="gpg --verify-files -"
```

3.11.3 Setting up your own arch repository

This section is for those who want to hack the game and set up their own repositories. This will enable you to keep track of your patches, package them, and help the core maintainer merging them in the main repository.

You can introduce yourself and create a repository by issuing commands like:

You can introduce yourself and create a repository by issuing commands like:

```
tla my-id me@home.net
tla register-archive me@home.net--2008 /home/me/tla-archives
```

Then, you can get create your own repository, with a command like:

```
tla tag -S liquidwar6@sv.gnu.org/liquidwar6--beta--0.1 me@home.net--2008/liquidwar6--b
```

The idea is that you create, locally, a depot which has a name that matches the name on savannah (this is for convenience, you could technically give it any name...) and indicate that they represent, today, the same thing.

You can get a working copy of your depot with the command:

```
tla get me@home.net--2008/liquidwar6--beta--0.4
```

This will create a complete source tree, which you are free to modify, this is where you should hack.

3.11.4 Synchronizing your repository with upstream releases

To synchronize yourself with upstream developments, go into your copy (the directory created by `tla get`) and type:

```
tla star-merge liquidwar6@sv.gnu.org/liquidwar6--beta--0.1
```

This will apply locally all the changes that happened since the last synchronization. Of course this is one way to work, you can decide to cherry pick patches and such stuff, but for most dayly uses, a good'ol `star-merge` is fine.

Not that `star-merge` will only apply patches on your working copy, not on your repository. The only way to actually commit the modifications on the repository is to use the `commit` command.

3.11.5 Submitting patches

When using Arch, you can of course still send patches created with `diff`, or even send updated files directly, the way you would without revision control.

But it can be more convenient to either

- Send the patches stored in the depot (`/home/me/tla-archives` in our example).
- Make patches using `tla mkpatch`.

Here's an example of an `mkpatch` command, and which will compute the differences between a previous `liquidwar6--beta--0.4--patch-2` snapshot and a not yet commited latest version:

```
tla mkpatch {arch}/++pristine-trees/unlocked/liquidwar6/liquidwar6--beta/liquidwar6--b
```

This will create a `my-patch` directory, which can be gzipped and sent by mail.

3.11.6 Recover from broken lock

Sometimes, when signing a patch, you might enter the wrong passphrase several times, or you might press CTRL+D inadvertantly. In that case, tla will be in a half-broken state, telling you it can't acquire revision lock... A quick workarround for this is to go to the depot, find the latest patch, and in this repository, create the following folders:

```
++revision-lock/+contents
```

Both are directories, note the two `++` and the single `+`. the `+contents` directory can be empty. Once you've done this, try to re-commit.

4 Reference

This chapter is a technical reference. Most of its content is self-generated by the program itself. That is to say, most if its content is already available to you if you have the game installed. Running `liquidwar6 --list` and `liquidwar6 --about=<keyword>` is very likely to give you the very same informations, the advantage being that you'll be sure the information is up-to-date and corresponds to the exact version of the program you have. However, publishing this in a reader-friendly way is convenient, plus it enables web search engines to harvest the content.

4.1 Basic options

4.1.1 about

`--about=<value>` [Command-line option]

Type: string.

Will allow you to get informations about a given keyword. Let's say that, for instance, you want informations about the keyword 'map-path'. Simply run 'liquidwar6 –about=map-path'. Note that this internal self-documentation system can describe command line switches as well as XML config file parameters or environment variables, and even some Guile script functions. The '–list' command line switch will give you the list of all available keywords.

4.1.2 copyright

`--copyright` [Command-line option]

Returns the copyright notice for the program.

4.1.3 debug

`--debug` [Command-line option]

Enables debug mode. This will turn on maximum log information, and display everything on stderr, even messages which are normally only stored in the log file.

4.1.4 defaults

`--defaults` [Command-line option]

Clears the config file and run the game with default settings. The difference with '–reset' is that '–defaults' runs the game, while '–reset' does not.

4.1.5 help

`--help` [Command-line option]

Returns a short help for the program.

4.1.6 list

`--list` [Command-line option]

Returns the list of all keywords which can be queried for information. This includes command-line options, environment variables, and so on. This is the companion option of '–about'. Results obtained with '–list' can be passed to '–about'.

4.1.7 pedigree

`--pedigree` [Command-line option]

Display all build values, these are general constants which can help debugging, tracing what binary you are running, and so on. It's a good idea to take a look at the output of 'pedigree' if you have problems running the game.

4.1.8 reset

`--reset` [Command-line option]

Clears the config file so that the game will run with defaults next time. The idea is to get rid of traces of previous executions.

4.1.9 test

`--test` [Command-line option]

Runs a (hopefully) complete test suite which will call most internal Liquid War 6 functions and check out wether they work, in a simple context, without any game interference. Usefull for troubleshooting.

4.1.10 version

`--version` [Command-line option]

Returns the version of the program, as defined by the GNU Coding Standards.

4.2 Doc options

4.2.1 example-hints-xml

`--example-hints-xml` [Command-line option]

Dumps on stdout an example hints.xml file. Such a file is normally shipped with the game. It is indeed generated using this command.

4.2.2 example-rules-xml

`--example-rules-xml` [Command-line option]

Dumps on stdout an example options.xml file. Such a file is normally shipped with the game. It is indeed generated using this command.

4.2.3 example-style-xml

`--example-style-xml` [Command-line option]

Dumps on stdout an example style.xml file. Such a file is normally shipped with the game. It is indeed generated using this command.

4.2.4 list-aliases

`--list-aliases` [Command-line option]
List the keyword aliases. These are here for convenience.

4.2.5 list-doc

`--list-doc` [Command-line option]
List documentation-related command line options. These commands allow you to list all the keywords related to a given domain.

4.2.6 list-funcs

`--list-funcs` [Command-line option]
List the C-functions which are exported to Guile, thus usable in scripts.

4.2.7 list-graphics

`--list-graphics` [Command-line option]
List graphics options (resolution, fullscreen...).

4.2.8 list-hooks

`--list-hooks` [Command-line option]
List user-modifiable hooks.

4.2.9 list-input

`--list-input` [Command-line option]
List input (AKA controls) related options. Use these to change keyboard, joystick and mouse settingds.

4.2.10 list-map

`--list-map` [Command-line option]
List map-related entries, excluding rules.xml, hints.xml and style.xml entries.

4.2.11 list-map-hints

`--list-map-hints` [Command-line option]
List 'hints.xml' entries. These parameters enable you to modify the behavior of the map loader.

4.2.12 list-map-rules

`--list-map-rules` [Command-line option]
List 'options.xml' entries. These parameters enable you to modify the gameplay.

4.2.13 list-map-style

`--list-map-style` [Command-line option]
List 'style.xml' entries. These parameters enable you to modify the aspect of the game.

4.2.14 list-network

`--list-network` [Command-line option]
List network options.

4.2.15 list-path

`--list-path` [Command-line option]
List parameters which allow you to override the defaults of the game, and force the game your own file paths and directories.

4.2.16 list-players

`--list-players` [Command-line option]
List player-related entries, that is to say 'who plays'.

4.2.17 list-quick

`--list-quick` [Command-line option]
List quick help entries, this includes the GNU standard options and a few troubleshooting tools.

4.2.18 list-show

`--list-show` [Command-line option]
List command-line options which begin with '–show-...'. These will display on the console many internal parameters. Usefull when debugging.

4.2.19 list-sound

`--list-sound` [Command-line option]
List sound options (volume...).

4.2.20 list-tuning

`--list-tuning` [Command-line option]
List advanced options which can be used for fine-tuning the game.

4.3 Show options

4.3.1 show-build-cflags

`--show-build-cflags` [Command-line option]
Shows what value you should put in 'CFLAGS' (environment variable) if you want to compile programs that use Liquid War 6 as a library, and include 'liquidwar6.h'.

4.3.2 show-build-codename

`--show-build-codename` [Command-line option]
Shows the codename associated with this version, generally the name of someone famous who is war-related (a general, an emperor...).

4.3.3 show-build-configure-args

`--show-build-configure-args` [Command-line option]
Shows the arguments that have been passed to the GNU Autoconf './configure' script when building the program. This can be very usefull if you want to know how the program has been built.

4.3.4 show-build-copyright

`--show-build-copyright` [Command-line option]
Shows a very short copyright notice.

4.3.5 show-build-datadir

`--show-build-datadir` [Command-line option]
Shows the 'datadir' value as passed to the GNU Autoconf './configure' script when compiling the program. Default is '/usr/local/share'. This is the generic, non Liquid War 6 specific data directory. Liquid War 6 related data is stored elsewhere (usually in a sub-directory), see the 'data-dir' switch for more information. 'datadir' is not 'data-dir'. That's the point.

4.3.6 show-build-date

`--show-build-date` [Command-line option]
Shows the date when the binary was compiled.

4.3.7 show-build-docdir

`--show-build-docdir` [Command-line option]
Shows the 'docdir' value as passed to the GNU Autoconf './configure' script when compiling the program. Default is '/usr/local/share/doc/liquidwar6'.

4.3.8 show-build-enable-allinone

`--show-build-enable-allinone` [Command-line option]
Shows wether the 'allinone' option has been chosen when building the game. This depends on parameters passed to './configure'.

4.3.9 show-build-enable-console

`--show-build-enable-console` [Command-line option]
Shows wether the console has been enabled when building the game. This depends on parameters passed to './configure' and also on the presence of ncurses and readline.

4.3.10 show-build-enable-fullstatic

`--show-build-enable-fullstatic` [Command-line option]
Shows wether the 'fullstatic' option has been chosen when building the game. This depends on parameters passed to './configure'.

4.3.11 show-build-enable-gprof

`--show-build-enable-gprof` [Command-line option]
Shows wether the game was build with suitable informations for gprof. This depends on parameters passed to './configure'.

4.3.12 show-build-enable-mod-csound

`--show-build-enable-mod-csound` [Command-line option]
Shows wether the mod-csound audio backend has been enabled when building the game. This depends on parameters passed to './configure' and also on the presence of the csound library.

4.3.13 show-build-enable-mod-gl

`--show-build-enable-mod-gl` [Command-line option]
Shows wether the mod-gl graphical backend has been enabled when building the game. This depends on parameters passed to './configure' and also on the presence of SDL and related libraries.

4.3.14 show-build-enable-mod-http

`--show-build-enable-mod-http` [Command-line option]
Shows wether the mod-http network backend has been enabled when building the game. This depends on parameters passed to './configure' and also on the presence of libCurl.

4.3.15 show-build-enable-mod-ogg

`--show-build-enable-mod-ogg` [Command-line option]
Shows wether the mod-ogg audio backend has been enabled when building the game. This depends on parameters passed to './configure' and also on the presence of SDL and related libraries.

4.3.16 show-build-enable-optimize

`--show-build-enable-optimize` [Command-line option]
Shows wether the 'optimize' option has been chosen when building the game. This depends on parameters passed to './configure'.

4.3.17 show-build-enable-valgrind

`--show-build-enable-valgrind` [Command-line option]
Shows wether the game was build with valgrind later use in mind. This depends on parameters passed to './configure'.

4.3.18 show-build-endianness

`--show-build-endianness` [Command-line option]
Returns the endianness. 'little' corresponds to x86-like systems, 'big' to ppc-like systems.

4.3.19 show-build-gcc-version

`--show-build-gcc-version` [Command-line option]
Returns the version of the GNU C compiler which was used to compile the program.

4.3.20 show-build-hostname

`--show-build-hostname` [Command-line option]
Shows the name of the host where the binary was compiled.

4.3.21 show-build-includedir

`--show-build-includedir` [Command-line option]
Shows the 'includedir' value as passed to the GNU Autoconf './configure' script when compiling the program. Default is '/usr/local/include'.

4.3.22 show-build-ldflags

`--show-build-ldflags` [Command-line option]
Shows what value you should put in 'LDFLAGS' (environment variable) if you want to link programs against libliquidwar6.

4.3.23 show-build-libdir

`--show-build-libdir` [Command-line option]
Shows the 'libdir' value as passed to the GNU Autoconf './configure' script when compiling the program. Default is '/usr/local/lib'. This is the generic, non Liquid War 6 specific library directory. Dedicated Liquid War 6 modules are stored elsewhere (usually in a sub-directory), see the 'mod-dir' switch for more information.

4.3.24 show-build-license

`--show-build-license` [Command-line option]
Shows the license of the program (GNU GPL v3 or later).

4.3.25 show-build-localedir

`--show-build-localedir` [Command-line option]
Shows the 'localedir' value as passed to the GNU Autoconf './configure' script when compiling the program. Default is '/usr/local/share/locale'.

4.3.26 show-build-md5sum

`--show-build-md5sum` [Command-line option]
Shows the MD5 checksum, which has been calculated from the C source files. Complementary with 'show-build-stamp'.

4.3.27 show-build-ms-windows

`--show-build-ms-windows` [Command-line option]
Returns 1 (true) if target OS is Microsoft Windows 32-bit platform, 0 (false) if not.

4.3.28 show-build-package-name

`--show-build-package-name` [Command-line option]
Shows the package name, that is, 'Liquid War 6'.

4.3.29 show-build-package-string

`--show-build-package-string` [Command-line option]
Shows the package string, that is, 'Liquid War 6 <version>

4.3.30 show-build-package-tarname

`--show-build-package-tarname` [Command-line option]
Shows the package tarname, that is, liquidwar6.

4.3.31 show-build-pointer-size

`--show-build-pointer-size` [Command-line option]
Returns the pointer size, in bytes. Should be 4 on 32-bit systems and 8 on 64-bit systems.

4.3.32 show-build-prefix

`--show-build-prefix` [Command-line option]
Shows the 'prefix' value as passed to the GNU Autoconf './configure' script when compiling the program. Default is '/usr/local'.

4.3.33 show-build-stamp

`--show-build-stamp` [Command-line option]
Shows the build stamp. A very usefull value, more precise than the version to track down binaries. It is incremented each time the core C code is updated. It won't reflect all the programs for it does not take scripts in account, but if you are running a work-in-progress version, it might be very convenient to use this to know what your are running exactly.

4.3.34 show-build-target-cpu

`--show-build-target-cpu` [Command-line option]
Shows the target CPU, as defined by 'target_cpu' in GNU Autoconf.

4.3.35 show-build-target-os

`--show-build-target-os` [Command-line option]
Shows the target OS, as defined by 'target_os' in GNU Autoconf.

4.3.36 show-build-time

`--show-build-time` [Command-line option]
Shows the time when the binary was compiled.

4.3.37 show-build-top-srcdir

`--show-build-top-srcdir` [Command-line option]
Shows the top source directory on the machine where the binary was compiled.

4.3.38 show-build-version

`--show-build-version` [Command-line option]
Shows the version. Note that this is different from the standard GNU 'version' command line option which shows a complete message with a short copyright notice. This one will just return the version, without the package tarname or anything else.

4.3.39 show-config-file

`--show-config-file` [Command-line option]
Shows the config file path. Default is '$HOME/.liquidwar6/config.xml'.

4.3.40 show-cwd

`--show-cwd` [Command-line option]
Shows the current working directory, the value that the pwd command would return.

4.3.41 show-data-dir

`--show-data-dir` [Command-line option]
Shows the data directory path. This is where the games searches for most of its data,the most important exception being maps, which are stored elsewhere. Default is '/usr/local/share/liquidwar6-<version>/data'.

4.3.42 show-default-config-file

`--show-default-config-file` [Command-line option]
Shows the default config file path. Default is '$HOME/.liquidwar6/config.xml'.

4.3.43 show-default-data-dir

`--show-default-data-dir` [Command-line option]
Shows the default data directory path. This is where the games searches for most of its data,the most important exception being maps, which are stored elsewhere. Default is '/usr/local/share/liquidwar6-<version>/data'.

4.3.44 show-default-log-file

`--show-default-log-file` [Command-line option]
Shows the default log file path. Default is '$HOME/.liquidwar6/log.csv'.

4.3.45 show-default-map-dir

`--show-default-map-dir` [Command-line option]
Shows the default map directory. This is where builtin maps are stored. Default is '/usr/local/share/liquidwar6-<version>/map'.

4.3.46 show-default-map-path

`--show-default-map-path` [Command-line option]
Shows the default map search path. This is where the game searches for maps. It's the combination of command-line arguments and builtin paths. Might return more directories than the one specified in a single 'map-path=dir1:dir2' argument.

4.3.47 show-default-mod-dir

`--show-default-mod-dir` [Command-line option]
Shows the default module directory path. This is where all dynamically loaded modules are stored. Default is '/usr/local/lib/liquidwar6-<version>'.

4.3.48 show-default-prefix

`--show-default-prefix` [Command-line option]
Shows the default prefix used. This should logically be the value passed to the GNU Autoconf ./configure script when building the game. Most other path are deduced from this one. Default is '/usr/local'.

4.3.49 show-default-script-file

`--show-default-script-file` [Command-line option]
Shows the default main script file path. This file is very important, since the program is more or less a hudge scheme interpreter, and this file is the file loaded by Guile. In short, it is the main program. Default is '/usr/local/share/liquidwar6-<version>/script/liquidwar6.scm'.

4.3.50 show-default-user-dir

`--show-default-user-dir` [Command-line option]
Shows the default user directory path. This is where run-time data, config files, log files, are stored. Default is '$HOME/.liquidwar6/'.

4.3.51 show-log-file

`--show-log-file` [Command-line option]
Shows the log file path. Default is '$HOME/.liquidwar6/log.csv'.

4.3.52 show-map-dir

`--show-map-dir` [Command-line option]
Shows the map directory. This is where builtin maps are stored. Default is '/usr/local/share/liquidwar6-<version>/map'.

4.3.53 show-map-path

`--show-map-path` [Command-line option]
Shows the map search path. This is where the game searches for maps. It's the combination of command-line arguments and builtin paths. Might return more directories than the one specified in a single 'map-path=dir1:dir2' argument.

4.3.54 show-mod-dir

`--show-mod-dir` [Command-line option]

Shows the module directory path. This is where all dynamically loaded modules are stored. Default is '/usr/local/lib/liquidwar6-<version>'.

4.3.55 show-prefix

`--show-prefix` [Command-line option]

Shows the prefix used. This should logically be the value passed to the GNU Autoconf ./configure script when building the game. Most other path are deduced from this one. Default is '/usr/local'.

4.3.56 show-run-dir

`--show-run-dir` [Command-line option]

Shows the run directory, usually the path where the binary is. It depends on how and where the program is launched. It is guessed from the argc/argv values at runtime.

4.3.57 show-script-file

`--show-script-file` [Command-line option]

Shows the main script file path. This file is very important, since the program is more or less a hudge scheme interpreter, and this file is the file loaded by Guile. In short, it is the main program. Default is '/usr/local/share/liquidwar6-<version>/script/liquidwar6.scm'.

4.3.58 show-user-dir

`--show-user-dir` [Command-line option]

Shows the user directory path. This is where run-time data, config files, log files, are stored. Default is '$HOME/.liquidwar6/'.

4.4 Path options

4.4.1 config-file

`--config-file` [Command-line option]
`LW6_CONFIG_FILE` [Environment variable]

Type: string.

Set the config file path. This enables you to use whatever config file you like, keeping all other informations in the same place. Default is '$HOME/.liquidwar6/config.xml'.

4.4.2 data-dir

`--data-dir` [Command-line option]
`LW6_DATA_DIR` [Environment variable]

Type: string.

Set the data directory. By changing ths value you'll be able to use an alternative data directory. Default is '/usr/local/share/liquidwar6-<version>/data'.

4.4.3 log-file

`--log-file=<value>` [Command-line option]
`LW6_LOG_FILE` [Environment variable]
`log-file` [XML key]

Type: string.

Set the log file path. This enables you to use whatever log file you like, keeping all other informations in the same place. Default is '$HOME/.liquidwar6/log.csv'.

4.4.4 map-dir

`--map-dir` [Command-line option]
`LW6_MAP_DIR` [Environment variable]

Type: string.

Set the map directory path. By changing this value you'll be able to play with your own maps in your own directory. Note that there are other ways to achieve that, but using this option will work. However, a side effect is that you might not see builtin maps anymore. Default is '/usr/local/share/liquidwar6-<version>/map'.

4.4.5 map-path

`--map-path=<value>` [Command-line option]
`LW6_MAP_PATH` [Environment variable]
`map-path` [XML key]

Type: string.

Set the map search path. By changing this value you'll be able to play with your own maps in your own directory. This is different from 'map-dir', since it includes 'map-dir', plus it adds a number of other search paths. Unlike most other parameters, the values given from the command-line, from the environment variables, or from the config file, are not overwritten, but appended. That is to say if you specify a 'map-path' with the command-line argument 'map-path=path', but also define the 'LW6_MAP_PATH' value and finally edit 'config.xml' to change the 'map-path' entry in it, you'll end up with the game searching for maps in all these directories. Additionnally, 'map-dir' and '<user-dir>/map' will always be in the list. Any given value can itself include several pathes, separated by the path separator. This separator is ':' on GNU/Linux, and ';' on Microsoft Windows. For instance, on a GNU/Linux box, you could use the command-line argument 'map-path=/foo/bar/map:/home/user/map/:/map'.

4.4.6 mod-dir

`--mod-dir` [Command-line option]
`LW6_MOD_DIR` [Environment variable]

Type: string.

Set the module directory path. By changing this you will load dynamic shared libraries (game specific modules such as the graphical backend) from an alternative place. Use this at your own risks, for there can always be a binary incompatibility. You've been warned. Default is '/usr/local/lib/liquidwar6-<version>'.

4.4.7 prefix

`--prefix` [Command-line option]
`LW6_PREFIX` [Environment variable]

Type: string.

Override the prefix value given to the GNU Autoconf ./configure script when building the game. Not all path will be changed, some of them might remain the same, for instance message translations (localedir). But most game-specific data including maps, graphics, sounds, will be searched according to the new given parameter. Default is '/usr/local'.

4.4.8 script-file

`--script-file` [Command-line option]
`LW6_SCRIPT_FILE` [Environment variable]

Type: string.

Set the main script file path. This file is very important, since the program is more or less a hudge scheme interpreter, and this file is the file loaded by Guile. In short, it is the main program. Default is '/usr/local/share/liquidwar6-<version>/script/liquidwar6.scm'.

4.4.9 user-dir

`--user-dir=<value>` [Command-line option]
`LW6_USER_DIR` [Environment variable]
`user-dir` [XML key]

Type: string.

Set the user directory path. This is where run-time data, config files, log files, are stored. If you override this value, other parameters such as where the config and log files reside, will change. Default is '$HOME/.liquidwar6'.

4.5 Graphics options

4.5.1 fullscreen

`--fullscreen=<value>` [Command-line option]
`LW6_FULLSCREEN` [Environment variable]
`fullscreen` [XML key]

Type: boolean.

Force the game to fun fullscreen. Note that the graphics backend might ignore this hint.

4.5.2 gfx-backend

`--gfx-backend=<value>` [Command-line option]
`LW6_GFX_BACKEND` [Environment variable]
`gfx-backend` [XML key]

Type: string.

Sets the graphics backend AKA 'gfx' to use. For now the only choice is 'gl' and will use an OpenGL/SDL 3D-accelerated driver.

4.5.3 height

`--height=<value>` [Command-line option]
`LW6_HEIGHT` [Environment variable]
`height` [XML key]

Type: integer.

Run the game with the given screen height.Note that the graphics backend might ignore this hint.Use with its companion option 'width'.

4.5.4 preset-resolution

`--preset-resolution=<value>` [Command-line option]
`LW6_PRESET_RESOLUTION` [Environment variable]
`preset-resolution` [XML key]

Type: integer.

The last preset resolution used. 0 is low, 1 is medium, 2 is high. Real resolutions depend on your hardware, OS and drivers.

4.5.5 width

`--width=<value>` [Command-line option]
`LW6_WIDTH` [Environment variable]
`width` [XML key]

Type: integer.

Run the game with the given screen width. Note that the graphics backend might ignore this hint.Use with its companion option 'height'.

4.5.6 windowed-mode-limit

`--windowed-mode-limit=<value>` [Command-line option]
`LW6_WINDOWED_MODE_LIMIT` [Environment variable]
`windowed-mode-limit` [XML key]

Type: float.

When switching back from fullscreen mode to windowed mode, if we're in maximum resolution, then this coefficient will be applied before resizing the window. The idea is that (obviously) a windowed mode is prefered when a little smaller that totally fullscreen. So set this to a value just below 1.0.

4.6 Sound options

4.6.1 music-volume

`--music-volume=<value>` [Command-line option]
`LW6_MUSIC_VOLUME` [Environment variable]

`music-volume` [XML key]

Type: float.

Set the music volume.This is a floating point value.0 is mute.Maximum value is 1.

4.6.2 snd-backend

`--snd-backend=<value>` [Command-line option]
`LW6_SND_BACKEND` [Environment variable]
`snd-backend` [XML key]

Type: string.

Sets the sound backend AKA 'snd' to use. Can be 'ogg' or 'csound' but only 'ogg' will produce sound in the current release.

4.6.3 sound-volume

`--sound-volume=<value>` [Command-line option]
`LW6_SOUND_VOLUME` [Environment variable]
`sound-volume` [XML key]

Type: float.

Set the sound volume.This is a floating point value.0 is mute.Maximum value is 1.

4.7 Network options

4.8 Map parameters

4.8.1 chosen-map

`--chosen-map=<value>` [Command-line option]
`LW6_CHOSEN_MAP` [Environment variable]
`chosen-map` [XML key]

Type: string.

The last map chosen by the player, locally. This is the map which will be used for a quick-start game, a local game, or a game started as a server.

4.8.2 force

`--force=<value>` [Command-line option]
`LW6_FORCE` [Environment variable]
`force` [XML key]

Type: string.

A comma separated list of options which should be ignored when reading map XML files. For instance, if this contains 'rounds-per-sec,moves-per-round' then whatever values were defined for this in 'rules.xml', then game will ignore them and use the user's values, stored in 'config.xml', running the game at the requested speed. This ultimately allows the player to control everything despite the values set by the map designer.

4.8.3 use-hints-xml

`--use-hints-xml=<value>` [Command-line option]
`LW6_USE_HINTS_XML` [Environment variable]
`use-hints-xml` [XML key]

Type: boolean.

If set, then hints will be picked up from the map defined hints.xml, if it exists. This is the default.

4.8.4 use-rules-xml

`--use-rules-xml=<value>` [Command-line option]
`LW6_USE_RULES_XML` [Environment variable]
`use-rules-xml` [XML key]

Type: boolean.

If set, then rules will be picked up from the map defined rules.xml, if it exists. This is the default. Use force-time and force-size to override this and use user-defined values anyway.

4.8.5 use-style-xml

`--use-style-xml=<value>` [Command-line option]
`LW6_USE_STYLE_XML` [Environment variable]
`use-style-xml` [XML key]

Type: boolean.

If set, then style will be picked up from the map defined style.xml, if it exists. This is the default. Use force-time and force-background to override this and use user-defined values anyway.

4.8.6 use-texture

`--use-texture=<value>` [Command-line option]
`LW6_USE_TEXTURE` [Environment variable]
`use-texture` [XML key]

Type: boolean.

Defines wether the map texture should be used. Of course if there's no map texture, the texture... won't be used. But if there is one, this parameter will force the game to ignore it and play with solid colors. This probably won't look as nice as the textured map in most cases, but some players might find it more readable and confortable to play when throwing eye candy away.

4.9 Map rules.xml

4.9.1 color-conflict-mode

`--color-conflict-mode=<value>` [Command-line option]
`LW6_COLOR_CONFLICT_MODE` [Environment variable]

`color-conflict-mode` [XML key]

Type: integer.

How to handle color conflicts, that is, when a player requests a color, but this color is already used, what should be done? If 0, wether a color already exists won't affect the color of a new cursor. If 1, then two players on the same computer will be allowed to share the same color/team, but if another computer is already playing with a color, any new computer will need to use another team. If 2, then it's impossible for a new cursor to use a pre-existing color, any new cursor will require a new color, if that color is already used, a new color will be picked randomly.

4.9.2 cursor-pot-init

`--cursor-pot-init=<value>` [Command-line option]
`LW6_CURSOR_POT_INIT` [Environment variable]
`cursor-pot-init` [XML key]

Type: integer.

Defines the cursor potential at startup. Not really any reason to change it. Theorically, there could be maps where the default value doesn't fit, but none has been seen yet.

4.9.3 fighter-attack

`--fighter-attack=<value>` [Command-line option]
`LW6_FIGHTER_ATTACK` [Environment variable]
`fighter-attack` [XML key]

Type: integer.

Defines how hard fighters will attack others, that is, in one attack, how many life-points the attacked fighter will loose. Increasing this will cause your opponents to melt faster when you attack them. With a low value, it will take ages to take on your opponents. Different styles of game. Can radically change the gameplay.

4.9.4 fighter-defense

`--fighter-defense=<value>` [Command-line option]
`LW6_FIGHTER_DEFENSE` [Environment variable]
`fighter-defense` [XML key]

Type: integer.

Defines how fast fighters will regenerate after an attack. When this parameter is set low, an attacked fighter, which is very dark and almost dead will take a very long time to regain energy. If the parameter is set high, it can almost instantaneously regain energy.

4.9.5 fighter-new-health

`--fighter-new-health=<value>` [Command-line option]
`LW6_FIGHTER_NEW_HEALTH` [Environment variable]
`fighter-new-health` [XML key]

Type: integer.

Defines how healthy fighters will be when they appear on the map. This can be either at the beginning of the game of when a fighter changes team. Setting this low will allow battefields to switch from one side to another very fast, for freshly gained fighters will be feeble and very likely to return to their original camp. To calibrate this parameter, keep in mind that the absolute maximum health a fighter can have is always 10000 (ten-thousands).

4.9.6 fighter-regenerate

`--fighter-regenerate=<value>` [Command-line option]
`LW6_FIGHTER_REGENERATE` [Environment variable]
`fighter-regenerate` [XML key]

Type: integer.

Defines at which speed fighters will self-regenerate, without even begin packed together. This will allow lone fighters to regenerate a bit by hiding somewhere in the map. This is typically a low value, might even be 0.

4.9.7 max-cursor-pot

`--max-cursor-pot=<value>` [Command-line option]
`LW6_MAX_CURSOR_POT` [Environment variable]
`max-cursor-pot` [XML key]

Type: integer.

Defines the maximum cursor potential. Not really any reason to change it. Any high value should produce the same results. Low values might reveal algorithm bugs and inconsistencies.

4.9.8 max-cursor-pot-offset

`--max-cursor-pot-offset=<value>` [Command-line option]
`LW6_MAX_CURSOR_POT_OFFSET` [Environment variable]
`max-cursor-pot-offset` [XML key]

Type: integer.

Defines the maximum cursor potential offset. The idea is that in some cases, the potential of a cursor can increase in burst mode, for instance to make this cursor more important than others, so that fighters rally to it, neglecting other cursors (talking about a multi-cursor controlled team). This parameter is here to limit this burst effect and avoid bugs.

4.9.9 max-nb-cursors

`--max-nb-cursors=<value>` [Command-line option]
`LW6_MAX_NB_CURSORS` [Environment variable]
`max-nb-cursors` [XML key]

Type: integer.

Defines the maximum number of cursors who can enter the game. Really makes sense in network games. Default value is 26, the maximum.

4.9.10 max-nb-servers

`--max-nb-servers=<value>` [Command-line option]
`LW6_MAX_NB_SERVERS` [Environment variable]
`max-nb-servers` [XML key]

Type: integer.

Defines the maximum number of servers who can enter the game. Really makes sense in network games. Default value is 10, and should fit in most cases. Can be raised up to 26.

4.9.11 max-nb-teams

`--max-nb-teams=<value>` [Command-line option]
`LW6_MAX_NB_TEAMS` [Environment variable]
`max-nb-teams` [XML key]

Type: integer.

Defines the maximum number of teams who can enter the game. Really makes sense in network games. Default value is 10, the maximum.

4.9.12 max-round-delta

`--max-round-delta=<value>` [Command-line option]
`LW6_MAX_ROUND_DELTA` [Environment variable]
`max-round-delta` [XML key]

Type: integer.

This is the companion value of 'round-delta'. Will put an absolute limit to the delta, which (what did you think?) is of course incremented in some cases by the core algorithm. If in doubt, don't touch.

4.9.13 max-zone-size

`--max-zone-size=<value>` [Command-line option]
`LW6_MAX_ZONE_SIZE` [Environment variable]
`max-zone-size` [XML key]

Type: integer.

Defines the maximum zone size, which is an internal and rather technical parameter. The idea is that to optimize things, Liquid War 6 divides the battlefield in squares, where it can, and tries to make these squares as big as possible, the idea being that everywhere in this square, fighters follow the same intructions. Just a technical optimization. The problem is that setting it too high will reveal the optimization and its tradeoffs to the player, who will see the fighter behave strangely, following invisible paths. Plus, it's ugly. Depending on your tastes (speed, look'n'feel) you'll prefer something nice or something fast. Note that anyways passed a certain value, this does not optimize anything anymore. In doubt, don't touch it.

4.9.14 moves-per-round

`--moves-per-round=<value>` [Command-line option]

`LW6_MOVES_PER_ROUND` [Environment variable]
`moves-per-round` [XML key]

Type: integer.

Defines how many times fighters move per round. Increasing this will just make fighters move faster, but won't change anything for the rest, that is keyboard and mouse responsivity, and network traffic will stay the same. Multiplying the number of moves per round by the number of rounds per second will give the number of moves per second, which is, in fact, how fast fighters move on the screen.

4.9.15 nb-attack-tries

`--nb-attack-tries=<value>` [Command-line option]
`LW6_NB_ATTACK_TRIES` [Environment variable]
`nb-attack-tries` [XML key]

Type: integer.

Defines how many tries a fighter will do before giving-up attacking and choosing another behvior (defense). By tries we mean: how many directions it will try. Going North? Going North-West? Setting this to a low value will make fighters somewhat less aggressive. This idea is that they'll prefer to switch to the next option, that is, defense/regeneration, if there's no opponent right in front of them.

4.9.16 nb-defense-tries

`--nb-defense-tries=<value>` [Command-line option]
`LW6_NB_DEFENSE_TRIES` [Environment variable]
`nb-defense-tries` [XML key]

Type: integer.

Defines how many tries a fighter will do before giving-up attacking and choosing another behavior (do nothing). By tries we mean: how many directions it will try. Going North? Going North-West? Setting this to a low value, you'll need a very compact pack of fighters for regeneration to operate, else fighters will hang arround unhealthy.

4.9.17 nb-move-tries

`--nb-move-tries=<value>` [Command-line option]
`LW6_NB_MOVE_TRIES` [Environment variable]
`nb-move-tries` [XML key]

Type: integer.

Defines how many tries a fighter will do before giving-up moving and choosing another behvior (attack or defense). By tries we mean: how many directions it will try. Going North? Going North-West? Setting this to a low value, your fighters will look very stubborn and always try to move in one direction, neglecting the fact that they could dodge. This can lead to queues of fighters and other strange behaviors. On the other hand, setting it too high will cause fighter to always avoid the enemy, and groups of fighters will just pass each other without any fight. Matter of taste.

4.9.18 respawn-team

`--respawn-team=<value>` [Command-line option]
`LW6_RESPAWN_TEAM` [Environment variable]
`respawn-team` [XML key]

Type: integer.

Defines what to do when a team dies. If set to 0, team disappears forever, if set to 1, team reappears automatically with fresh fighters. It's a deathmatch mode, where the winner is not the one who stays alive the longest time, since it makes no real sens in this case, but the one who has died less often than others.

4.9.19 round-delta

`--round-delta=<value>` [Command-line option]
`LW6_ROUND_DELTA` [Environment variable]
`round-delta` [XML key]

Type: integer.

Conditions by how much the cursor potential will be incremented each time gradient is spreaded. Sounds cryptic? It is. The idea is that at each time you move your cursor of 1 pixel, theorically, you'll need in the worst case to move of 1 more pixel to reach any point on the map. Of course this is not true but this is the default asumption, and gradient spread will fix that. Only in Liquid War 6 this is not even the worst case, for you can control your cursor with the mouse and cross walls. Whenever you cross a wall, you might have done a great distance from the fighters' point of view, if the map is a maze. Thus this parameter, which corrects things, experience shows it does give acceptable results to increase the cursor potential by more than one at each turn. Toy arround with this if you find fighters take wrong paths on some given map. If in doubt, don't touch.

4.9.20 rounds-per-sec

`--rounds-per-sec=<value>` [Command-line option]
`LW6_ROUNDS_PER_SEC` [Environment variable]
`rounds-per-sec` [XML key]

Type: integer.

Defines the overall speed of the game. All other settings being equal, raising this value will cause the game to behave faster. Everything will be faster, except probably the display since your computer will calculate more game positions in a given time and spend more CPU time. It will also increase network traffic. Values between 10 and 50 really make sense.

4.9.21 side-attack-factor

`--side-attack-factor=<value>` [Command-line option]
`LW6_SIDE_ATTACK_FACTOR` [Environment variable]
`side-attack-factor` [XML key]

Type: integer.

Defines how hard fighters will attack sideways. It's an algorithm trick, fighters attack by default the opponent right in front, but if there's no fighter there, they will still try to attack someone else, maybe sideways. But doing this their attack is not as strong. This parameter enables you to tune this. This is a percentage.

4.9.22 side-defense-factor

`--side-defense-factor=<value>` [Command-line option]
`LW6_SIDE_DEFENSE_FACTOR` [Environment variable]
`side-defense-factor` [XML key]

Type: integer.

Defines how fast fighters will regenerate, when being side by side instead of being right in front of the other. This is a percentage.

4.9.23 single-army-size

`--single-army-size=<value>` [Command-line option]
`LW6_SINGLE_ARMY_SIZE` [Environment variable]
`single-army-size` [XML key]

Type: integer.

Defines the proportion of the whole available space, which will be occupied by an army at the beginning of the game. You can either imagine playing with almost empty maps, or play very crowded with almost no space left. This is a percentage, but will be multiplied by itself to get the actual surface. That is, 50 means 50%*50%, that is, a square of 1/2 the size of a square map, so it represents 25% (1/4) of the total surface.

4.9.24 spread-thread

`--spread-thread=<value>` [Command-line option]
`LW6_SPREAD_THREAD` [Environment variable]
`spread-thread` [XML key]

Type: integer.

If set to 1, the core algorithm with fire a separate thread to spread the gradient. By default this is turned off (set to 0). Consider this as an experimental feature, the program is already rather heavily threaded, turning this on will probably not offer any significant performance gain, even on SMP systems. This might change in the future.

4.9.25 spreads-per-round

`--spreads-per-round=<value>` [Command-line option]
`LW6_SPREADS_PER_ROUND` [Environment variable]
`spreads-per-round` [XML key]

Type: integer.

Defines how many times the gradient is spread per round. Gradient spread is a very Liquid War 6 specific feature, just remember that the more often you do it, the more accurately fighters will move. That is, you will be sure they really take the shortest

path. Usually this does not have much effect, the default value should fit in most cases, but you might want to decrease it on very simple maps where the gradient is obvious, or increase it on complex maps where you want fighters to be real smart.

4.9.26 start-blue-x

`--start-blue-x=<value>` [Command-line option]
`LW6_START_BLUE_X` [Environment variable]
`start-blue-x` [XML key]

Type: integer.

X start position for the blue team. This is a percentage of map width, value between 0 and 100.

4.9.27 start-blue-y

`--start-blue-y=<value>` [Command-line option]
`LW6_START_BLUE_Y` [Environment variable]
`start-blue-y` [XML key]

Type: integer.

Y start position for the blue team. This is a percentage of map height, value between 0 and 100.

4.9.28 start-cyan-x

`--start-cyan-x=<value>` [Command-line option]
`LW6_START_CYAN_X` [Environment variable]
`start-cyan-x` [XML key]

Type: integer.

X start position for the cyan team. This is a percentage of map width, value between 0 and 100.

4.9.29 start-cyan-y

`--start-cyan-y=<value>` [Command-line option]
`LW6_START_CYAN_Y` [Environment variable]
`start-cyan-y` [XML key]

Type: integer.

Y start position for the cyan team. This is a percentage of map height, value between 0 and 100.

4.9.30 start-green-x

`--start-green-x=<value>` [Command-line option]
`LW6_START_GREEN_X` [Environment variable]
`start-green-x` [XML key]

Type: integer.

X start position for the green team. This is a percentage of map width, value between 0 and 100.

4.9.31 start-green-y

`--start-green-y=<value>` [Command-line option]
`LW6_START_GREEN_Y` [Environment variable]
`start-green-y` [XML key]

Type: integer.

Y start position for the green team. This is a percentage of map height, value between 0 and 100.

4.9.32 start-lightblue-x

`--start-lightblue-x=<value>` [Command-line option]
`LW6_START_LIGHTBLUE_X` [Environment variable]
`start-lightblue-x` [XML key]

Type: integer.

X start position for the lightblue team. This is a percentage of map width, value between 0 and 100.

4.9.33 start-lightblue-y

`--start-lightblue-y=<value>` [Command-line option]
`LW6_START_LIGHTBLUE_Y` [Environment variable]
`start-lightblue-y` [XML key]

Type: integer.

Y start position for the lightblue team. This is a percentage of map height, value between 0 and 100.

4.9.34 start-magenta-x

`--start-magenta-x=<value>` [Command-line option]
`LW6_START_MAGENTA_X` [Environment variable]
`start-magenta-x` [XML key]

Type: integer.

X start position for the magenta team. This is a percentage of map width, value between 0 and 100.

4.9.35 start-magenta-y

`--start-magenta-y=<value>` [Command-line option]
`LW6_START_MAGENTA_Y` [Environment variable]
`start-magenta-y` [XML key]

Type: integer.

Y start position for the magenta team. This is a percentage of map height, value between 0 and 100.

4.9.36 start-orange-x

`--start-orange-x=<value>` [Command-line option]
`LW6_START_ORANGE_X` [Environment variable]
`start-orange-x` [XML key]

Type: integer.

X start position for the orange team. This is a percentage of map width, value between 0 and 100.

4.9.37 start-orange-y

`--start-orange-y=<value>` [Command-line option]
`LW6_START_ORANGE_Y` [Environment variable]
`start-orange-y` [XML key]

Type: integer.

Y start position for the orange team. This is a percentage of map height, value between 0 and 100.

4.9.38 start-pink-x

`--start-pink-x=<value>` [Command-line option]
`LW6_START_PINK_X` [Environment variable]
`start-pink-x` [XML key]

Type: integer.

X start position for the pink team. This is a percentage of map width, value between 0 and 100.

4.9.39 start-pink-y

`--start-pink-y=<value>` [Command-line option]
`LW6_START_PINK_Y` [Environment variable]
`start-pink-y` [XML key]

Type: integer.

Y start position for the pink team. This is a percentage of map height, value between 0 and 100.

4.9.40 start-position-mode

`--start-position-mode=<value>` [Command-line option]
`LW6_START_POSITION_MODE` [Environment variable]
`start-position-mode` [XML key]

Type: integer.

Defines how teams or set up on the map at game startup. 0, the default, means teams respect the pre-defined start positions. 1 means that a random position will be picked, among the existing positions. That is, red could take green's place. 2 means total randomness, teams can appear anywhere.

4.9.41 start-purple-x

`--start-purple-x=<value>` [Command-line option]
`LW6_START_PURPLE_X` [Environment variable]
`start-purple-x` [XML key]

Type: integer.

X start position for the purple team. This is a percentage of map width, value between 0 and 100.

4.9.42 start-purple-y

`--start-purple-y=<value>` [Command-line option]
`LW6_START_PURPLE_Y` [Environment variable]
`start-purple-y` [XML key]

Type: integer.

Y start position for the purple team. This is a percentage of map height, value between 0 and 100.

4.9.43 start-red-x

`--start-red-x=<value>` [Command-line option]
`LW6_START_RED_X` [Environment variable]
`start-red-x` [XML key]

Type: integer.

X start position for the red team. This is a percentage of map width, value between 0 and 100.

4.9.44 start-red-y

`--start-red-y=<value>` [Command-line option]
`LW6_START_RED_Y` [Environment variable]
`start-red-y` [XML key]

Type: integer.

Y start position for the red team. This is a percentage of map height, value between 0 and 100.

4.9.45 start-yellow-x

`--start-yellow-x=<value>` [Command-line option]
`LW6_START_YELLOW_X` [Environment variable]
`start-yellow-x` [XML key]

Type: integer.

X start position for the yellow team. This is a percentage of map width, value between 0 and 100.

4.9.46 start-yellow-y

`--start-yellow-y=<value>` [Command-line option]
`LW6_START_YELLOW_Y` [Environment variable]
`start-yellow-y` [XML key]

Type: integer.

Y start position for the yellow team. This is a percentage of map height, value between 0 and 100.

4.9.47 total-armies-size

`--total-armies-size=<value>` [Command-line option]
`LW6_TOTAL_ARMIES_SIZE` [Environment variable]
`total-armies-size` [XML key]

Type: integer.

Defines the proportion of the whole available space, which can be occupied by all the armies present together. Setting this low, whenever a new team arrives on the map, fighters might be stolen to other teams, otherwise the ame would get too crowded. This allows you to play with reasonnably enough fighters with 2 players, while still allowing interesting gameplay with many players. This is a percentage, but will be multiplied by itself to get the actual surface. That is, 50 means 50%*50%, that is, a square of 1/2 the size of a square map, so it represents 25% (1/4) of the total surface.

4.9.48 total-time

`--total-time=<value>` [Command-line option]
`LW6_TOTAL_TIME` [Environment variable]
`total-time` [XML key]

Type: integer.

Defines the maximum time of the game, in seconds. Note that in some cases, the game can end much earlier if some player has managed to win before the bell rings. Also, technically, this value will be translated into rounds and moves, and the game engine will wait until enough rounds and moves have been played. So if the computer is too slow and the desired speed is not reached, then the game will last for a longer time.

4.9.49 vertical-move

`--vertical-move=<value>` [Command-line option]
`LW6_VERTICAL_MOVE` [Environment variable]
`vertical-move` [XML key]

Type: integer.

Defines when to process a vertical move (along the Z 'depth' axis). If set to 0, fighters never spontaneously move along this axis. If set to 1, it will be tried just after the first move failed. If set to 2, it will be tried just after the second move failed. And so on.

4.9.50 x-polarity

`--x-polarity=<value>` [Command-line option]
`LW6_X_POLARITY` [Environment variable]
`x-polarity` [XML key]

Type: integer.

Defines how the map will be wrapped on the X (horizontal) axis. If set to 0, nothing is wrapped. If set to 1, the right and left borders are connected, any fighter can disappear on the right border and reappear on the left border, for instance. If set to -1, it will be wrapped but also inversed, that is on a 320x240 map, a fighter disappearing on the left border at position (0,60) will reapper on the right border at position (319,180). You can combine it with 'y-polarity'.

4.9.51 y-polarity

`--y-polarity=<value>` [Command-line option]
`LW6_Y_POLARITY` [Environment variable]
`y-polarity` [XML key]

Type: integer.

Defines how the map will be wrapped on the Y (vertical) axis. If set to 0, nothing is wrapped. If set to 1, the top and bottom borders are connected, any fighter can disappear on the top border and reappear on the bottom border, for instance. If set to -1, it will be wrapped but also inversed, that is on a 320x240 map, a fighter disappearing on the bottom border at position (40,239) will reapper on the top border at position (280,0). You can combine it with 'x-polarity'.

4.9.52 z-polarity

`--z-polarity=<value>` [Command-line option]
`LW6_Z_POLARITY` [Environment variable]
`z-polarity` [XML key]

Type: integer.

Defines how the map will be wrapped on the Z (deep) axis. If set to 0, nothing is wrapped. If set to 1, when using a 4 layer map, for instance, fighters on layer 1 will be able to go directly to layer 4 even if layers 2 and 3 are filled with walls. A value of -1 is forbidden, this is not like x and y axis, it does not really make sense. Consider this an advanced setting which might save a layer in some tricky cases, the default value of 0 should fit in most cases.

4.10 Map hints.xml

4.10.1 background-color-auto

`--background-color-auto=<value>` [Command-line option]
`LW6_BACKGROUND_COLOR_AUTO` [Environment variable]
`background-color-auto` [XML key]

Type: boolean.

Defines wether hud colors will be set automatically from base and alternate colors. This is a time saver to keep map designers from requiring to redefined every single color in the game. You only need to set color-base-bg, color-base-fg, color-alternate-bg and color-alternate-fg. Then hud_color_frame_bg, hud_color_frame_fg, hud_color_text_bg and hud_color_text_fg will be automatically set.

4.10.2 fighter-scale

`--fighter-scale=<value>` [Command-line option]
`LW6_FIGHTER_SCALE` [Environment variable]
`fighter-scale` [XML key]

Type: float.

Defines how wide (in pixels) fighters must be. This parameter is very important and will largely condition the number of fighters on the map. It is used when loading the map. If it is, for instance, set to 1, there will be exactly a fighter per pixel on the screen. That is, if you play 640x480 on an empty map, the maximum fighters you could have is about 300000. The idea is that by changing the resolution, you also define the density of the map. In pratice, this is done in the hope that someone with a slow computer will pick up a low resolution and therefore play small levels. Conversely, someone with a brand new computer with powerfull CPU & GPU will use great resolutions and be happy with many fighters on the map. Still, changing the resolution after loading the map will not affet the number of fighters. Same for network games, the first player, who loads the map, defines its properties according to its own settings.

4.10.3 guess-colors

`--guess-colors=<value>` [Command-line option]
`LW6_GUESS_COLORS` [Environment variable]
`guess-colors` [XML key]

Type: boolean.

Defines wether colors should be set automatically from texture colors. If set to true, then the program will try to pick up colors automatically from the texture, and will override the values of the color-base-bg, color-base-fg, color-alternate-bg and color-alternate-fg parameters. How these colors are picked up can't be garanteed, so if the map does not have strong contrast or if there can be any form of ambiguity, it's safe to set this to false and define one's own colors.

4.10.4 hud-color-auto

`--hud-color-auto=<value>` [Command-line option]
`LW6_HUD_COLOR_AUTO` [Environment variable]
`hud-color-auto` [XML key]

Type: boolean.

Defines wether hud colors will be set automatically from base and alternate colors. This is a time saver to keep map designers from requiring to redefined every single color in the game. You only need to set color-base-bg, color-base-fg, color-alternate-bg and

color-alternate-fg. Then hud_color_frame_bg, hud_color_frame_fg, hud_color_text_bg and hud_color_text_fg will be automatically set.

4.10.5 max-map-height

`--max-map-height=<value>` [Command-line option]
`LW6_MAX_MAP_HEIGHT` [Environment variable]
`max-map-height` [XML key]

Type: integer.

Allows you to give a maximum map height. When designing a map you might wonder: this is dumb I'm conceiving this map I know its height, why should I limit it? Now think of the play who plays on a old slowish computer with a tiny screen. He might redefine this himself, and does not necessarly wishes to fire Gimp to rescale the map.

4.10.6 max-map-surface

`--max-map-surface=<value>` [Command-line option]
`LW6_MAX_MAP_SURFACE` [Environment variable]
`max-map-surface` [XML key]

Type: integer.

Allows you to give a maximum map surface. Map surface is simply (width * height). This parameter is just here to save you the hassle of defining both 'max-map-width' and 'max-map-height' in a consistent manner.

4.10.7 max-map-width

`--max-map-width=<value>` [Command-line option]
`LW6_MAX_MAP_WIDTH` [Environment variable]
`max-map-width` [XML key]

Type: integer.

Allows you to give a maximum map width. When designing a map you might wonder: this is dumb I'm conceiving this map I know its width, why should I limit it? Now think of the play who plays on a old slowish computer with a tiny screen. He might redefine this himself, and does not necessarly wishes to fire Gimp to rescale the map.

4.10.8 menu-color-auto

`--menu-color-auto=<value>` [Command-line option]
`LW6_MENU_COLOR_AUTO` [Environment variable]
`menu-color-auto` [XML key]

Type: boolean.

Defines wether menu colors will be set automatically from base and alternate colors. This is a time saver to keep map designers from requiring to redefined every single color in the game. You only need to set color-base-bg, color-base-fg, color-alternate-bg and color-alternate-fg. Then menu_color_default_bg, menu_color_default_fg, menu_color_selected_bg, menu_color_selected_fg, menu_color_disabled_bg and menu_color_disabled_fg will be automatically set.

4.10.9 min-map-height

`--min-map-height=<value>` [Command-line option]
`LW6_MIN_MAP_HEIGHT` [Environment variable]
`min-map-height` [XML key]

Type: integer.

Allows you to give a minimum map height. When designing a map you might wonder: this is dumb I'm conceiving this map I know its height, why should I limit it? Now think of the player who decided to play with highly-defined maps because he has a super calculator and a hudge screen. He might redefine this himself, and does not necessarly wishes to fire Gimp to rescale the map.

4.10.10 min-map-surface

`--min-map-surface=<value>` [Command-line option]
`LW6_MIN_MAP_SURFACE` [Environment variable]
`min-map-surface` [XML key]

Type: integer.

Allows you to give a minimum map surface. Map surface is simply (width * height). This parameter is just here to save you the hassle of defining both 'min-map-width' and 'min-map-height' in a consistent manner.

4.10.11 min-map-width

`--min-map-width=<value>` [Command-line option]
`LW6_MIN_MAP_WIDTH` [Environment variable]
`min-map-width` [XML key]

Type: integer.

Allows you to give a minimum map width. When designing a map you might wonder: this is dumb I'm conceiving this map I know its width, why should I limit it? Now think of the player who decided to play with highly-defined maps because he has a super calculator and a hudge screen. He might redefine this himself, and does not necessarly wishes to fire Gimp to rescale the map.

4.10.12 resample

`--resample=<value>` [Command-line option]
`LW6_RESAMPLE` [Environment variable]
`resample` [XML key]

Type: boolean.

If set to true, maps will always be resampled to a size which depends on your screen resolution, zoom factor, and the rest. If false, maps will be set at the exact resolution of map.png.

4.10.13 system-color-auto

`--system-color-auto=<value>` [Command-line option]
`LW6_SYSTEM_COLOR_AUTO` [Environment variable]

`system-color-auto` [XML key]

Type: boolean.

Defines wether system colors will be set automatically from base and alternate colors. This is a time saver to keep map designers from requiring to redefined every single color in the game. You only need to set color-base-bg, color-base-fg, color-alternate-bg and color-alternate-fg. Then system_color_bg and system_color_fg will be automatically set.

4.10.14 view-color-auto

`--view-color-auto=<value>` [Command-line option]
`LW6_VIEW_COLOR_AUTO` [Environment variable]
`view-color-auto` [XML key]

Type: boolean.

Defines wether view colors will be set automatically from base and alternate colors. This is a time saver to keep map designers from requiring to redefined every single color in the game. You only need to set color-base-bg, color-base-fg, color-alternate-bg and color-alternate-fg. Then view_color_cursor_bg, view_color_cursor_fg, view_color_map_bg and view_color_map_fg will be automatically set.

4.11 Map style.xml

4.11.1 animation-density

`--animation-density=<value>` [Command-line option]
`LW6_ANIMATION_DENSITY` [Environment variable]
`animation-density` [XML key]

Type: float.

Density of the background animation, that is, for instance, if the background animation is about displaying bubbles, using a high value will display many bubbles. A value of 1.0 corresponds to the default setting.

4.11.2 animation-speed

`--animation-speed=<value>` [Command-line option]
`LW6_ANIMATION_SPEED` [Environment variable]
`animation-speed` [XML key]

Type: float.

Speed of the background animation, that is, for instance, if the background animation is about displaying bubbles, using a high value will cause bubbles to move very fast. A value of 1.0 corresponds to the default setting.

4.11.3 background-color-root-bg

`--background-color-root-bg=<value>` [Command-line option]
`LW6_BACKGROUND_COLOR_ROOT_BG` [Environment variable]
`background-color-root-bg` [XML key]

Type: color.

Defines the main background color. This is, for instance, the color which will be used to clear the screen before drawing thing. Will be automatically guessed from the map texture if color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.4 background-color-root-fg

`--background-color-root-fg=<value>` [Command-line option]
`LW6_BACKGROUND_COLOR_ROOT_FG` [Environment variable]
`background-color-root-fg` [XML key]

Type: color.

Defines a color which will be used together with color-base-bg to compose the background. It can be wise to have a minimum contrast between this color and color-base-bg, but it is not mandatory, especially if other colors are manually redefined. Will be automatically guessed from the map texture if color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.5 background-color-stuff-bg

`--background-color-stuff-bg=<value>` [Command-line option]
`LW6_BACKGROUND_COLOR_STUFF_BG` [Environment variable]
`background-color-stuff-bg` [XML key]

Type: color.

Defines a color which will be used together with color-alternate-fg to draw things (animations, sprites, text, whatever) in the background. It should be different enough from color-alternate-fg so that one can really distinguish these colors. Will be automatically guessed from the map texture if color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.6 background-color-stuff-fg

`--background-color-stuff-fg=<value>` [Command-line option]
`LW6_BACKGROUND_COLOR_STUFF_FG` [Environment variable]
`background-color-stuff-fg` [XML key]

Type: color.

Defines a color which will be used to draw things (animations, sprites, text, whatever) in the background. It should be different enough from color-alternate-bg so that one can really distinguish these colors. Think of this as the sprite, the text, the whatever-needs-to-be-seen-uses-this color. Will be automatically guessed from the map texture if color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.7 background-style

`--background-style=<value>` [Command-line option]
`LW6_BACKGROUND_STYLE` [Environment variable]
`background-style` [XML key]

Type: string.

The background defines, of course, what is displayed at the background, but it also conditions the colors used for other items, such as the menus for instance. The only possible value for now is 'bubbles'.

4.11.8 color-alternate-bg

`--color-alternate-bg=<value>` [Command-line option]
`LW6_COLOR_ALTERNATE_BG` [Environment variable]
`color-alternate-bg` [XML key]

Type: color.

Defines the alternate color, more precisely, its bg (background) part. Colors are always defined by a bg/fg pair. Most colors in the game can be deduced from this one, usually to color a map you only need to define color-base-bg, color-base-fg, color-alternate-bg and color-alternate-fg.

4.11.9 color-alternate-fg

`--color-alternate-fg=<value>` [Command-line option]
`LW6_COLOR_ALTERNATE_FG` [Environment variable]
`color-alternate-fg` [XML key]

Type: color.

Defines the alternate color, more precisely, its fg (foreground) part. Colors are always defined by a bg/fg pair. Most colors in the game can be deduced from this one, usually to color a map you only need to define color-base-bg, color-base-fg, color-alternate-bg and color-alternate-fg.

4.11.10 color-base-bg

`--color-base-bg=<value>` [Command-line option]
`LW6_COLOR_BASE_BG` [Environment variable]
`color-base-bg` [XML key]

Type: color.

Defines the base color, more precisely, its bg (background) part. Colors are always defined by a bg/fg pair. Most colors in the game can be deduced from this one, usually to color a map you only need to define color-base-bg, color-base-fg, color-alternate-bg and color-alternate-fg.

4.11.11 color-base-fg

`--color-base-fg=<value>` [Command-line option]
`LW6_COLOR_BASE_FG` [Environment variable]
`color-base-fg` [XML key]

Type: color.

Defines the base color, more precisely, its fg (foreground) part. Colors are always defined by a bg/fg pair. Most colors in the game can be deduced from this one, usually to color a map you only need to define color-base-bg, color-base-fg, color-alternate-bg and color-alternate-fg.

4.11.12 colorize

`--colorize=<value>` [Command-line option]
`LW6_COLORIZE` [Environment variable]

`colorize` [XML key]

Type: boolean.

If set, then all background drawings including textures will use the bakground colors. This means, for instance, that if background colors are set automatically by color-auto from the map texture, then the background will adopt the same range of colors than the map itself. In short, the background will mimic the map.

4.11.13 cursor-size

`--cursor-size=<value>` [Command-line option]
`LW6_CURSOR_SIZE` [Environment variable]
`cursor-size` [XML key]

Type: float.

Size of the cursors on the map. 1 is the default, setting it to a higher value will make cursors bigger, a lower value will make them smaller.

4.11.14 hidden-layer-alpha

`--hidden-layer-alpha=<value>` [Command-line option]
`LW6_HIDDEN_LAYER_ALPHA` [Environment variable]
`hidden-layer-alpha` [XML key]

Type: float.

Whenever players are supposed to be hidden behind a wall, for instance if they are in layer 2 and layer 1 is filled with walls, it's still possible to see them, but with a low alpha value (almost transparent). This parameter allows you to trick this value, 0 will make these players absolutely invisible, 1 will make them totally opaque, like if they were on layer 1.

4.11.15 hud-color-frame-bg

`--hud-color-frame-bg=<value>` [Command-line option]
`LW6_HUD_COLOR_FRAME_BG` [Environment variable]
`hud-color-frame-bg` [XML key]

Type: color.

Defines the background color for the hud frame. Ignored if hud-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.16 hud-color-frame-fg

`--hud-color-frame-fg=<value>` [Command-line option]
`LW6_HUD_COLOR_FRAME_FG` [Environment variable]
`hud-color-frame-fg` [XML key]

Type: color.

Defines the foreground color for the hud frame. Ignored if hud-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.17 hud-color-text-bg

`--hud-color-text-bg=<value>` [Command-line option]
`LW6_HUD_COLOR_TEXT_BG` [Environment variable]
`hud-color-text-bg` [XML key]

Type: color.

Defines the background color for hud text. Ignored if hud-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.18 hud-color-text-fg

`--hud-color-text-fg=<value>` [Command-line option]
`LW6_HUD_COLOR_TEXT_FG` [Environment variable]
`hud-color-text-fg` [XML key]

Type: color.

Defines the foreground color for hud text. Ignored if hud-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.19 hud-style

`--hud-style=<value>` [Command-line option]
`LW6_HUD_STYLE` [Environment variable]
`hud-style` [XML key]

Type: string.

The hud is where informations about the game are displayed. This means, who is winning, are other status-like informations. Possible values include 'floating' and 'tactical'.

4.11.20 keep-ratio

`--keep-ratio=<value>` [Command-line option]
`LW6_KEEP_RATIO` [Environment variable]
`keep-ratio` [XML key]

Type: boolean.

Defines wether the map should keep its ratio, or if it should be stretched to fill the shape of your screen.

4.11.21 menu-color-default-bg

`--menu-color-default-bg=<value>` [Command-line option]
`LW6_MENU_COLOR_DEFAULT_BG` [Environment variable]
`menu-color-default-bg` [XML key]

Type: color.

Defines the default background color for menus. Ignored if menu-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.22 menu-color-default-fg

`--menu-color-default-fg=<value>` [Command-line option]
`LW6_MENU_COLOR_DEFAULT_FG` [Environment variable]
`menu-color-default-fg` [XML key]

Type: color.

Defines the default foreground color for menus. In fact, this is the main color for menu text, the color used to draw letters in menus. Ignored if menu-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.23 menu-color-disabled-bg

`--menu-color-disabled-bg=<value>` [Command-line option]
`LW6_MENU_COLOR_DISABLED_BG` [Environment variable]
`menu-color-disabled-bg` [XML key]

Type: color.

Defines the background color for a disabled menu item. Ignored if menu-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.24 menu-color-disabled-fg

`--menu-color-disabled-fg=<value>` [Command-line option]
`LW6_MENU_COLOR_DISABLED_FG` [Environment variable]
`menu-color-disabled-fg` [XML key]

Type: color.

Defines the foreground color for a disabled menu item. Ignored if menu-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.25 menu-color-selected-bg

`--menu-color-selected-bg=<value>` [Command-line option]
`LW6_MENU_COLOR_SELECTED_BG` [Environment variable]
`menu-color-selected-bg` [XML key]

Type: color.

Defines the background color for a selected menu item. Ignored if menu-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.26 menu-color-selected-fg

`--menu-color-selected-fg=<value>` [Command-line option]
`LW6_MENU_COLOR_SELECTED_FG` [Environment variable]
`menu-color-selected-fg` [XML key]

Type: color.

Defines the foreground color for a selected menu item. Ignored if menu-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.27 menu-style

`--menu-style=<value>` [Command-line option]
`LW6_MENU_STYLE` [Environment variable]
`menu-style` [XML key]

Type: string.

The menu style is simply the name of the engine used to power the menu system. The only possible value, for now, is 'cylinder'.

4.11.28 system-color-bg

`--system-color-bg=<value>` [Command-line option]
`LW6_SYSTEM_COLOR_BG` [Environment variable]
`system-color-bg` [XML key]

Type: color.

Defines the system background color, used when displaying system info, such as the number of frames per second. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.29 system-color-fg

`--system-color-fg=<value>` [Command-line option]
`LW6_SYSTEM_COLOR_FG` [Environment variable]
`system-color-fg` [XML key]

Type: color.

Defines the system foreground color, used when displaying system info, such as the number of frames per second. This will typically be text color. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.30 team-color-blue

`--team-color-blue=<value>` [Command-line option]
`LW6_TEAM_COLOR_BLUE` [Environment variable]
`team-color-blue` [XML key]

Type: color.

Defines the color for the blue team. Syntax is HTML-like, #RGB or #RRGGBB.

4.11.31 team-color-cyan

`--team-color-cyan=<value>` [Command-line option]
`LW6_TEAM_COLOR_CYAN` [Environment variable]
`team-color-cyan` [XML key]

Type: color.

Defines the color for the cyan team. Syntax is HTML-like, #RGB or #RRGGBB.

4.11.32 team-color-green

`--team-color-green=<value>` [Command-line option]
`LW6_TEAM_COLOR_GREEN` [Environment variable]
`team-color-green` [XML key]

Type: color.

Defines the color for the green team. Syntax is HTML-like, #RGB or #RRGGBB.

4.11.33 team-color-lightblue

`--team-color-lightblue=<value>` [Command-line option]
`LW6_TEAM_COLOR_LIGHTBLUE` [Environment variable]
`team-color-lightblue` [XML key]

Type: color.

Defines the color for the light blue team. Syntax is HTML-like, #RGB or #RRGGBB.

4.11.34 team-color-magenta

`--team-color-magenta=<value>` [Command-line option]
`LW6_TEAM_COLOR_MAGENTA` [Environment variable]
`team-color-magenta` [XML key]

Type: color.

Defines the color for the magenta team. Syntax is HTML-like, #RGB or #RRGGBB.

4.11.35 team-color-orange

`--team-color-orange=<value>` [Command-line option]
`LW6_TEAM_COLOR_ORANGE` [Environment variable]
`team-color-orange` [XML key]

Type: color.

Defines the color for the orange team. Syntax is HTML-like, #RGB or #RRGGBB.

4.11.36 team-color-pink

`--team-color-pink=<value>` [Command-line option]
`LW6_TEAM_COLOR_PINK` [Environment variable]
`team-color-pink` [XML key]

Type: color.

Defines the color for the pink team. Syntax is HTML-like, #RGB or #RRGGBB.

4.11.37 team-color-purple

`--team-color-purple=<value>` [Command-line option]
`LW6_TEAM_COLOR_PURPLE` [Environment variable]
`team-color-purple` [XML key]

Type: color.

Defines the color for the purple team. Syntax is HTML-like, #RGB or #RRGGBB.

4.11.38 team-color-red

`--team-color-red=<value>` [Command-line option]
`LW6_TEAM_COLOR_RED` [Environment variable]
`team-color-red` [XML key]

Type: color.

Defines the color for the red team. Syntax is HTML-like, #RGB or #RRGGBB.

4.11.39 team-color-yellow

`--team-color-yellow=<value>` [Command-line option]
`LW6_TEAM_COLOR_YELLOW` [Environment variable]
`team-color-yellow` [XML key]

Type: color.

Defines the color for the yellow team. Syntax is HTML-like, #RGB or #RRGGBB.

4.11.40 view-color-cursor-bg

`--view-color-cursor-bg=<value>` [Command-line option]
`LW6_VIEW_COLOR_CURSOR_BG` [Environment variable]
`view-color-cursor-bg` [XML key]

Type: color.

Defines the background cursor color. Will typically be used to draw the shape of the cursor. Ignored if view-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.41 view-color-cursor-fg

`--view-color-cursor-fg=<value>` [Command-line option]
`LW6_VIEW_COLOR_CURSOR_FG` [Environment variable]
`view-color-cursor-fg` [XML key]

Type: color.

Defines the foreground cursor color. Will typically be used to draw text in the cursor. Ignored if view-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.42 view-color-map-bg

`--view-color-map-bg=<value>` [Command-line option]
`LW6_VIEW_COLOR_MAP_BG` [Environment variable]
`view-color-map-bg` [XML key]

Type: color.

Defines the background map color. If there's no map texture defined or if use-texture is false, this is the color of the places where armies will go. Ignored if view-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.43 view-color-map-fg

`--view-color-map-fg=<value>` [Command-line option]
`LW6_VIEW_COLOR_MAP_FG` [Environment variable]
`view-color-map-fg` [XML key]

Type: color.

Defines the foreground map color. If there's no map texture defined or if use-texture is false, this is the color of walls, what armies can't go through. Ignored if view-color-auto is set. Can be #RGB, #RGBA, #RRGGBB or #RRGGBBAA.

4.11.44 view-style

`--view-style=<value>` [Command-line option]
`LW6_VIEW_STYLE` [Environment variable]
`view-style` [XML key]

Type: string.

The view style conditions which renderer is used for the map, the area where fighters are displayed. This is not the graphics backend. Indeed, the graphics backend defines which technical tool one uses (which library) one runs, wether this parameter says what kind of rendering one wants.

4.11.45 zoom

`--zoom=<value>` [Command-line option]
`LW6_ZOOM` [Environment variable]
`zoom` [XML key]

Type: float.

Defines the map zoom. If lower than 1.0, map will occupy only a fraction of the screen, if greater than 1.0, some areas will be outside the screen, and the player will need to scroll through it.

4.12 Advanced settings

4.12.1 audit

`--audit` [Command-line option]
`LW6_AUDIT` [Environment variable]

Display all path values, defaults and current settings. This output is very usefull to track down problems such as missing directories, broken installations. If you get an error message that suggests some file is missing, then give this option a try.

4.12.2 bench

`--bench` [Command-line option]
`LW6_BENCH` [Environment variable]

Runs a benchmarking test which will report an approximative performance estimation of the game on your computer.

4.12.3 bot-iq

`--bot-iq=<value>` [Command-line option]
`LW6_BOT_IQ` [Environment variable]
`bot-iq` [XML key]

Type: integer.

The IQ (intelligence quotient) of bots. Typically, a value of 100 will make the bot behave normally, performing at its best. A value of 0 will just make it act the worst way it can. Values over 100 probably won't change anything compared to 100, but this truely depends on which bot backend you're running.

4.12.4 bot-speed

`--bot-speed=<value>` [Command-line option]
`LW6_BOT_SPEED` [Environment variable]
`bot-speed` [XML key]

Type: float.

The speed of bots, 1 means normal speed, higher value will speed it up, lower will slow it down. Note that this only has an impact on bot engines, not on the game speed itself.

4.12.5 checkpoint-period

`--checkpoint-period=<value>` [Command-line option]
`LW6_CHECKPOINT_PERIOD` [Environment variable]
`checkpoint-period` [XML key]

Type: integer.

Defines with what period (in msec) system values such as frames per sec, network traffic, and others, will be updated. Default value should fit in most cases.

4.12.6 commands-per-sec

`--commands-per-sec=<value>` [Command-line option]
`LW6_COMMANDS_PER_SEC` [Environment variable]
`commands-per-sec` [XML key]

Type: integer.

Defines the number of commands per second. When a command is generated, orders are actually sent to the game engine, for instance, 'this cursor moved there'. So this option will affect game responsiveness, setting this to a high value will make the game more responsive but consume bandwidth on network games.

4.12.7 demo

`--demo` [Command-line option]
`LW6_DEMO` [Environment variable]

Start the game in demo mode. 2 bots play against each other forever.

4.12.8 display-console

`--display-console=<value>` [Command-line option]
`LW6_DISPLAY_CONSOLE` [Environment variable]
`display-console` [XML key]

Type: boolean.

Defines wether the interactive system console must be displayed. Note that console support must have been enabled at compilation time. It might not be available on your computer, for instance if you are running a system such as Microsoft Windows.

4.12.9 display-fps

`--display-fps=<value>` [Command-line option]
`LW6_DISPLAY_FPS` [Environment variable]
`display-fps` [XML key]

Type: boolean.

Set this to 'true' to display the number of frames per second. When this gets too low... play a smaller map, buy a new computer or contribute and hack Liquid War 6 so that it runs faster!

4.12.10 frames-per-sec

`--frames-per-sec=<value>` [Command-line option]
`LW6_FRAMES_PER_SEC` [Environment variable]
`frames-per-sec` [XML key]

Type: integer.

Defines how many frames will be displayed per second. Of course this is a maximum value, if your hardware can't keep up with this value, display will just be slow, no matter what value you define here. Note that you might really wish to have something rather low here, to keep network and 'logic' function responsiveness. Passed 60 frames per second, speed is really only for visual comfort, as Liquid War 6 is now so fast-paced that it requires 200 frames/sec to outperform opponents.

4.12.11 io-per-sec

`--io-per-sec=<value>` [Command-line option]
`LW6_IO_PER_SEC` [Environment variable]
`io-per-sec` [XML key]

Type: integer.

Defines the number of calls to input/output functions per second. This can affect speed of menus but also cursors, but won't change the speed of the game itself. It's a cosmectic, comfort option.

4.12.12 loader-sleep

`--loader-sleep=<value>` [Command-line option]
`LW6_LOADER_SLEEP` [Environment variable]

`loader-sleep` [XML key]

Type: float.

Defines how long the loader thread should wait between two polls. Default value should fit in most cases.

4.12.13 log-level

`--log-level=<value>` [Command-line option]
`LW6_LOG_LEVEL` [Environment variable]
`log-level` [XML key]

Type: integer.

Defines the log level, that is, how verbose the program will be regarding logs and console output. 0 (ERROR) is the minimum, only errors are reported. 1 (WARNING) means errors + warnings. 2 (NOTICE) displays most important messages. 3 (INFO) is the default, the log file will contain all messages but debug stuff. 4 (DEBUG) logs everything, including debug informations.

4.12.14 memory-bazooka-eraser

`--memory-bazooka-eraser=<value>` [Command-line option]
`LW6_MEMORY_BAZOOKA_ERASER` [Environment variable]
`memory-bazooka-eraser` [XML key]

Type: boolean.

The memory eraser is a tool which will systematically fill allocated memory with 'M', and overwrite all allocated bytes with 'F' before freeing memory. It will even handle realloc calls. This is usefull to track bugs. Indeed, with this option enabled, freshly allocated memory will never contain zeroes unless one calls calloc, and if you ever free some memory zone before being done with it, it will be filled with junk and therefore not be usable. The memory bazooka must be big enough if you want this feature to actually work.

4.12.15 memory-bazooka-size

`--memory-bazooka-size=<value>` [Command-line option]
`LW6_MEMORY_BAZOOKA_SIZE` [Environment variable]
`memory-bazooka-size` [XML key]

Type: integer.

The memory bazooka is a brute-force tool, conceived after a full night spent tracking some memory leak. The idea is to keep a track of all allocated pointers, when the data was allocated (timestamp), where in the code (file, line), and even point out what data there is in that place. A memory bazooka report at the end of the game will just show what's left. There should be nothing. This parameter is here to avoid wasting CPU cycles on a feature which is very debug-oriented and does not really make sense for the casual user. Set it to 0 for best performance, something like 100 might just be helpfull, but 1000000 is the right way to seriously debug code.

4.12.16 modules

`--modules` [Command-line option]
`LW6_MODULES` [Environment variable]

Tells which modules have been enabled when the game was compiled. It's still possible to add or remove modules afterwards, but this option allows you to know how things were at first.

4.12.17 pilot-lag

`--pilot-lag=<value>` [Command-line option]
`LW6_PILOT_LAG` [Environment variable]
`pilot-lag` [XML key]

Type: integer.

Maximum lag, in rounds, until the game engine is slowed down. This will typically be usefull if your computer is too slow for the map resolution and the game speed you set up.

4.12.18 pilot-sleep

`--pilot-sleep=<value>` [Command-line option]
`LW6_PILOT_SLEEP` [Environment variable]
`pilot-sleep` [XML key]

Type: float.

Defines how long the pilot thread should wait between two polls. Default value should fit in most cases.

4.12.19 quick-start

`--quick-start` [Command-line option]
`LW6_QUICK_START` [Environment variable]

Start the game just like if the player had requested a quick start, without showing any menu.

4.12.20 server

`--server` [Command-line option]
`LW6_SERVER` [Environment variable]

Start the game in server mode, without requiring any graphics backend.

4.12.21 target

`--target` [Command-line option]
`LW6_TARGET` [Environment variable]

Display all known system target properties, including os and cpu informations.

4.13 Script hooks

4.14 C to Guile API

4.15 C functions

4.15.1 libbot

4.15.2 libcfg

`int lw6cfg_parse_command_line (void * context)` [Function]

context: opaque pointer on a context

Overwrites any existing option with command line args

Return value: 1 if success, 0 if error

`int lw6cfg_merge_env (void * cfg_context)` [Function]

cfg_context: a context returned by `lw6cfg_init`

Overwrites any existing vale in the config with environment variables prefixed by LW6_.

Return value: 1 if successfull, 0 if error.

`char * lw6cfg_format (char * key, char * value, lw6hlp_type_t type)` [Function]

key: the key of the value to format

value: the value to format

type: the type of the value to format

Formats, converts, a given value to its cannonical representation. Booleans will be converted to true/false, strings containing integers will be stripped from junk, and so on. This is a performance killer but will ensure everything is correct.

Return value: a newly allocated string, containing the same as the input, but reformatted the pedantic way.

`char * lw6cfg_format_guess_type (char * key, char * value)` [Function]

key: the key of the value to format

value: the value to format

Formats, converts, a given value to its cannonical representation. Booleans will be converted to true/false, strings containing integers will be stripped from junk, and so on. This is a performance killer but will ensure everything is correct. This function will automatically guess the type of the value from its description in the help system.

Return value: a newly allocated string, containing the same as the input, but reformatted the pedantic way.

`int lw6cfg_load (void * cfg_context, char * filename)` [Function]

cfg_context: a context returned by `lw6cfg_init`

filename: a file path, absolute or relative

Loads the given config file, and stores its values into the current context. Parameters which are both in the config file and given as command line parameters, will be taken from the command-line.

Return value: 1 if successfull, 0 if error.

`int lw6cfg_save` (*void * `cfg_context`, char * `filename`*) [Function]

cfg_context: a context returned by `lw6cfg_init`

filename: a file path, absolute or relative

Save current options into the given config file. Before saving the file, all command line arguments will be read and will override current values. This means the saved file will contain values given as command line arguments.

Return value: 1 if successfull, 0 if error.

`void * lw6cfg_init` (*int `argc`, char * [] `argv`*) [Function]

argc: number of command line arguments, as given to `main`

argv: a list of command line arguments, as given to `main`

Initializes a config context object. This object is hidden behind an opaque void * pointer to avoid direct access to its elements.

Return value: an opaque pointer, must be freed with `lw6cfg_quit`.

`void lw6cfg_quit` (*void * `cfg_context`*) [Function]

cfg_context: a context returned by `lw6cfg_init`

Frees a config cfg_context object. You must call this once you're done with the context.

Return value: none.

`void lw6cfg_reset` (*int `argc`, char * [] `argv`*) [Function]

argc: number of command line arguments, as given to `main`

argv: a list of command line arguments, as given to `main`

Overwrites the config file with defaults. Use this to get rid of old configurations.

`char * lw6cfg_unified_get_value` (*int `argc`, char * [] `argv`, char * `key`*) [Function]

argc: number of command-line args, as passed to `main`

argv: arry of command-line args, as passed to `main`

key: the key to query

Unified "value" getter, which gets informations from environment variables, command line, and config file. The rules is that the command-line argument always has the last word. It will override any other value. Follows environment variables, which will be used if no command-line argument is supplied. Note that these are "LW6_" prefixed and uppercased environment variables as opposed to lowercased and "dash-separated" keys. Finally, if there's no environment variable, nor any config-file corresponding entry, the value will be searched in the config file. If there's no information in the config file, NULL is returned.

Return value: a string with the value. Can be NULL. Must be freed.

`char * lw6cfg_unified_get_user_dir` (*int `argc`, char * [] `argv`*) [Function]

argc: number of command-line args, as passed to `main`

argv: arry of command-line args, as passed to `main`

Gets the user dir, taking all parameters in account, that's to say the "LW6_USER_DIR" env value, the "–user-dir" command-line paramater and the LW6DEF_USER_DIR config file entry.

Return value: the directory path, might be NULL, must be freed.

`char * lw6cfg_unified_get_log_file` (*int* `argc`, *char * [] `argv`*) [Function]

argc: number of command-line args, as passed to `main`

argv: arry of command-line args, as passed to `main`

Gets the user dir, taking all parameters in account, that's to say the "LW6_LOG_FILE" env value, the "–log-file" command-line paramater and the LW6DEF_LOG_FILE config file entry.

Return value: the directory path, might be NULL, must be freed.

`char * lw6cfg_unified_get_map_path` (*int* `argc`, *char * [] `argv`*) [Function]

argc: number of command-line args, as passed to `main`

argv: arry of command-line args, as passed to `main`

Gets the user dir, taking all parameters in account, that's to say the "LW6_MAP_PATH" env value, the "–map-path" command-line paramater and the LW6DEF_MAP_PATH config file entry.

Return value: the directory path, might be NULL, must be freed.

4.15.3 libcli

4.15.4 libcns

4.15.5 libdyn

`lw6dyn_dl_handle_t * lw6dyn_dlopen_backend_so` (*char * `so_file`*) [Function]

Opens a .so file directly, using a valid (full) path name.

Return value: a handle to the module, once it's opened. You might still need to call a module specific `init`() function, but it's another story.

`lw6dyn_dl_handle_t * lw6dyn_dlopen_backend` (*int `argc`, char * [] `argv`, char * `top_level_lib`, char * `backend_name`*) [Function]

argc: the number of command-line arguments as passed to `main`

top_level_lib: the top-level library concerned, this means is it "cli", "gfx", "snd" or "srv". This will tell the function to search for the .so file in the correct subdirectory. Think of this as a category.

Opens a .so file corresponding to the given backend, it is capable to search for system libraries installed after "make install" but if not found, it will also search in the current build directory, finding the .so files in hidden .libs subdirectories.

Return value: a handle to the module, once it's opened. You might still need to call a module specific `init`() function, but it's another story.

`int lw6dyn_dlclose_backend (`*lw6dyn_dl_handle_t* * **`handle`**`)` [Function]

handle: the backend to close.

Closes an opened backend. Note that you must call any backend specific clear, destroy, deinit, exit, function before.

Return value: 1 if success, 0 on error.

`void * lw6dyn_dlsym (`*lw6dyn_dl_handle_t* * **`handle`**`,` *char* * **`func_name`**`)` [Function]

handle: the backend concerned

func_name: the function name, as a NULL terminated string

Finds a C function in the given backend.

Return value: a pointer to the function, NULL if not found.

`lw6sys_assoc_t * lw6dyn_list_backends (`*int* **`argc`**`,` *char* * [] **`argv`**`,` *char* * **`top_level_lib`**`)` [Function]

argc: the number of command line args, as passed to main

argv: the commind line args, as passed to main

top_level_lib: the library category to query (gfx, snd, cli, srv ...)

Returns an assoc which lists all the available modules. The key of the assoc entries in the module internal name such as 'gl' and the value associated is a NULL terminated string describing the module, for instance 'OpenGL'.

Return value: an assoc object containing key/label pairs.

`char * lw6dyn_path_find_backend (`*int* **`argc`**`,` *char* * [] **`argv`**`,` *char* * **`top_level_lib`**`,` *char* * **`backend_name`**`)` [Function]

argc: the number of command-line arguments as passed to `main`

top_level_lib: the top-level library concerned, this means is it `"cli"`, `"gfx"`, `"snd"` or `"srv"`. This will tell the function to search for the .so file in the correct subdirectory. Think of this as a category.

backend_name: the actual name of the backend, this is the name of the .so file, between `"libmod_"` and `".so"`. For instance, to find `"libmod_gl.so"`, the right argument is `"gl"`.

Get the full path to a .so file corresponding to the given backend, it is capable to search for system libraries installed after `"make install"` but if not found, it will also search in the current build directory, finding the .so files in hidden .libs subdirectories.

Return value: the full path of the .so file, needs to be freed.

`int lw6dyn_test ()` [Function]

Runs the `dyn` module test suite, testing most (if not all...) functions. Will try to load libraries and query them for standard LW6-expected functions.

Return value: 1 if test is successfull, 0 on error.

4.15.6 libgfx

4.15.7 libgui

`lw6gui_menu_t * lw6gui_menu_new` (*char * `title`*) [Function]
title: the string to be displayed, what the user sees. Can be freed after the call is done, function will make a copy internally.

Constructs a new menu object. Note that you can always call other functions to modify it afterwards.

Return value: a pointer to the newly allocated object.

`void lw6gui_menu_free` (*lw6gui_menu_t * `menu`*) [Function]
menu: a pointer to the menu.

Frees the menu, checking if things are OK before doing so.

Return value: none.

`int lw6gui_menu_memory_footprint` (*lw6gui_menu_t * `menu`*) [Function]
menu: a pointer to the menu.

Gets the memory occupied by the menu. Could be usefull to help a garbage collector taking decisions or reporting erros, for instance.

Return value: the number of bytes used.

`char * lw6gui_menu_repr` (*lw6gui_menu_t * `menu`*) [Function]
menu: a pointer to the menu.

Constructs a readable description of the object. Usefull for debugging, or to introspect things using scripts, at run-time. Does not necessarly describe all the informations about the object, but helps knowing what it is.

Return value: a string describing the object, must be freed.

`void lw6gui_menu_set_title` (*lw6gui_menu_t * `menu`, char * `title`*) [Function]
menu: a pointer to the menu.

title: the new title, you can free it after calling the function, an internal copy will be made.

Change the title of the menu. That is to say, its title. Use this function to change the title, don't try to access the struct directly. The idea is to have safe memory management.

Return value: none

`lw6gui_menuitem_t * lw6gui_menu_get_item` (*lw6gui_menu_t * `menu`, int `position`*) [Function]
menu: the menu we want to query

position: the order of the item we want

Gets the menu item at the given position. First item is 0, last is N-1. Returns a pointer on the real object, not a copy.

Return value: a pointer to a menu item, NULL if out of range.

`int lw6gui_menu_select (`*lw6gui_menu_t * **menu**, int **position**, int **now***`)` [Function]

menu: the menu we want to modify

position: the position of the item we want to select

now: the current time, as a timestamp.

Selects the item at the given position. Use this function to be sure that only one item is selected, and all other states are consistent. Timestamp is needed for the sake of eye-candy.

Return value: 1 if success, 0 if failure (out of range).

`int lw6gui_menu_scroll_up (`*lw6gui_menu_t * **menu***`)` [Function]

menu: the menu to scroll

Scrolls a menu up, used as a callback for mouse wheel up for instance. The idea is just to decrement the first displayed item index.

Return value: 1 if OK, 0 if failed (out of range).

`int lw6gui_menu_scroll_down (`*lw6gui_menu_t * **menu***`)` [Function]

menu: the menu to scroll

Scrolls a menu down, used as a callback for mouse wheel down for instance. The idea is just to increment the first displayed item index.

Return value: 1 if OK, 0 if failed (out of range).

`void lw6gui_menu_center (`*lw6gui_menu_t * **menu**, int **position**, int **max_displayed_items***`)` [Function]

menu: the menu to center

position: the position of the menuitem to be put in the center

max_displayed_items: the maximum number of items displayed

Centers the menu on a given menuitem. Typically used when pushing a menu with a menuitem selected 'anywhere' in the list.

Return value: none.

`int lw6gui_menu_insert (`*lw6gui_menu_t * **menu**, lw6gui_menuitem_t * **menuitem**, int **position**, int **now***`)` [Function]

menu: the menu we want to modify

menuitem: the item to insert

position: the position the new item will occupy ("insert before" mode)

now: the current time, as a timestamp.

Inserts the given item in the menu. All items starting at the insert position will be "pushed" (that is, their position incremented by 1). Once the menuitem is inserted, the menu object will take care of memory management and automatically free it when needed.

Return value: 1 if success, 0 if failure (memory problem, out of range).

`int lw6gui_menu_append (`*lw6gui_menu_t * **menu**, lw6gui_menuitem_t * **menuitem**, int **now**`)` [Function]

menu: the menu we want to modify

menuitem: the item to insert

now: the current time, as a timestamp.

Appends the given item to the menu. Once the menuitem is appended, the menu object will take care of memory management and automatically free it when needed.

Return value: 1 if success, 0 if failure (memory problem).

`int lw6gui_menu_remove (`*lw6gui_menu_t * **menu**, int **position**, int **now**`)` [Function]

menu: the menu we want to modify

position: the item to insert

now: the current time, as a timestamp.

Removes an item from the menu. It will automatically be freed.

Return value: 1 if success, 0 if failure (out of range).

`void lw6gui_menu_update_display_range (`*lw6gui_menu_t * **menu**, int **max_displayed_items**`)` [Function]

menu: the menu concerned

max_displayed_items: the maximum number of items to display at once

Updates the display range. The reason for having this is that the first item, that is, how far we scroll in a very long menu, depends on the previous position. Plus you have to handle limit cases (begin/end). Thus, this function, which will automatically pick-up a suitable position. Of course, `first_item_displayed` is not necessarly equal to `selected_item`.

Return value: none.

`int lw6gui_menu_insert_for_id_use (`*lw6gui_menu_t * **menu**, char * **label**, int **value**, int **enabled**, int **selected**, int **colored**, int **position**, int **now**`)` [Function]

menu: the menu to work on

label: the label of the menuitem to append

value: the value of the menuitem to append

enabled: wether the inserted menuitem should be enabled

selected: wether the inserted menuitem should be selected

colored: wether the inserted menuitem should use value as its color

now: current time (timestamp)

Inserts a menu item at the given position. The idea is that the menu item object is automatically constructed on the fly, and an id is returned, which can be passed to '_using_id' menu-related functions. This is typically for using in scripts. The idea is that the script just keeps a copy of the id returned, and can this way operate directly on the menuitem without keeping a pointer, a smob or anything internally. From the

C point of view, having a real C structure enables persistent data from one display to the other, and this is nice and conveninent. I acknowledge the prototype is scary.

Return value: 0 if error, or else an id which will later be used with '_using_id' functions.

`int lw6gui_menu_append_for_id_use` (*lw6gui_menu_t * `menu`*, *char * `label`*, *int `value`*, *int `enabled`*, *int `selected`*, *int `colored`*, *int `now`*) [Function]

menu: the menu to work on

label: the label of the menuitem to append

value: the value of the menuitem to append

enabled: wether the appended menuitem should be enabled

selected: wether the appended menuitem should be selected

colored: wether the appended menuitem should use value as its color

now: current time (timestamp)

Appends a menuitem using the same logic as `lw6gui_menu_insert_for_id_use` that is to say a parameter is returned which can later be used to directly operate on a given menuitem, without having its pointer, and even if its position changes.

Return value: 0 if error, or else an id which will later be used with '_using_id' functions.

`int lw6gui_menu_remove_using_id` (*lw6gui_menu_t * `menu`*, *int `menuitem_id`*, *int `now`*) [Function]

menu: the menu to work on

menuitem_id: the id of the menuitem to remove

now: current time (timestamp)

Deletes the menuitem with the given id. Very important: the id is not the position. Id are arbitrary numbers that stick to menuitems, but they are not directly linked with the position. This function is practical to use if, for some reason, you don't have the pointer on the menuitem.

Return value: 1 if success, 0 if failure (out of range).

`void lw6gui_menu_sync_using_id` (*lw6gui_menu_t * `menu`*, *int `menuitem_id`*, *char * `label`*, *int `value`*, *int `enabled`*, *int `selected`*, *int `colored`*, *int `now`*) [Function]

menu: the menu to work on

menuitem_id: the id of the menuitem to synchronize

now: current time (timestamp)

Updates the menuitem with the given id. Very important: the id is not the position. Id are arbitrary numbers that stick to menuitems, but they are not directly linked with the position. This function is practical to use if, for some reason, you don't have the pointer on the menuitem. In practice, it's heavily used in the game to transmit informations from the scripts to the core C engine. Additionnaly, this function will automatically synchronize the `selected_item` field of the menu struct.

Return value: 1 if success, 0 if failure (out of range).

`lw6gui_menuitem_t * lw6gui_menuitem_new` (*char * `label`*, *int `value`*, *int `enabled`*, *int `selected`*, *int `colored`*) [Function]

label: the string to be displayed, what the user sees. Can be freed after the call is done, function will make a copy internally.

value: the value. No GUI function uses this, this is the "real" value associated to the item.

enabled: wether the menu item can be selected, used, and so on

selected: wether the menu item is the item selected among all menu items.

colored: wetherr the menu item must, when drawn, be colored according to its value.

Constructs a new menuitem object. Note that you can always call other functions to modify these values afterwards, this might change rendering since `lw6gui_menuitem_set_value` or `lw6gui_menuitem_set_label` will, for instance, modify the "when was that item last modified" information.

Return value: a pointer to the newly allocated object.

`void lw6gui_menuitem_free` (*lw6gui_menuitem_t * `menuitem`*) [Function]

menuitem: a pointer to the menuitem.

Frees the menuitem, checking if things are OK before doing so.

Return value: none.

`int lw6gui_menuitem_memory_footprint` (*lw6gui_menuitem_t * `menuitem`*) [Function]

menuitem: a pointer to the menuitem.

Gets the memory occupied by the menuitem. Could be usefull to help a garbage collector taking decisions or reporting erros, for instance.

Return value: the number of bytes used.

`char * lw6gui_menuitem_repr` (*lw6gui_menuitem_t * `menuitem`*) [Function]

menuitem: a pointer to the menuitem.

Constructs a readable description of the object. Usefull for debugging, or to introspect things using scripts, at run-time. Does not necessarly describe all the informations about the object, but helps knowing what it is.

Return value: a string describing the object, must be freed.

`void lw6gui_menuitem_set_label` (*lw6gui_menuitem_t * `menuitem`*, *char * `label`*, *int `now`*) [Function]

menuitem: a pointer to the menuitem.

label: the new label, you can free it after calling the function, an internal copy will be made.

now: the current time, as a timestamp.

Change the label of the menu item. That is to say, what the user sees. Use this function to change the menuitem value, don't try to access the struct directly. The idea is 1) to have safe memory management and 2) to keep the `last_change` member up to date. It can be later used for eye-candy effects.

Return value: none

`void lw6gui_menuitem_set_value` (*lw6gui_menuitem_t * **menuitem***, *int **value***, *int **now***) [Function]

menuitem: a pointer to the menuitem.

now: the current time, as a timestamp.

Changes the value of a menuitem. This is the internal value, not what the user sees. Use this function to change the menuitem value, don't try to access the struct directly. The idea is to keep the `last_change` member up to date. It can be later used for eye-candy effects.

Return value: none

`void lw6gui_menuitem_select` (*lw6gui_menuitem_t * **menuitem***, *int **now***) [Function]

menuitem: a pointer to the menuitem.

now: the current time, as a timestamp.

Switches the menuitem to selected state. Use this function, don't try to modify the struct members directly. The idea is to have the `last_select` parameter up to date. It can be later used for eye-candy effects.

Return value: none

`void lw6gui_menuitem_unselect` (*lw6gui_menuitem_t * **menuitem***, *int **now***) [Function]

menuitem: a pointer to the menuitem.

now: the current time, as a timestamp.

Switches the menuitem to unselected state. Use this function, don't try to modify the struct members directly. The idea is to have the `last_unselect` parameter up to date. It can be later used for eye-candy effects.

Return value: none

`u_int32_t lw6gui_menuitem_checksum` (*lw6gui_menuitem_t * **menuitem***, *lw6gui_look_t * **look***) [Function]

menuitem: the menuitem we want to identify

Returns a checksum which can be used to know, for instance, wether the menuitem has changed or not, and if we should redraw it.

Return value: a checksum.

`int lw6gui_resolution_find_closest` (*lw6sys_whd_t * **closest***, *lw6sys_whd_t * **wished***, *lw6sys_list_t * **available***) [Function]

closest: the closest resolution found

wished: the wished resolution

available: a list of available resolutions (list of lw6sys_whd_t *)

Finds the closest resolution available, this is just a small utility to cope with different screen shapes and avoid requesting 640x480 when it's just not available but there's a 640x400 instead.

Return value: 1 if the wished resolution exists in available list and was found, else 0 if the wished resolution doesn't exist and an approximative match was picked.

`int lw6gui_test ()` [Function]
Run tests in the gui module.

Return value: 1 if successfull, 0 if failed.

4.15.8 libhlp

`int lw6hlp_is_documented (char * keyword)` [Function]
keyword: the keyword we want to check out

Checks wether a given keyword is documented or not.

Return value: 1 if documented, 0 if not.

`char * lw6hlp_about (lw6hlp_type_t * type, char * keyword)` [Function]
type: the type of the data associated to the keyword, will be written

keyword: the keyword we want help about

Returns the documentation string associated to a keyword. The keyword might be a command-line option, a Guile function, an XML file entry. Raises a warning if the keyword is undocumented, but never returns NULL, you can use the returned value without checking it. String is localized if a translation is available. It's safe to call this function with type being NULL.

Return value: a help string, never NULL, must not be freed. Additionnally, type will be updated.

`lw6hlp_type_t lw6hlp_get_type (char * keyword)` [Function]
keyword: the keyword we want the type of

Returns the type of a keyword. Calls lw6hlp_about internally.

Return value: the type, might be LW6HLP_TYPE_VOID.

`int lw6hlp_match (char * keyword1, char * keyword2)` [Function]
keyword1: the 1st keyword

keyword2: the 2nd keyword

Checks wether a keyword matches another. Not only a string comparison, will also try and guess if the error is only about dash "-" replaced by underscode "_", for instance.

Return value: 1 if matches, 0 if different.

`lw6sys_list_t * lw6hlp_list ()` [Function]
Returns a list of all available keywords.

Return value: a list containing all the keywords. Strings are not dynamically allocated, you can't modify them.

`void lw6hlp_print_keyword (lw6sys_list_t ** list, FILE * f)` [Function]
list: a pointer to a list of keywords f: the file to print the content to

Prints all the keywords from the list. One keyword per line.

Return value: none.

`void lw6hlp_print_content` (*lw6sys_list_t ** `list`, FILE * `f`*) [Function]
list: a pointer to a list of keywords f: the file to print the content to

Prints all the keywords from the list, with the associated keyword help, to the given file. Output is formatted to fit on the standard terminal/console.

Return value: none.

`int lw6hlp_reference_init` () [Function]
Initializes the help reference, this must be called before any call to lw6hlp_about or such help related functions.

Return value: 1 on success, 0 if failed

`void lw6hlp_reference_quit` () [Function]
Un-initializes the help reference, this must be called at the end of the program.

Return value: 1 on success, 0 if failed

4.15.9 libimg

4.15.10 libker

4.15.11 libldr

`int lw6ldr_body_read` (*lw6map_body_t * `body`, char * `dirname`, lw6map_param_t * `param`, lw6ldr_hints_t * `hints`, int `display_w`, int `display_h`, float `ratio`, lw6sys_progress_t * `progress`*) [Function]
body: the body to read, must point to allocated memory

dirname: the directory of the map

param: map parameters

hints: map hints

ratio: wished map ratio

progress: structure to transmit loading progress

Reads the map body, that is, all the layers.

Return value: 1 if OK, 0 if failed.

`void lw6ldr_auto_colors` (*lw6map_style_t * `style`, lw6ldr_hints_t * `hints`*) [Function]
style: the style structure to process.

hints: additionnal hints to know what to set automatically

Deduces all colors from background color, if needed. The function will check color_auto parameters and replace all other colors by base and alternate colors if needed. Note that the background color itself is not changed by this function. Background can only be guessed from texture.

Return value: none.

`void lw6ldr_free_entry` (*lw6ldr_entry_t * `entry`*) [Function]
entry: the entry to free

Frees a map entry.

Return value: none.

`lw6sys_list_t * lw6ldr_get_entries` (*char * **map_path***, *char * **relative_path***) [Function]

map_path: the map_path environment config variable, delimited path list

relative_path: the relative path to use to find the map directory

Lists all maps in a given directory. Returns a list of lw6ldr_entry_t which can contain both directories with subdirs and actual maps. Maps are sorted before being returned, first directories, then maps, sorted in alphabetical order.

Return value: a list of dynamically allocated lw6ldr_entry_t.

`void lw6ldr_for_all_entries` (*char * **map_path***, *char * **relative_path***, *int **recursive***, *lw6sys_list_callback_func_t **callback_func***, *void * **func_data***) [Function]

map_path: the map_path environment config variable, delimited path list

relative_path: the relative path to use to find the map directory

recursive: if non-zero, map search will recurse in subdirs

callback_func: the function which will be called on each entry

func_data: an extra pointer to pass data to callback_func

Executes a given function on all maps in a given place, typically used in test programs.

Return value: none.

`int lw6ldr_hints_read` (*lw6ldr_hints_t * **hints***, *char * **dirname***) [Function]

dirname: the directory of the map

Read the hints (hints.xml) of a map. Pointer to hints must be valid, and values already initialized, either zeroed or filled in with defaults or custom values.

Return value: 1 if success, 0 if failed.

`int lw6ldr_hints_set` (*lw6ldr_hints_t * **hints***, *char * **key***, *char * **value***) [Function]

hints: the hints to modify

key: the key to modify

value: the value to affect to the key, as a string

Sets one single parameter in a hints structure. Value must always be passed as a string, will be converted to the right type automatically when storing it in the structure.

Return value: 1 if success, 0 if failed. Note that while 0 really means there's a problem, some affectations can fail and return 1, needs to be worked on.

`int lw6ldr_hints_update` (*lw6ldr_hints_t * **hints***, *lw6sys_assoc_t * **values***) [Function]

hints: the hints struct to fill with values (read/write parameter)

values: an assoc containing strings with the new values

Overrides hints with values. Pointer to hints must be valid, and values already initialized, either zeroed or filled in with defaults or custom values. Not all parameters need be defined in values. It can even be NULL. The idea is just that if something is defined in values, it will override the existing hints.

Return value: 1 if success, 0 if failed.

`int lw6ldr_param_read` (*lw6map_param_t* * **`param`**, *char* * **`dirname`**) [Function]

param: the parameter struct to fill with values (read/write parameter)

dirname: the directory of the map

Read the parameters associated to a map. Pointer to param must be valid, and values already initialized, either zeroed or filled in with defaults or custom values.

Return value: 1 if success, 0 if failed.

`int lw6ldr_param_update` (*lw6map_param_t* * **`param`**, *lw6sys_assoc_t* * **`values`**) [Function]

param: the parameter struct to fill with values (read/write parameter)

values: an assoc containing strings with the new values

Overrides param with values. Pointer to param must be valid, and values already initialized, either zeroed or filled in with defaults or custom values. Not all parameters need be defined in values. It can even be NULL. The idea is just that if something is defined in values, it will override the existing param.

Return value: 1 if success, 0 if failed.

`void lw6ldr_print_example_rules_xml` (*FILE* * **`f`**) [Function]

f: file to output content to

Print to a file a typical map rules.xml file.

Return value: none.

`void lw6ldr_print_example_hints_xml` (*FILE* * **`f`**) [Function]

f: file to output content to

Print to a file a typical map hints.xml file.

Return value: none.

`void lw6ldr_print_example_style_xml` (*FILE* * **`f`**) [Function]

f: file to output content to

Print to a file a typical map style.xml file.

Return value: none.

`int lw6ldr_print_examples` (*char* * **`user_dir`**) [Function]

user_dir: the user directory or at least, a writable one

Writes all example XML files in 'user_dir/example/', will create the directory if needed.

Return value: 1 if success, 0 if failed.

`lw6map_level_t * lw6ldr_read` (*char* * **`dirname`**, *lw6sys_assoc_t* * **`default_param`**, *lw6sys_assoc_t* * **`forced_param`**, *int* **`display_w`**, *int* **`display_h`**, *lw6sys_progress_t* * **`progress`**) [Function]

dirname: the directory containing the map

default_param: default parameters, as strings

forced_param: forced parameters, as strings

display_w: the width of the display output (resolution)

display_h: the height of the display output (resolution)

progress: information used to handle the progress bar

Loads a map from dist. The default_param and forced_param can contain values corresponding to rules.xml and style.xml entries. Parameters are read in 4 steps. 1st, a default value is picked by the program. 2nd, any value in `default_param` replaces previous values. 3rd, any value in rules.xml or style.xml replaces previous values. 4th, any value in `forced_param` replaces previous values. In practice, the `default_param` allows the user to set defaults which can still be overwritten by the map, while `forced_param` is a definitive 'ignore what is is defined in the map' way of doing things. See also `lw6ldr_read_relative`.

Return value: 1 if success, 0 if failed.

`lw6map_level_t * lw6ldr_read_relative` (*char * **map_path***, *char * **relative_path***, *lw6sys_assoc_t * **default_param***, *lw6sys_assoc_t * **forced_param***, *int **display_w***, *int **display_h***, *lw6sys_progress_t * **progress***) [Function]

map_path: a collection of paths where to find maps

relative_path: something which will be appended to a `map_path` member

default_param: default parameters, as strings

forced_param: forced parameters, as strings

display_w: the width of the display output (resolution)

display_h: the height of the display output (resolution)

progress: information used to handle the progress bar

Reads a map from disk, using the map-path value, which is a collection of paths defined by the command-line, the environment variables, and the config file. `default_param` and `forced_param` work as in the function `lw6ldr_read`.

Return value: 1 if success, 0 if failure.

`int lw6ldr_rules_read` (*lw6map_rules_t * **rules***, *char * **dirname***) [Function]

dirname: the directory of the map

Read the rules (rules.xml) of a map. Pointer to rules must be valid, and values already initialized, either zeroed or filled in with defaults or custom values.

Return value: 1 if success, 0 if failed.

`int lw6ldr_rules_update` (*lw6map_rules_t * **rules***, *lw6sys_assoc_t * **values***) [Function]

rules: the rules struct to fill with values (read/write parameter)

values: an assoc containing strings with the new values

Overrides rules with values. Pointer to rules must be valid, and values already initialized, either zeroed or filled in with defaults or custom values. Not all parameters need be defined in values. It can even be NULL. The idea is just that if something is defined in values, it will override the existing rules.

Return value: 1 if success, 0 if failed.

`int lw6ldr_style_read` (*lw6map_style_t * **style***, *char * **dirname***) [Function]

dirname: the directory of the map

Read the style (style.xml) of a map. Pointer to style must be valid, and values already initialized, either zeroed or filled in with defaults or custom values.

Return value: 1 if success, 0 if failed.

`int lw6ldr_style_set` (*lw6map_style_t * **style***, *char * **key***, *char * **value***) [Function]

style: the style to modify

key: the key to modify

value: the value to affect to the key, as a string

Sets one single parameter in a style structure. Value must always be passed as a string, will be converted to the right type automatically when storing it in the structure.

Return value: 1 if success, 0 if failed. Note that while 0 really means there's a problem, some affectations can fail and return 1, needs to be worked on.

`int lw6ldr_style_update` (*lw6map_style_t * **style***, *lw6sys_assoc_t * **values***) [Function]

style: the style struct to fill with values (read/write parameter)

values: an assoc containing strings with the new values

Overrides style with values. Pointer to style must be valid, and values already initialized, either zeroed or filled in with defaults or custom values. Not all parameters need be defined in values. It can even be NULL. The idea is just that if something is defined in values, it will override the existing style.

Return value: 1 if success, 0 if failed.

`int lw6ldr_use_update` (*lw6ldr_use_t * **use***, *lw6sys_assoc_t * **values***) [Function]

use: the use struct to fill with values (read/write parameter)

values: an assoc containing strings with the new values

Overrides use with values. Pointer to use must be valid, and values already initialized, either zeroed or filled in with defaults or custom values. Not all parameters need be defined in values. It can even be NULL. The idea is just that if something is defined in values, it will override the existing use.

Return value: 1 if success, 0 if failed.

4.15.12 libmap

`void lw6map_color_invert` (*lw6map_color_couple_t * **color***) [Function]

color: the color to invert

Inverts a color couple, that is, replace fg by bg and vice-versa.

Return value: none.

`int lw6map_color_is_same` (*lw6map_color_couple_t * **color1***, *lw6map_color_couple_t * **color2***) [Function]

color1: 1st color to compare

color2: 2nd color to compare

Compares two colors.

Return value: 1 if equal, 0 if not.

`char * lw6map_team_color_index_to_key` (*int* ***index***) [Function]

index: index of the color between 0 & 9

Transforms a team color index into its readable string form, which can be used in config files for instance.

Return value: a string, must *not* be freed.

`int lw6map_team_color_key_to_index` (*char * **key***) [Function]

key: key of the color, for instance "red"

The index of the color, between 0 & 9

Return value: an integer.

`lw6map_level_t * lw6map_dup` (*lw6map_level_t * **source***) [Function]

source: the map to copy

Performs a deep copy of the map, all elements are newly allocated and source can safely be destroyed after it's been duplicated.

Return value: a newly allocated map, may be NULL.

`char * lw6map_to_hexa` (*lw6map_level_t * **level***) [Function]

Converts a map to something that is later readable by `lw6map_from_hexa` to reproduce the exact same map. Just a serializer.

Return value: a newly allocated pointer, NULL if conversion failed.

`lw6map_level_t * lw6map_from_hexa` (*char * **hexa***) [Function]

hexa: an hexadecimal ASCII string, created by `lw6map_to_hexa`

Constructs a map from an hexadecimal string generated by `lw6map_to_hexa`. Just an un-serializer.

Return value: a new map, might be NULL if string isn't correct.

`lw6map_level_t * lw6map_new` () [Function]

Creates a new empty map. This object is perfectly unusable as is, since it has a 0x0 size, and many things set to "NULL". Still, it's used internally and is the canonical way to create the object, it ensures later calls that set up default parameters, for instance, will succeed.

Return value: a newly allocated pointer.

`lw6map_level_t * lw6map_defaults` (*int* ***nb_layers***, *int* `w`, *int* `h`) [Function]

nb_layers: the number of layers of the map

w: the width of the map

h: the height of the map

Creates a map, set to defaults. This is usefull mostly for testing, it will just create a dull rectangle plain and uninteresting map, however it's a quick way to have a working object and test stuff on it.

Return value: a newly allocated map.

`void lw6map_free` (*lw6map_level_t * level*) [Function]
Frees a map and releases all its internal ressources.

Return value: none.

`int lw6map_memory_footprint` (*lw6map_level_t * level*) [Function]
Reports how many bytes the map needs, in memory. Note that this is not contiguous memory, it involves a bunch of pointers, and possibly much more...

`char * lw6map_repr` (*lw6map_level_t * level*) [Function]
Returns a string describing the map. This is a very short description, use it for logs, and to debug stuff. By no means it's a complete exhaustive description. Still, the string returned should be unique.

Return value: a dynamically allocated string.

`void lw6map_param_defaults` (*lw6map_param_t * param*) [Function]
param: the param struct to modify

Sets a param structure to its default value, note that current structured must be zeroed or correctly initialized.

Return value: none

`void lw6map_param_clear` (*lw6map_param_t * param*) [Function]
param: the param struct to modify

Resets a param structure to nothing. Note that current structured must be zeroed or correctly initialized. The idea is just to free member pointers before calling free.

Return value: none

`void lw6map_param_copy` (*lw6map_param_t * dst*, *lw6map_param_t * src*) [Function]
dst: the destination param struct

src: the source param struct

Copies parameters. Both structures must be zeroed or correctly initialized.

Return value: none

`int lw6map_param_set` (*lw6map_param_t * param*, *char * key*, *char * value*) [Function]
param: the param struct to modify

key: the name of the parameter to modify

value: the value of the parameter to modify

Sets an entry in a param struct. All values must be submitted as strings, internally, the function will call atoi to convert to integers if needed, for instance. It will also dispatch automatically between rules and style.

Return value: 1 if parameter successfully set, 0 on error.

4.15.13 libnet

`int lw6net_last_error ()` [Function]

Reports the last network error. This is basically a debug function, designed mostly for Microsoft Winsock API, but can be safely called on any platform.

Return value: the last error code, has no universal meaning, depends on the platform you're working on.

`char * lw6net_recv_line_tcp (int sock)` [Function]

sock: the socket descriptor

Receives a line terminated by LF ("\n", chr(10)) or CR/LF ("\r\n", chr(10)chr(13)) on a TCP socket, that is, stream oriented. If there's no complete line available, function returns immediately with NULL. Same if socket is closed, broken, whatever. Only if there's something consistent will the function return non-NULL.

Return value: a dynamically allocated string with the content received. The tailing (CR)/LF is stripped.

`int lw6net_send_line_tcp (int sock, char * line)` [Function]

sock: the socket descriptor

line: the line to be sent, without the "\n" at the end

Sends a line terminated by LF ("\n", chr(10)) on a TCP socket, that is, stream oriented. The "\n" is automatically added, do not bother sending it.

Return value: non-zero if success

`char * lw6net_recv_line_udp (int sock, char ** incoming_ip, int * incoming_port)` [Function]

sock: the socket descriptor

incoming_ip: the IP address of the sender (returned)

incoming_port: the IP port of the sender (returned)

Receives a line terminated by LF ("\n", chr(10)) or CR/LF ("\r\n", chr(10)chr(13)) on a UDP socket, that is, datagram oriented. If there's no complete line available, function returns immediately with NULL. Same if socket is closed, broken, whatever. Only if there's something consistent will the function return non-NULL. By-value parameters allow the caller to know where the data come from.

Return value: a dynamically allocated string with the content received. The tailing (CR)/LF is stripped.

`int lw6net_send_line_udp (int sock, char * line, char * ip, int port)` [Function]

sock: the socket descriptor

line: the line to be sent, without the "\n" at the end

ip: the IP address of the target

port: the IP port of the target

Sends a line terminated by LF ("\n", chr(10)) on a UDP socket, that is, datagram oriented. The "\n" is automatically added, do not bother sending it.

Return value: the number of bytes sent, 0 if failure

`int lw6net_init` (*int* `argc`, *char* * [] `argv`) [Function]

Initializes the low-level network API, you must call this before calling any other network related function, for it allocates a dynamic context which is in turn used by every function.

Return value: non-zero if success

`void lw6net_quit` () [Function]

Frees memory, joins active threads, and releases everything set up by network code.

Return value: void

4.15.14 libp2p

4.15.15 libpil

`int lw6pil_bench` (*float* * `bench_result`) [Function]

bench_result: pointer to float, will contain the bench result

Runs a standard, normalized bench on a default map. Results can be interpreted as an estimated speed/power of your computer.

Return value: 1 on success, 0 if failure

`void lw6pil_coords_fix` (*lw6map_rules_t* * `rules`, *lw6sys_whd_t* * `shape`, *float* * `x`, *float* * `y`, *float* * `z`) [Function]

rules: the set of rules to use (defines polarity)

shape: the shape of the map (logical part)

x: the x coord to fix

y: the y coord to fix

z: the z coord to fix

Similar to `lw6map_coords_fix` but using floats, this function can be used to check cursor position boundaries. Any float pointer can be NULL.

Return value: none.

`lw6pil_pilot_t * lw6pil_pilot_new` (*lw6ker_game_state_t* * `game_state`, *float* `sleep`, *int* `ticks`) [Function]

game_state: the game state we're going to work on

sleep: sleep latency, in seconds, typically very low (0.002)

ticks: the current ticks (1000 ticks per sec, used to calibrate)

Initializes a 'pilot' object, this object is responsible for interpreting messages, transform them into low-level 'ker' module function calls, and handle all the thread-spooky stuff.

Return value: a working pilot object. May be NULL on memory failure.

`void lw6pil_pilot_free` (*lw6pil_pilot_t* * `pilot`) [Function]

pilot: the object to free.

Frees a 'pilot' object, note that this might involve joining some threads, so it can 'take some time'.

Return value: none.

`int lw6pil_pilot_send_command` (*lw6pil_pilot_t* * **`pilot`**, *char* * **`command_text`**, *int* **`verified`**) [Function]

pilot: the object to send commands to.

command_text: the text of the command, as received form network

verified: wether we're sure this message is valid.

Sends a command and handles it internally.

Return value: 1 if OK, 0 if not.

`int lw6pil_pilot_commit` (*lw6pil_pilot_t* * **`pilot`**) [Function]

pilot: the object to commit.

Commits all commands sent and actually send them to the corresponding threads. This commit system allows better performance by sending, for instance, all the commands for a given round together.

Return value: none.

`int lw6pil_pilot_make_backup` (*lw6pil_pilot_t* * **`pilot`**) [Function]

pilot: the object to perform the backup on

Makes a new backup in the pilot, that is, copy 'reference' to 'backup'.

Return value: 1 if OK, 0 if not.

`int lw6pil_pilot_sync_from_backup` (*lw6ker_game_state_t* * **`target`**, *lw6pil_pilot_t* * **`pilot`**) [Function]

target: the game_state structure which will get the informations.

pilot: the object to get informations from.

Gets the backup from the pilot object. This is the last snapshot taken by `make_backup` or, by default, the game_state the pilot was constructed with.

Return value: 1 if OK, 0 if not.

`int lw6pil_pilot_sync_from_reference` (*lw6ker_game_state_t* * **`target`**, *lw6pil_pilot_t* * **`pilot`**) [Function]

target: the game_state structure which will get the informations.

pilot: the object to get informations from.

Gets the latest reference game_state, that is, a stable snapshot of the game, with no inconsistency, a game position that exists and that we can rely on. Note that getting this can take time since a global mutex is required, and computations must end before you get the data.

Return value: 1 if OK, 0 if not.

`int lw6pil_pilot_sync_from_draft` (*lw6ker_game_state_t* * **`target`**, *lw6pil_pilot_t* * **`pilot`**) [Function]

target: the game_state structure which will get the informations.

pilot: the object to get informations from.

Gets the informations from the pilot object, not being worried about game consistency, this one will just return the latest version available. It might even be in an inconsistent state, the position could reflect a position which will never exist. Still, the data

returned will not correspond to a half-spread or half-moved game_state. The data has at least some basic consistency and getting this does require some mutex lock, however wait time should be fairly small (max. a round).

Return value: 1 if OK, 0 if not.

`char * lw6pil_pilot_repr` (*lw6pil_pilot_t* * `pilot`) [Function]

Returns a string describing the pilot. This is a very short description, use it for logs, and to debug stuff. By no means it's a complete exhaustive description. Still, the string returned should be unique.

Return value: a dynamically allocated string.

`void lw6pil_pilot_calibrate` (*lw6pil_pilot_t* * `pilot`, *int* `ticks`, *int* `round`) [Function]

pilot: the object to calibrate

ticks: the current ticks setting (1000 ticks per second)

round: the round expected to be returned with this ticks value

Calibrates the pilot, that is, initializes it so that subsequent calls to `lw6pil_pilot_get_round` return consistent values.

Return value: none.

`void lw6pil_pilot_speed_up` (*lw6pil_pilot_t* * `pilot`, *int* `round_inc`) [Function]

pilot: the pilot to speed up

round_inc: the number of rounds

Re-calibrates the pilot so that it speeds up a bit. This will basically increase next_round by round_inc.

Return value: none.

`void lw6pil_pilot_slow_down` (*lw6pil_pilot_t* * `pilot`, *int* `round_dec`) [Function]

pilot: the pilot to speed up

round_dec: the number of rounds

Re-calibrates the pilot so that it slows down a bit. This will basically decrease next_round by round_inc.

Return value: none.

`int lw6pil_pilot_get_next_round` (*lw6pil_pilot_t* * `pilot`, *int* `ticks`) [Function]

pilot: the object to query

ticks: the current ticks setting (1000 ticks per second)

Returns the round one should use to generate new events/commands at a given time (given in ticks).

Return value: none.

`int lw6pil_pilot_get_last_commit_round` (*lw6pil_pilot_t* * `pilot`) [Function]

pilot: the object to query

Returns the round of the last commit (reference game_state) for this object.

Return value: the commit round (reference object)

`int lw6pil_pilot_get_reference_target_round (`*lw6pil_pilot_t * pilot*`)` [Function]

pilot: the object to query

Returns the round which is targetted in the reference game_state, this is 'how far computation will go in the reference game_state if no new commands are issued'. Note that there can always be some commands which are not yet processed, so you should not rely on this too heavily, however it gives a good idea of how things are going.

Return value: the target round (reference object)

`int lw6pil_pilot_get_reference_current_round (`*lw6pil_pilot_t * pilot*`)` [Function]

pilot: the object to query

Returns the current round in the reference game_state. There's no lock on this call so don't rely on this too heavily, it just gives you an idea of wether the pilot is very late on its objectives or just on time.

Return value: the current round (reference object)

`int lw6pil_pilot_get_max_round (`*lw6pil_pilot_t * pilot*`)` [Function]

pilot: the object to query

Returns the max current round in the reference or draft game states. No lock on this call so don't rely on this too heavily, it just gives you an idea of computation state.

Return value: the current round (reference object)

`int lw6pil_test ()` [Function]

Runs the `pil` module test suite.

Return value: 1 if test is successfull, 0 on error.

4.15.16 libsnd

4.15.17 libsrv

4.15.18 libsys

`int lw6sys_arg_match (`*char * keyword, char * argv_string*`)` [Function]

keyword: the option to match, without the prefix `"-"` or `"–"`

argv_string: the argv value, for instance argv[1]

This is an utility function which allow the program to handle options in a uniform manner. Key comparison is insensitive, that is, –option and –OPTION are equivalent. Besides, -option and –OPTION are equivalent too. Liquid War 6 documentation mentions options in lowercase with a double dash (–option) by default, but it's a fact, the program supports variants. This is just for convenience, the philosophy behind this behavior is "be as permissive as possible when interpreting input, and as strict as possible when generating output". In fact, it's even said that Liquid War 6 will accept the argument without any prefix dash as being valid... This is to say running "liquidwar6 –option" is the same as running "liquidwar6 option". But, this is a secret 8-)

Return value: non zero if it matches, 0 if it doesn't.

`int lw6sys_arg_exists (`*int* `argc`, *char * []* `argv`, *char *` `keyword``)` [Function]

argc: the number of arguments, as passed to `main`

argv: an array of arguments, as passed to `main`

keyword: the keyword to match

Parses all command-line arguments, searching for one precise "–key[=...]" entry.

Return value: 1 if key is present, 0 if not.

`char * lw6sys_arg_get_value (`*int* `argc`, *char * []* `argv`, *char *` `keyword``)` [Function]

argc: the number of arguments, as passed to `main`

argv: an array of arguments, as passed to `main`

keyword: the keyword to match

Parses all command-line arguments, searching for one precise "–key=value" pair, and returns the value.

Return value: a pointer to the value. May be NULL. Must be freed.

`char * lw6sys_arg_get_value_with_env (`*int* `argc`, *char * []* `argv`, *char *` `keyword``)` [Function]

argc: the number of arguments, as passed to `main`

argv: an array of arguments, as passed to `main`

keyword: the keyword to match

Parses all command-line arguments, searching for one precise "–key=value" pair, and returns the value. If a corresponding environment variable is available, but no command-line parameter was passed, the environment variable is intepreted. Such environment variables are uppercased, prefixed by "LW6_" and "_" replaces "-". The environment variable will be overriden if the command-line parameter is present.

Return value: a pointer to the value. May be NULL. Must be freed.

`lw6sys_assoc_t * lw6sys_assoc_new (`*lw6sys_free_func_t* `free_func`)` [Function]

free_func: optional callback used to free memory when stored date is a pointer. Can be NULL when one stores non dynamically allocated data, such as an integer or a static array.

Creates an empty assoc. There's a difference between NULL and an empty assoc. The empty assoc would (in Scheme) be '() whereas NULL corresponds to undefined "is not a assoc and will generate errors if you ever call assoc functions on it". Such created assoc are not performant hash tables but slowish "strcmp me for each key" associative arrays, the key being a "char *" string and the value a "void *" pointer.

Return value: a pointer to the newly allocated associative array. Must be freed with `lw6sys_assoc_free`.

`void lw6sys_assoc_free (`*lw6sys_assoc_t *` `assoc`)` [Function]

assoc: the assoc to be freed.

The function will cascade delete all elements, using (if not NULL...) the callback passed when first creating the assoc.

Return value: void

`int lw6sys_assoc_has_key` (*lw6sys_assoc_t* * `assoc`, *char* * `key`) [Function]

assoc: the assoc to test

key: the key to search

Not a very fast function, since on a "big" assoc, strcmp will be called internally until the key is found.

Return value: non-zero if there's an entry with the corresponding key.

`void * lw6sys_assoc_get` (*lw6sys_assoc_t* * `assoc`, *char* * `key`) [Function]

assoc: the assoc to query

key: the key of which we want the value

Return value: a void pointer to the data contained in the assoc. Note that the pointer on the actual data is returned, that is, if it's static data, you must not try to free it... As long as memory management is concerned, destroying the assoc will actually free the data if needed.

`void lw6sys_assoc_set` (*lw6sys_assoc_t* ** `assoc`, *char* * `key`, *void* * `value`) [Function]

assoc: the assoc to modify

key: the key we want to updated

value: the new value

Sets a value in an associative array. The key pointer need not be persistent, it can be freed after affectation. In fact a new string will be created internally. This is not true for the value, it's hard to find way to copy "any object". So if you want an associative array of strings, key can disappear after calling this function, but not value. The function passed as free_func when creating the assoc will be used to free stuff whenever needed (unset or free).

Return value: void

`void lw6sys_assoc_unset` (*lw6sys_assoc_t* * `assoc`, *char* * `key`) [Function]

assoc: the assoc concerned

key: the key to unset

Clears an entry in an associative array. The callback passed when creating the assoc will be called if needed, to free the data automatically.

Return value: void

`lw6sys_list_t * lw6sys_assoc_keys` (*lw6sys_assoc_t* * `assoc`) [Function]

assoc: the assoc to work on

Returns a list containing all the keys of the assoc. The list must be free with lw6sys_list_free by the caller. This list copies all the keys of the assoc, so it is safe to use it once the assoc is deleted. However the keys will of course be of little interest in this case. But the program won't segfault.

Return value: the list of keys.

`void lw6sys_assoc_map` (*lw6sys_assoc_t* * **assoc**, *lw6sys_assoc_callback_func_t* **func**, *void* * **func_data**) [Function]

assoc: the assoc to work on

func: a callback to call on each entry

func_data: a pointer on some data which will be passed to the callback

Executes a function on all assoc items. The func_data parameter allows you to pass extra values to the function, such as a file handler or any variable which can not be inferred from list item values, and you of course do not want to make global...

Return value: void

`void lw6sys_assoc_sort_and_map` (*lw6sys_assoc_t* * **assoc**, *lw6sys_assoc_callback_func_t* **func**, *void* * **func_data**) [Function]

assoc: the assoc to work on

func: a callback to call on each entry, may be NULL

func_data: a pointer on some data which will be passed to the callback

Executes a function on all assoc items, like `lw6sys_assoc_sort_and_map` but befor doing so, sorts all entries in alphabetical order.

Return value: void

`lw6sys_assoc_t * lw6sys_assoc_dup` (*lw6sys_assoc_t* * **assoc**, *lw6sys_dup_func_t* **dup_func**) [Function]

assoc: the assoc to duplicate, can be NULL

dup_func: the function which will be called to duplicate data

Duplicates an assoc. All keys will be copied so that if the first assoc is deleted, the duplicated one is fine. Additionnaly, dup_func will be called with all data fields. If dup_func is NULL, then data values will simply be copied. This is likely to be usefull when data is not dynamically allocated.

Returned value: a newly allocated assoc.

`int lw6sys_default_memory_bazooka` () [Function]

Will set up a default memory bazooka, a slow yet convenient tool to track down and hopefully kill memory leaks. Named bazooka after a night wasted to track down an unfoundable leak... BAZOOOOOOKA!!!

Return value: 1 if success, 0 if failed.

`void lw6sys_clear_memory_bazooka` () [Function]

Clears the memory bazooka.

Return value: none.

`int lw6sys_set_memory_bazooka_size` (*int* **size**) [Function]

size: number of items (calls to malloc) to keep

Resizes, the memory bazooka. What's this? It's an inelegant yet efficient tool to track down memory leak. Memory bazooka will keep track of every call to malloc, keeping a trace of what has been malloced, where it has been called (from which file, which line), how much memory was allocated, it will even show you what's at the address

in a 0-terminated string-friendly fashion. Of course this slows down the program, so in production, you might set this to 0, but for debugging, a million bazooka is worth the megabytes and CPU cycles it wastes.

Return value: 1 if success, 0 if failure.

`int lw6sys_get_memory_bazooka_size ()` [Function]

The companion of `lw6sys_set_memory_bazooka_size`. This function will return how many calls to malloc can be traced. A return value of 0 indicates that feature is disabled.

Return value: size of the bazooka array.

`int lw6sys_set_memory_bazooka_eraser (`*int* `state`*`)` [Function]

state: the state of the eraser

Sets the memory bazooka eraser state. Note that to really work, it requires the memory bazooka to be "big enough".

Return value: 1 if activated, 0 if not. Note that the main reason for it not to be activated is if the memory bazooka has zero size.

`int lw6sys_get_memory_bazooka_malloc_count ()` [Function]

Provided you have always called the `LW6SYS_MALLOC` an `LW6SYS_CALLOC` to allocate memory, this function will tell you how many times `malloc` has been called.

Return value: the number of calls to `lw6sys_malloc` or `lw6sys_calloc` since program was started.

`int lw6sys_get_memory_bazooka_free_count ()` [Function]

Provided you have always called the `LW6SYS_FREE` macro to free memory, this function will tell you how many times `free` has been called.

Return value: the number of calls to `lw6sys_free` since program was started.

`int lw6sys_memory_bazooka_report ()` [Function]

Reports memory bazooka diagnostics on the console. Carefull, this one is not reentrant, call at the end of your program when all threads are joined.

Return value: 1 if no allocated stuff left, 0 if there are still malloc'ed stuff

`char * lw6sys_build_get_package_tarname ()` [Function]

Returns the name of the package. This is the `PACKAGE_TARNAME` constant defined by the GNU Autoconf ./configure script. While it's always possible to use the defined constant directly, using this function will return the value defined when compiling the binary, not the one you're using when compiling another program relying on Liquid War as a library.

Return value: a non-NULL string "liquidwar6", must not be freed.

`char * lw6sys_build_get_package_name ()` [Function]

Returns the name of the package, in a user friendly form, which can include spaces, for instance. This is the `PACKAGE_NAME` constant defined by the GNU Autoconf ./configure script. While it's always possible to use the defined constant directly, using this function will return the value defined when compiling the binary, not the one you're using when compiling another program relying on Liquid War as a library.

Return value: a non-NULL string "Liquid War 6", must not be freed.

`char * lw6sys_build_get_package_string ()` [Function]
Returns the description of the package. This is the `PACKAGE_STRING` constant defined by the GNU Autoconf ./configure script. It's the concatenation of `PACKAGE_NAME` and `VERSION`. While it's always possible to use the defined constant directly, using this function will return the value defined when compiling the binary, not the one you're using when compiling another program relying on Liquid War as a library.

Return value: a non-NULL string "Liquid War 6 <version>", must not be freed.

`char * lw6sys_build_get_version ()` [Function]
Returns the version of the program. This is the `VERSION` constant defined by the GNU Autoconf ./configure script. Same as `PACKAGE_VERSION`. Note that while using a function to get `PACKAGE_TARNAME` might seem useless, having both ways to get the version, that is, a function and a constant, is very usefull. Think, for instance, that a dynamically loaded shared library might need to check its own version against the version of the core program.

Return value: a non-NULL string, which must not be freed.

`char * lw6sys_build_get_codename ()` [Function]
Returns the the program codename. This is the little name of the version. It's been decided that all LW6 releases would take the name of a famous general, warrior, whatever. For instance, it could be "Napoleon".

Return value: a non-NULL string, traditionnally the name of a famous general, someone which has been involved in war. Must not be freed (I mean, the string, not the general).

`char * lw6sys_build_get_stamp ()` [Function]
Returns the program stamp. This is like a serial number. It's is not the same as the version. The version is meant to be set to something readable. This is just a cryptic thing, incremented at each ./configure or each developper's "I feel like it needs to be incremented". The idea is just to keep (one more...) track of which source code is build. Ideally, this would be plugged to the source revision control system but this has some drawbacks, including that it would require it to modify files before commiting them, which is not safe, and almost impossible if you sign archives. One more point: this is a string. It's true the return value is actually a string containing the representation of an integer, but because all other build parameters are strings, and because we don't know what the future reserves, it's a string.

Return value: a non-NULL string like "42", which must not be freed.

`char * lw6sys_build_get_md5sum ()` [Function]
Returns an md5 checkum which is caculated from C (.c and .h) source files. This is complementary with the build stamp. By default the stamp will be enough to check what has been compiled, but one can always imagine a case where Bob compiles something a little different than Alice, with the same stamp, incremented by 1 from a common source tree. They apply their own patches, for instance. This md5sum double-checks that two binaries have been built from the same sources. Note that this is not the md5 checksum of the generated binary. Nor does it include any information about scheme scripts and data.

Return value: a non-NULL string, which must not be freed.

`char * lw6sys_build_get_copyright ()` [Function]
Returns a (very) short copyright information about the program.

Return value: a non-NULL string, single line whithout '\n' at the end. Must not be freed.

`char * lw6sys_build_get_license ()` [Function]
Returns the license for the program (GNU GPL v3 or later).

Return value: a non-NULL string, single line whithout '\n' at the end. Must not be freed.

`char * lw6sys_build_get_configure_args ()` [Function]
Returns the arguments passed to the GNU Autoconf ./configure script when buildling the game. Very usefull to know how the binary was generated, that is, what kind of optimizations are peculiar settings it uses.

Return value: a non-NULL string, which, passed to ./configure again, would hopefully generate the same binary. Must not be freed.

`char * lw6sys_build_get_gcc_version ()` [Function]
Returns __VERSION__ GCC preprocessor value, that is, the human readable version of the compiler.

Return value: a non-NULL string, must not be freed.

`char * lw6sys_build_get_cflags ()` [Function]
Returns the arguments which would allow another program to use liquidwar6 as a library. Typically, pass this to gcc when compiling your sources. Basically contains `"-I"` switches which tell where the headers are.

Return value: a non-NULL string, which must not be freed.

`char * lw6sys_build_get_ldflags ()` [Function]
Returns the arguments which would allow another program to link against liquidwar6. Pass this to gcc or libtool when compiling your program. Basically contains a `"-L"` option which says where the library is. Note that this will only allow you to link against the main libliquidwar6 library, but not the dynamically loaded modules.

Return value: a non-NULL string, which must not be freed.

`char * lw6sys_build_get_hostname ()` [Function]
Returns the value return by the standard shell `hostname` command on the machine where the game has been built. Usefull to track binaries and know where do they come from.

Return value: a non-NULL string, must not be freed.

`char * lw6sys_build_get_date ()` [Function]
Returns the compilation date. While this information can easily be obtained with the C `__DATE__` macro, having this function is convenient for it returns a value which is the same for the whole program, and does not possibly change in every file.

Return value: a non-NULL string, must not be freed.

`char * lw6sys_build_get_time ()` [Function]
Returns the compilation date. While this information can easily be obtained with the C `__TIME__` macro, having this function is convenient for it returns a value which is the same for the whole program, and does not possibly change in every file.

Return value: a non-NULL string, must not be freed.

`char * lw6sys_build_get_target_cpu ()` [Function]
Returns the CPU this program is designed for. Convenient on i386 compatible CPUs to know which flavor (i386, i586...) the binary is made for.

Return value: a non-NULL string, must not be freed.

`char * lw6sys_build_get_endianness ()` [Function]
Returns the endianness of the computer.

Return value: 'little' (x86-like) or 'big' (ppc-like), as a string. Must not be freed.

`int lw6sys_build_get_pointer_size ()` [Function]
Returns the system pointer size, in bytes.

Return value: 4 for 32-bit, 8 for 64-bit.

`char * lw6sys_build_get_target_os ()` [Function]
Returns the OS this program is designed for. Usefull for bug reports.

Return value: a non-NULL string, must not be freed.

`int lw6sys_build_is_ms_windows ()` [Function]
Tells wether the program was compiled for Microsoft Windows, or not.

Return value: 1 if compiled on windows 32-bit, 0 if not

`char * lw6sys_build_get_top_srcdir ()` [Function]
Returns the top source directory, when the game was built. This can seem useless and non relevant on the end-user's machine, but... it's a must-have for developpers and packagers. Without this, binaries would never find their associated data, especially when building outside the source tree. Or, testing the game would be impossible without installing it, given the fact that most of the code is in scripts that are stored in /usr/local by default, this would be painfull. So this function is here to help finding data within the source tree when the game is not installed yet. Note that the function is rather clever, since it will automatically try to remove useless '../' sequences at the beginning of a possibly relative path.

Return value: a non-NULL string, must not be freed.

`char * lw6sys_build_get_prefix ()` [Function]
Returns the `prefix` value as given to the GNU Autoconf ./configure script. Used to deduce the path to other directories and files.

Return value: a non-NULL string, `"/usr/local"` by default. Must not be freed.

`char * lw6sys_build_get_datadir ()` [Function]
Returns the `datadir` value defined by the GNU Autoconf ./configure script. This is not the value which can be overriden by the Liquid War 6 specific. `"–data-dir"` option. `datadir` is usually something like `"/usr/local/share"` while the actual Liquid War 6

defined data dir is a more profound path which includes the name of the package, its version, and so on.

Return value: a non-NULL string, "/usr/local/share" by default. Must not be freed.

`char * lw6sys_build_get_libdir ()` [Function]
Returns the `libdir` value defined by the GNU Autoconf ./configure script. This is not the value which can be overriden by the Liquid War 6 specific. "–mod-dir" option. `datadir` is usually something like "/usr/local/lib" while the actual Liquid War 6 defined module dir is a more profound path which includes the name of the package, its version, and so on.

Return value: a non-NULL string, "/usr/local/lib" by default. Must not be freed.

`char * lw6sys_build_get_includedir ()` [Function]
Returns the `includedir` value defined by the GNU Autoconf ./configure script. As for other options, it's interesting to have this value, this enables the program to inform people who want to hack the game of the place headers are supposed to be installed.

Return value: a non-NULL string, "/usr/local/include" by default. Must not be freed.

`char * lw6sys_build_get_localedir ()` [Function]
Returns the `localedir` value defined by the GNU Autoconf ./configure script. Used as an argument for gettext / libintl functions.

Return value: a non-NULL string, "/usr/local/share/locale" by default. Must not be freed.

`char * lw6sys_build_get_docdir ()` [Function]
Returns the `docdir` value defined by the GNU Autoconf ./configure script. Used to write consistent XML file headers.

Return value: a non-NULL string, "/usr/local/share/doc/liquidwar6" by default. Must not be freed.

`char * lw6sys_build_get_enable_console ()` [Function]
Tells wether console is enabled or not.

Return value: "yes" or "no", must no be freed.

`char * lw6sys_build_get_enable_mod_gl ()` [Function]
Tells wether the graphical mod-gl backend was compiled.

Return value: "yes" or "no", must no be freed.

`char * lw6sys_build_get_enable_mod_csound ()` [Function]
Tells wether the audio mod-csound backend was compiled.

Return value: "yes" or "no", must no be freed.

`char * lw6sys_build_get_enable_mod_ogg ()` [Function]
Tells wether the audio mod-ogg backend was compiled.

Return value: "yes" or "no", must no be freed.

`char * lw6sys_build_get_enable_mod_http ()` [Function]
Tells wether the network mod-http backend was compiled.

Return value: "yes" or "no", must no be freed.

`char * lw6sys_build_get_enable_optimize ()` [Function]
Tells wether the game was compiled in optimize mode.

Return value: "yes" or "no", must no be freed.

`char * lw6sys_build_get_enable_allinone ()` [Function]
Tells wether the game was compiled in allinone mode.

Return value: "yes" or "no", must no be freed.

`char * lw6sys_build_get_enable_fullstatic ()` [Function]
Tells wether the game was compiled in fullstatic mode.

Return value: "yes" or "no", must no be freed.

`char * lw6sys_build_get_enable_gprof ()` [Function]
Tells wether the game was compiled with suitable informations for gprof.

Return value: "yes" or "no", must no be freed.

`char * lw6sys_build_get_enable_gcov ()` [Function]
Tells wether the game was compiled with suitable informations for gcov.

Return value: "yes" or "no", must no be freed.

`char * lw6sys_build_get_enable_valgrind ()` [Function]
Tells wether the game was compiled for later use with valgrind.

Return value: "yes" or "no", must no be freed.

`void lw6sys_build_log_all ()` [Function]
Dumps in the log file the whole program pedigree, target, modules, that is, what are the values of all the build options. Usefull for bug reports.

Return value: none.

`u_int32_t lw6sys_checksum (`*unsigned char * `data`, int `len`*`)` [Function]
data: the data to process

len: the length, in bytes, of the data to process

Creates a checksum from a byte array. This could be mapped on any standard CRC-32 and/or MD5 algorithm, but licence issues for those are such a headache that for the sake of simplicity, it's wrapped here. In LW6 context, we do not really really fear any attack for these checksums are used internally to track bugs and check, for instance, that two game states are actually the same on two distant computers in a network game. Data encryption and security of network links is another debate. Additionnally, this function returns an integer, easier to handle in standard C than any malloc'ed stuff.

Return value: the checksum, as an integer.

`u_int32_t lw6sys_checksum_str` (*char * `value`*) [Function]

value: the string to process

Creates a checksum from a string. This is a convenience function to save the programmer the hassle of calling strlen before any checksum calculation.

Return value: the checksum, as an integer.

`u_int32_t lw6sys_checksum_int32` (*u_int32_t `value`*) [Function]

value: the integer to process

Creates a checksum from an integer. This is a convenience function to save the programmer the hassle of passing a pointer to the integer with the size of it each time there's a checksum to do. Additionnally, with this one you can pass an int8 or an int16, and function will work just the same indenpendantly of endianness.

Return value: the checksum, as an integer.

`u_int32_t lw6sys_checksum_int64` (*u_int64_t `value`*) [Function]

value: the integer to process

Creates a checksum from an integer. This is a convenience function to save the programmer the hassle of passing a pointer to the integer with the size of it each time there's a checksum to do. This function handles 64-bit long long integers..

Return value: the checksum, as an integer.

`u_int32_t lw6sys_checksum_whd` (*lw6sys_whd_t * `whd`*) [Function]

whd: a pointer to the wh struct to be processed

Creates a checksum from the given structure. Convenience function to save the hassle of passing a pointer to and the size of the `lw6sys_wh_t` struct each time, knowing that there are very often checksums calculated on it. Also avoids endianess issues.

Return value: the checksum, as an integer.

`u_int32_t lw6sys_checksum_xyz` (*lw6sys_xyz_t * `xyz`*) [Function]

xyz: a pointer to the xy struct to be processed

Creates a checksum from the given structure. Convenience function to save the hassle of passing a pointer to and the size of the `lw6sys_xy_t` struct each time, knowing that there are very often checksums calculated on it. Also avoids endianess issues.

Return value: the checksum, as an integer.

`void lw6sys_checksum_update` (*u_int32_t * `checksum`, unsigned char * `data`, int `len`*) [Function]

checksum: a pointer to the previous checksum

data: the data to process

len: the length, in bytes, of the data to process

Creates a checksum from the given data. The difference with `lw6sys_checksum` is that this one updates an existing checksum, thus enabling the programmer to call it sequentially and get a global checksum on different sources.

Return value: none.

`void lw6sys_checksum_update_str (`*u_int32_t * **checksum***, *char * **value**`)` [Function]

checksum: a pointer to the previous checksum

value: the string to process

Creates a checksum from the given string. The difference with `lw6sys_checksum_str` is that this one updates an existing checksum, thus enabling the programmer to call it sequentially and get a global checksum on different sources.

Return value: none.

`void lw6sys_checksum_update_int32 (`*u_int32_t * **checksum***, *int32_t **value**`)` [Function]

checksum: a pointer to the previous checksum

value: the integer to process

Creates a checksum from the given integer. The difference with `lw6sys_checksum_int32` is that this one updates an existing checksum, thus enabling the programmer to call it sequentially and get a global checksum on different sources.

Return value: none.

`void lw6sys_checksum_update_int64 (`*u_int32_t * **checksum***, *int64_t **value**`)` [Function]

checksum: a pointer to the previous checksum

value: the integer to process

Creates a checksum from the given integer. The difference with `lw6sys_checksum_int64` is that this one updates an existing checksum, thus enabling the programmer to call it sequentially and get a global checksum on different sources.

Return value: none.

`void lw6sys_checksum_update_whd (`*u_int32_t * **checksum***, *lw6sys_whd_t * **whd**`)` [Function]

checksum: a pointer to the previous checksum

whd: a pointer to the wh struct to be processed

Creates a checksum from the given structure. The difference with `lw6sys_checksum_whd` is that this one updates an existing checksum, thus enabling the programmer to call it sequentially and get a global checksum on different sources.

Return value: none.

`void lw6sys_checksum_update_xyz (`*u_int32_t * **checksum***, *lw6sys_xyz_t * **xyz**`)` [Function]

checksum: a pointer to the previous checksum

xyz: a pointer to the xy struct to be processed

Creates a checksum from the given structure. The difference with `lw6sys_checksum_xyz` is that this one updates an existing checksum, thus enabling the programmer to call it sequentially and get a global checksum on different sources.

Return value: none.

`u_int8_t lw6sys_color_float2char (`*float* `f``)` [Function]

f: the value to convert, from 0.0f to 1.0f

Converts a floating point value between 0.0f and 1.0f to its 8-bit equivalent between 0 and 255. Usefull in color conversion.

Return value: an integer between 0 and 255.

`float lw6sys_color_char2float (`*u_int8_t* `i``)` [Function]

i: the value to convert, from 0 to 255

Converts an 8-bit value between 0 and 255 to its floating-point equivalent between 0.0f and 1.0f. Usefull in color conversion.

Return value: a float between 0.0f and 1.0f.

`lw6sys_color_8_t lw6sys_color_f_to_8 (`*lw6sys_color_f_t* * `color_f`) [Function]

color_f: the color to convert

Converts a color from floating point format to the integer "0 to 255" common format. All fields (RGBA) are converted.

Return value: the color in 8-bit format.

`void lw6sys_color_8_to_f (`*lw6sys_color_f_t* * `color_f`, *lw6sys_color_8_t* `color_8`) [Function]

color_f: the converted color (pointer must point to writable memory)

color_8: the color to convert

Converts a color from the integer "0 to 255" common format to floating point format. All fields (RGBA) are converted.

Return value: none.

`u_int32_t lw6sys_color_f_to_i (`*lw6sys_color_f_t* * `color_f`) [Function]

color_f: the color to convert

Converts a color from floating point format to a single integer, where all fields (RGBA) are serialized. This serialization is endianess independant. Could be used directly by low-level libraries such as SDL.

Return value: the color serialized in an integer.

`u_int32_t lw6sys_color_8_to_i (`*lw6sys_color_8_t* `color_8`) [Function]

color_8: the color to convert

Converts a color from common "0 to 255" structured format to a single integer, where all fields (RGBA) are serialized. This serialization is endianess independant. Could be used directly by low-level libraries such as SDL.

Return value: the color serialized in an integer.

`void lw6sys_color_i_to_f (`*lw6sys_color_f_t* * `color_f`, *u_int32_t* `color_i`) [Function]

color_f: the converted color (point must point to writable memory)

color_i: the color to convert

Converts a color from a serialized integer format to a floating point structure.

Return value: none.

`lw6sys_color_8_t lw6sys_color_i_to_8 (`*u_int32_t* `color_i)` [Function]

color_i: the color to convert

Converts a color from a serialized integer format to a "0 to 255" based structure.

Return value: the converted color (structure).

`lw6sys_color_8_t lw6sys_color_a_to_8 (`*char ** `ascii)` [Function]

ascii: the color to convert

Converts a color from a human readable string to a "0 to 255" based structure. The string must be of the form "#RRGGBBAA" or "#RGB", in a general manner any HTML-valid value should work.

Return value: the converted color (structure).

`void lw6sys_color_a_to_f (`*lw6sys_color_f_t ** `color_f`, *char ** `ascii)` [Function]

color_f: the converted color (pointer must point to writable memory)

ascii: the color to convert

Converts a color from a human readable string to a float based structure. The string must be of the form "#RRGGBBAA" or "#RGB", in a general manner any HTML-valid value should work.

Return value: none.

`char * lw6sys_color_8_to_a (`*lw6sys_color_8_t* `color_8)` [Function]

color_8: the color to convert

Converts a color from a "0 - 255" integer based structure to its readable form "#RRGGBBAA". If alpha is 255 (0xFF), that is, if it's opaque, then the "AA" part is ommitted.

Return value: a newly allocated string.

`void lw6sys_color_rgb_to_hsv (`*lw6sys_color_hsv_t ** `color_hsv`, *lw6sys_color_8_t* `color_8)` [Function]

color_hsv: the target color, in HSV format

color_8: the source color, in RGB 256 format

Converts from HSV to RGB. Usefull for color manipulation, since most colors are stored in RGB but HSV is convenient for transformation. Alpha layer is kept as is.

Return value: none.

`lw6sys_color_8_t lw6sys_color_hsv_to_rgb (`*lw6sys_color_hsv_t ** `color_hsv)` [Function]

color_hsv: the source color, in HSV format

Converts from RGB to HSV. Usefull to make colors transformed in HSV format usable again by all display routines, which consume RGB. Alpha layer is kept as is.

Return value: the RGB color.

`lw6sys_color_8_t lw6sys_color_average` (*int* **`size`**, *lw6sys_color_8_t ** **`colors`**) [Function]

size: number of the color array (number of items)

colors: the colors to compute

Tries to find out the "average" color from an array of colors. The algorithm is far from perfect, but should output a color which reflects the colors passed in.

Return value: the (inexact) average color.

`lw6sys_color_8_t lw6sys_color_ponderate` (*lw6sys_color_8_t* **`color1`**, *lw6sys_color_8_t* **`color2`**, *float* **`coeff`**) [Function]

color1: first color

color2: second color

coeff: the ponderation coefficient

Tries to find a color between the two colors passed as an argument. The coefficient can be used, to set the relative weight of each color. Using 0 will return color1, 1 will return color2 and 0.5 will make an average between the two colors. Any value between 0 and 1 can be used.

Return value: the (inexact) ponderated color.

`float lw6sys_color_distance` (*lw6sys_color_8_t* **`color1`**, *lw6sys_color_8_t* **`color2`**) [Function]

color1: first color

color2: second color

Calculates the distance between two colors. The unit is arbitrary, a big value means "colors are different", 0 means they are the same. A distance of 1 corresponds to colors which have barely anything in common, but the result can still be greater than 1. Alpha layer is not taken in account.

Return value: the distance.

`void lw6sys_color_8_solid` (*lw6sys_color_8_t ** **`color`**) [Function]

color: the color to modify

Make a color "solid" that is make it not transparent at all.

Return value: none.

`void lw6sys_color_f_solid` (*lw6sys_color_f_t ** **`color`**) [Function]

color: the color to modify

Make a color "solid" that is make it not transparent at all.

Return value: none.

`int lw6sys_atoi` (*char ** **`str`**) [Function]

str: string to convert

Just a plain wrapper on `atoi`, it's here for API consistency.

Return value: an integer.

`int lw6sys_atob (char * str)` [Function]

str: string to convert

Transform a string into a boolean value. Accepts "0"/"1" in input, but also y/n, yes/no, true/false, on/off.

Return value: an integer, 0 or 1.

`float lw6sys_atof (char * str)` [Function]

str: string to convert

A wrapper on `atof`, makes sure the locale used is C (default) and won't change the decimal separator whatsoever. Usefull for serialization for instance.

Return value: a float.

`char * lw6sys_itoa (int value)` [Function]

value: the integer to convert

Converts an integer to a string, the advantage of this function is it allocates memory, and does the dirty job.

Return value: a newly allocated pointer, must be freed, may be NULL.

`char * lw6sys_btoa (int value)` [Function]

value: the boolean to convert

Converts a boolean to a string, the advantage of this function is it allocates memory, and does the dirty job.

Return value: a newly allocated pointer, must be freed, may be NULL.

`char * lw6sys_ftoa (float value)` [Function]

value: the float to convert

Converts a float to a string, the advantage of this function is it allocates memory, and does the dirty job.

Return value: a newly allocated pointer, must be freed, may be NULL.

`int lw6sys_debug_get ()` [Function]

Gets the debug mode.

`void lw6sys_debug_set (int mode)` [Function]

mode: the debug mode, 1 if set, 0 if not.

Sets the debug mode.

`void lw6sys_dump_clear (char * user_dir)` [Function]

user_dir: the user directory, where user can write data.

Clears the dump file. That is, resets it to a "0 byte" file.

Return value: none.

`int lw6sys_dump (char * user_dir, char * content)` [Function]

user_dir: the user directory, where user can write data.

content: the content to be written in the dump file.

Writes the dump file onto the disk. The dump is used for special error messages which do not really fit in the standard log, and require a special treatment. In pratice, it's used to log fatal script (Guile) errors.

Return value: 1 if success, 0 if failure.

`char lw6sys_env_separator_char ()` [Function]

Gets the ENV separator, that is, for instance, the character used to separate paths in environment variables. Typically, this would be ":" on GNU and ";" on Microsft platforms.

Return value: the ascii character code.

`char * lw6sys_env_separator_str ()` [Function]

Gets the ENV separator, that is, for instance, the character used to separate paths in environment variables. Typically, this would be ":" on GNU and ";" on Microsft platforms.

Return value: a pointer to a single 0-terminated character string which contains the character. Must not be freed.

`char * lw6sys_env_concat (`*char * value1, char * value2*`)` [Function]

value1: the left part to be concatenated

value2: the right part to be concatenated

Concatenates two values and puts the ENV separator, as returned by `lw6sys_env_separator_char` between them.

Return value: the concatenated string, must be freed.

`int lw6sys_env_exists (`*char * keyword*`)` [Function]

keyword: the keyword to be searched in the environment variables.

Searches environment variables for the given keyword. The keyword will be fixed so that all dashes "-" characters are replaced by underscores "_" characters. Characters will be changed to uppercase. Any non alphanumeric character will be replaced by "_". Finally, an "LW6_" prefix will be added. That is to say, calling this function with "my-param" will search for the "LW6_MY_PARAM" environment variable.

Return value: 1 if the environment variable exists, 0 if not.

`char * lw6sys_getenv (`*char * keyword*`)` [Function]

keyword: the keyword to be searched in the environment variables.

Searches environment variables for the given value. The keyword will be fixed so that all dashes "-" characters are replaced by underscores "_" characters. Characters will be changed to uppercase. Any non alphanumeric character will be replaced by "_". Finally, an "LW6_" prefix will be added. That is to say, calling this function with "my-param" will search for the "LW6_MY_PARAM" environment variable.

Return value: the value for the given keyword. May be NULL. Must be freed.

`int lw6sys_setenv (`*char * keyword, char * value*`)` [Function]

keyword: the environment variable to set

value: the value of the environment variable to set

Sets the environment variable to a given value. If value is NULL, variable is unset. Note that unlike lw6sys_getenv, this function does not transform the keyword into "LW6_..." before setting the value, so it's your responsability to call "lw6sys_keyword_as_env" if needed.

Return value: 1 if success, 0 if failed

`lw6sys_list_t * lw6sys_env_split` (*char * value*) [Function]

value: the value, a list of item separated by... the separator

Splits the environment value into a list of strings containing each element. All strings are dynamically allocated, but they will be freed automatically when the list is freed.

Return value: a list of strings.

`char * lw6sys_get_home` () [Function]

Gets the home directory of the user. Used internally to calculate the `user`-dir value. Note that Liquid War 6, by default, never stores files under '$HOME', instead it put things in '$HOME/.liquidwar6', that is 'user-dir'. If the environment variable 'HOME' is not set, will return '.'.

Return value: a newly allocated pointer, must be freed.

`char * lw6sys_get_username` () [Function]

Gets the name of the current user. Difference with the standard function `getlogin` is that this function will returned a dynamically allocated pointer, and provide a default value if it's undefined. Also, if will look at the content of the 'LOGNAME' environment variable if needed, and will even provide a default value.

Return value: a newly allocated pointer, must be freed.

`char * lw6sys_get_hostname` () [Function]

Gets the name of the current host. The name of the computer. Might not work perfectly, this function is just used to provide default values for player names and such things.

Return value: a newly allocated pointer, must be freed.

`int lw6sys_clear_file` (*char * filename*) [Function]

filename: absolute or relative filename

Clears a file, that is, make it a 0 byte file, empty, ready to be filled if needed. If this function is called successfully, program can reasonnably assume file will be writable during its execution.

Return value: 1 if success, 0 if failure.

`char * lw6sys_read_file_content` (*char * filename*) [Function]

filename: absolute or relative filename

Reads the content of a file, and returns it as a string. Note that content might or might not be ascii or binary, the function will however put a tailing 0 character at the end so that low-level standard C functions do not segfault when used with the returned value.

Return value: a newly allocated pointer, must be freed.

`int lw6sys_write_file_content (char * filename, char * content)` [Function]

filename: absolute or relative filename

content: the content to be written.

Writes the content into the file. Content is assumed to be a string, function will segfault if it's not correctly 0 terminated as in C string convention. So this function will not allow you to write down arbitrary binary data, however LW6 uses mostly text files to store information, and opaque binary data usage is not recommended.

`lw6sys_hash_t * lw6sys_hash_new (lw6sys_free_func_t free_func, int size)` [Function]

free_func: optional callback used to free memory when stored date is a pointer. Can be NULL when one stores non dynamically allocated data, such as an integer or a static array.

size: the estimated size of the hash table. Note that this is an estimation only. You could theorically fit 1000000 objects in a 3-sized hash. Problem -> this is inefficient, you'd better use an assoc or a bigger hash. If you store 3 elements in a 1000000-sized hash, you'll waste memory. It might be wise to use a prime number as the estimated size. 421 is prime ;)

Creates an empty hash. There's a difference between NULL and an empty hash.

Return value: a pointer to the newly allocated hash table. Must be freed with `lw6sys_hash_free`.

`void lw6sys_hash_free (lw6sys_hash_t * hash)` [Function]

hash: the hash to be freed.

The function will cascade delete all elements, using (if not NULL...) the callback passed when first creating the hash.

Return value: void

`int lw6sys_hash_has_key (lw6sys_hash_t * hash, char * key)` [Function]

hash: the hash to test

key: the key to search

Not a very fast function, since on a "big" hash, strcmp will be called internally until the key is found.

Return value: non-zero if there's an entry with the corresponding key.

`void * lw6sys_hash_get (lw6sys_hash_t * hash, char * key)` [Function]

hash: the hash to query

key: the key of which we want the value

Return value: a void pointer to the data contained in the hash. Note that the pointer on the actual data is returned, that is, if it's static data, you must not try to free it... As long as memory management is concerned, destroying the hash will actually free the data if needed.

`void lw6sys_hash_set (lw6sys_hash_t * hash, char * key, void * value)` [Function]

hash: the hash to modify

key: the key we want to updated

value: the new value

Sets a value in a hash table. The key pointer need not be persistent, it can be freed after affectation. In fact a new string will be created internally. This is not true for the value, it's hard to find way to copy "any object". So if you want a hash table of strings, key can disappear after calling this function, but not value. The function passed as free_func when creating the hash will be used to free stuff whenever needed (unset or free).

Return value: void

`void lw6sys_hash_unset` (*lw6sys_hash_t * **hash***, *char * **key***) [Function]

hash: the hash concerned

key: the key to unset

Clears an entry in a hash table. The callback passed when creating the hash will be called if needed, to free the data automatically.

Return value: void

`lw6sys_list_t * lw6sys_hash_keys` (*lw6sys_hash_t * **hash***) [Function]

hash: the hash to work on

Returns a list containing all the keys of the hash. The list must be free with lw6sys_list_free by the caller. This list copies all the keys of the hash, so it is safe to use it once the hash is deleted. However the keys will of course be of little interest in this case. But the program won't segfault.

Return value: the list of keys.

`void lw6sys_hash_map` (*lw6sys_hash_t * **hash***, [Function]
*lw6sys_assoc_callback_func_t **func***, *void * **func_data***)

hash: the hash to work on

func: a callback to call on each entry

func_data: a pointer on some data which will be passed to the callback

Executes a function on all hash items. The func_data parameter allows you to pass extra values to the function, such as a file handler or any variable which can not be inferred from list item values, and you of course do not want to make global...

Return value: void

`void lw6sys_hash_sort_and_map` (*lw6sys_hash_t * **hash***, [Function]
*lw6sys_assoc_callback_func_t **func***, *void * **func_data***)

hash: the hash to work on

func: a callback to call on each entry, may be NULL

func_data: a pointer on some data which will be passed to the callback

Executes a function on all hash items, like `lw6sys_hash_sort_and_map` but befor doing so, sorts all entries in alphabetical order.

Return value: void

`lw6sys_hash_t * lw6sys_hash_dup` (*lw6sys_hash_t* * ***hash***, *lw6sys_dup_func_t* ***dup_func***) [Function]

hash: the hash to duplicate, can be NULL

dup_func: the function which will be called to duplicate data

Duplicates an hash. All keys will be copied so that if the first hash is deleted, the duplicated one is fine. Additionnaly, dup_func will be called with all data fields. If dup_func is NULL, then data values will simply be copied. This is likely to be usefull when data is not dynamically allocated.

Returned value: a newly allocated hash.

`lw6sys_hexa_serializer_t * lw6sys_hexa_serializer_new` (*char* * ***hexa_string***) [Function]

hexa_string: an initialization string, can be NULL.

Creates an hexa serializer object. It can be initialized or not, if an initialization string is provided it must of course be valid hexadecimal ascii code, and all serialized content will simply be appended to it.

Return value: a newly allocated object.

`void lw6sys_hexa_serializer_free` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***) [Function]

hexa_serializer: an hexa serializer object

Frees an hexa serializer object.

Return value: none.

`void lw6sys_hexa_serializer_rewind` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***) [Function]

hexa_serializer: an hexa serializer object

Rewinds the serializer pointer, that is, make it point to start. Usefull before calling pop functions, when one wants to be sure to get the first object.

Return value: none.

`int lw6sys_hexa_serializer_eof` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***) [Function]

hexa_serializer: an hexa serializer object

Tests wether we're at EOF. Usefull when one wants to know if there's still some data or if all objects have been correctly popped.

Return value: 1 if at end of file, 0 if not.

`char * lw6sys_hexa_serializer_as_string` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***) [Function]

hexa_serializer: an hexa serializer object

Exports the current content of the serializer as a string. String can then safely be sent on the network, for instance. String is copied from internal value, so it's safe to use it after serializer has been freed or modified.

Return value: a newly allocated string, must be freed.

`int lw6sys_hexa_serializer_push_int64` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *int64_t* ***value***) [Function]

hexa_serializer: an hexa serializer object

value: value to push

Pushes a 64 bit integer in the serializer object.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_push_int32` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *int32_t* ***value***) [Function]

hexa_serializer: an hexa serializer object

value: value to push

Pushes a 32 bit integer in the serializer object.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_push_int16` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *int16_t* ***value***) [Function]

hexa_serializer: an hexa serializer object

value: value to push

Pushes a 16 bit integer in the serializer object.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_push_int8` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *int8_t* ***value***) [Function]

hexa_serializer: an hexa serializer object

value: value to push

Pushes an 8 bit integer in the serializer object.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_push_float` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *float* ***value***) [Function]

hexa_serializer: an hexa serializer object

value: value to push

Pushes a floating point value in the serializer object.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_push_str` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *char* * ***value***) [Function]

hexa_serializer: an hexa serializer object

value: value to push

Pushes a string in the serializer object. Note that the string is not directly copied in the serializer, instead all its characters are converted to their ASCII equivalent, then appended.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_push_xyz` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *lw6sys_xyz_t* **value**) [Function]

hexa_serializer: an hexa serializer object

value: value to push

Pushes a lw6sys_xyz_t structure in the serializer object. Calling this avoids calling push for 2 integers separately.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_push_whd` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *lw6sys_whd_t* **value**) [Function]

hexa_serializer: an hexa serializer object

value: value to push

Pushes a lw6sys_whd_t structure in the serializer object. Calling this avoids calling push for 2 integers separately.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_push_color` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *lw6sys_color_8_t* **value**) [Function]

hexa_serializer: an hexa serializer object

value: value to push

Pushes a color structure in the serializer object.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_pop_int64` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *int64_t* * **value**) [Function]

hexa_serializer: an hexa serializer object

value: value to pop (returned value)

Pops a 64 bit integer from the serializer object.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_pop_int32` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *int32_t* * **value**) [Function]

hexa_serializer: an hexa serializer object

value: value to pop (returned value)

Pops a 32 bit integer from the serializer object.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_pop_int16` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *int16_t* * **value**) [Function]

hexa_serializer: an hexa serializer object

value: value to pop (returned value)

Pops a 16 bit integer from the serializer object.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_pop_int8` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *int8_t* * ***value***) [Function]

hexa_serializer: an hexa serializer object

value: value to pop (returned value)

Pops an 8 bit integer from the serializer object.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_pop_float` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *float* * ***value***) [Function]

hexa_serializer: an hexa serializer object

value: value to pop (returned value)

Pops a floating point value from the serializer object.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_pop_str` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *char* ** ***value***) [Function]

hexa_serializer: an hexa serializer object

value: value to pop (returned value)

Pops a string from the serializer object. The returned value is a newly allocated pointer, which must be freed, you don't need to provide a buffer, just a valid pointer on a NULL pointer.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_pop_xyz` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *lw6sys_xyz_t* * ***value***) [Function]

hexa_serializer: an hexa serializer object

value: value to pop (returned value)

Pops a lw6sys_xyz_t structure from the serializer object. Avoids calling two integer pops.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_pop_whd` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *lw6sys_whd_t* * ***value***) [Function]

hexa_serializer: an hexa serializer object

value: value to pop (returned value)

Pops a lw6sys_whd_t structure from the serializer object. Avoids calling two integer pops.

Return value: 1 if success, 0 if failure

`int lw6sys_hexa_serializer_pop_color` (*lw6sys_hexa_serializer_t* * ***hexa_serializer***, *lw6sys_color_8_t* * ***value***) [Function]

hexa_serializer: an hexa serializer object

value: value to pop (returned value)

Pops a color from the serializer object.

Return value: 1 if success, 0 if failure

`void lw6sys_history_init ()` [Function]

Initializes the history system. Not initializing won't cause any segfault, but data will be inconsistent.

Return value: none.

`void lw6sys_history_register (`*char * msg*`)` [Function]

msg: the message to register.

Registers a message in the history log, that is, adds it.

Return value: none.

`char_ptr_t * lw6sys_history_get (`*int64_t timeout*`)` [Function]

timeout: the message age limit.

Get all the messages that are younger than timeout (in seconds).

Return value: a pointer on string pointers. May be NULL. Last pointer is NULL too, that's how you know the array is over.

`void lw6sys_history_free (`*char ** history*`)` [Function]

history: the data to free

Frees a pointer returned by `lw6sys_history_get`.

Return value: none.

`char * lw6sys_locale_to_utf8 (`*char * string*`)` [Function]

string: the string to convert

Used to force strings into UTF-8 mode, this is basically to match the TTF font settings used when displaying stuff on OpenGL. Indeed, in this case, the standard _ gettext function won't work, we need to force UTF-8 mode. If the locale is UTF-8, then function does nothing, but at least it's transparent usage won't hurt.

Returned value: a newly allocated string, always in UTF-8 no matter what the locale is.

`u_int16_t lw6sys_generate_id_16 ()` [Function]

Long 16-bit ID generator, calls the random function internally. As usual, those are not perfect random numbers, however the function implementation emphasizes more on 'real randomness' rather than relying on performance. Generating twice the same number should be fairly rare.

`u_int32_t lw6sys_generate_id_32 ()` [Function]

Long 32-bit ID generator, calls the random function internally. As usual, those are not perfect random numbers, however the function implementation emphasizes more on 'real randomness' rather than relying on performance. Generating twice the same number should be fairly rare.

`u_int64_t lw6sys_generate_id_64 ()` [Function]

Long 64-bit ID generator, calls the random function internally. As usual, those are not perfect random numbers, however the function implementation emphasizes more on 'real randomness' rather than relying on performance. Generating twice the same number should be fairly rare.

`int lw6sys_check_id_16 (`*u_int16_t* `id_16``)` [Function]
id_16: the id to check

Checks wether the given id is a valid 16-bit id.

Return value: 1 if OK, 0 if not a valid id.

`int lw6sys_check_id_32 (`*u_int32_t* `id_32``)` [Function]
id_32: the id to check

Checks wether the given id is a valid 32-bit id.

Return value: 1 if OK, 0 if not a valid id.

`int lw6sys_check_id_64 (`*u_int64_t* `id_64``)` [Function]
id_64: the id to check

Checks wether the given id is a valid 64-bit id.

Return value: 1 if OK, 0 if not a valid id.

`int lw6sys_check_id (`*u_int64_t* `id``)` [Function]
id: the id to check

Checks wether the given id is a valid id (16, 32 or 64-bit).

Return value: 1 if OK, 0 if not a valid id.

`char * lw6sys_id_ltoa (`*u_int64_t* `id``)` [Function]
id: the id to convert

Transform an id into its string representation. Error checking is done, if the id is invalid, returned value is NULL. All ids (16, 32 and 64-bit) are handled.

Return value: a newly allocated string, might be NULL.

`u_int64_t lw6sys_id_atol (`*char * id*`)` [Function]
id: the id to convert

Transform an id into a long integer. Error checking is done, if the id is invalid, returned value is 0. All ids (16, 32 and 64-bit) are handled.

Return value: the id as a long integer, 0 if incorrect source id.

`char * lw6sys_keyword_as_key (`*char * keyword*`)` [Function]
keyword: the keyword to transform

Transforms a keyword into a "key", that is, removes all heading dashes, switches to lowercase, and other stuff. This is used internally to match options and config file parameters, for instance.

Return value: a newly allocated pointer, must be freed.

`char * lw6sys_keyword_as_arg (`*char * keyword*`)` [Function]
keyword: the keyword to transform

Transforms a keyword into a command-line parameter to be matched. Does the same as `lw6sys_keyword_as_key`, and adds a "−" prefix.

Return value: a newly allocated pointer, must be freed.

`char * lw6sys_keyword_as_env` (*char * **keyword***) [Function]
keyword: the keyword to transform

Transforms a keyword into the corresponding environment variable name. It will uppercase the name, replace `"-"` by `"_"`, and add a `"LW6_"` prefix. `"my-param"` will become `"LW6_MY_PARAM"`.

Return value: a newly allocated pointer, must be freed.

`char * lw6sys_keyword_as_xml` (*char * **keyword***) [Function]
keyword: the keyword to transform

Transforms a keyword into the corresponding config file entry. In practice, just the same as `lw6sys_keyword_as_key`.

Return value: a newly allocated pointer, must be freed.

`lw6sys_list_t * lw6sys_list_new` (*lw6sys_free_func_t **free_func***) [Function]
free_func: a callback which will be called on data when freeing the list

Creates an empty list. There's a difference between NULL and an empty list. The empty list would (in Scheme) be '() whereas NULL corresponds to undefined `"is not a list and will generate errors if you ever call list functions on it"`.

Return value: a pointer to the created object, may be NULL.

`void lw6sys_list_free` (*lw6sys_list_t * **list***) [Function]
list: the list to delete.

Delete a list, this will cascade delete all the following items in the list.

Return value: none.

`lw6sys_list_t * lw6sys_list_next` (*lw6sys_list_t * **list***) [Function]
list: the current position in the list

It's safer to call this rather than dig right into the internals of the list.

Return value: a new position in the list, may be NULL.

`int lw6sys_list_is_empty` (*lw6sys_list_t * **list***) [Function]
list: the list we want informations about

Checks wether the list is empty or not. Note that being empty and being NULL is not the same. An empty list is a valid pointer on a list where there's no item, a NULL pointer is not a list at all. Do *NOT* call this function on NULL.

Return value: 1 if empty, 0 if there is at list one item.

`int lw6sys_list_length` (*lw6sys_list_t * **list***) [Function]
list: the list we want informations about

Calculates the length of the list. This is a performance killer for lists are inadapted to this. But it can still be usefull.

Return value: the number of elements, 0 is none (empty list).

`void lw6sys_list_map (`*lw6sys_list_t * **list**, lw6sys_list_callback_func_t **func**, void * **func_data**`)` [Function]

list: the list where elements will be taken

func: the function which will be executed

func_data: additionnal data to be passed to `func`

Executes a function on all list items. The `func_data` parameter allows you to pass extra values to the function, such as a file handler or any variable which can not be inferred from list item values, and you of course do not want to make global... Not as convenient as a real "for each" construct as can be found in any modern langage, but does the job. No return value, if you really want one, pass a structure in `func_data` and modify something in it on success, failure, whatever.

Return value: none.

`void lw6sys_list_push_front (`*lw6sys_list_t ** **list**, void * **data**`)` [Function]

list: a pointer to the list (pointer on pointer, read/write value)

data: the data to be pushed

Pushes data on the list. The `free_func` function is copied from the previous element. The pointer on the list is changed "in place" (in/out). Note that if there's a `malloc` problem it might end-up being NULL... This should be rare but it *can* happen. You cannot push something else than a pointer, pushing an int is a very bad idea. Push a pointer on the integer, and ensure it's always there, or `malloc` it and pass `lw6sys_free_callback` when creating the list. If you think you can cast an integer into a pointer, think 64-bit machines...

Return value: none.

`void * lw6sys_list_pop_front (`*lw6sys_list_t ** **list**`)` [Function]

list: a pointer to the list (pointer on pointer, read/write value)

Pops data from the list, the returned value is what was passed to list_push. The pointer on the list is changed "in place" (in/out). When data is popped, that needs some freeing (i.e. free_func was not NULL when creating the list) then it's the responsibility of the caller to free it when popping it. One popped it's not freed, but it's out of the list scope. Of course the lw6sys_list_t is freed, but not the data. If you happen to store non-NULL data in your list, you can call this function without bothering calling `lw6sys_list_is_empty` and assume that when you get NULL, there's no data left. At this stage, the list won't exist anymore BTW, you won't even need to free it. The idea is: popping a list which has no elements left (empty list) destroys the list and returns NULL.

Return value: a pointer on the popped data, whatever you pushed.

`void lw6sys_list_push_back (`*lw6sys_list_t ** **list**, void * **data**`)` [Function]

list: a pointer to the list (pointer on pointer, read/write value)

data: the data to be pushed

Pushes data on the list. The `free_func` function is copied from the previous element. The pointer on the list is changed "in place" (in/out). Note that if there's a `malloc` problem it might end-up being NULL... This should be rare but it *can* happen.

You cannot push something else than a pointer, pushing an int is a very bad idea. Push a pointer on the integer, and ensure it's always there, or `malloc` it and pass `lw6sys_free_callback` when creating the list. If you think you can cast an integer into a pointer, think 64-bit machines...

Return value: none.

`void * lw6sys_list_pop_back (`*lw6sys_list_t ** `list`*`)` [Function]

list: a pointer to the list (pointer on pointer, read/write value)

Pops data from the list, the returned value is what was passed to list_push. The pointer on the list is changed "in place" (in/out). When data is popped, that needs some freeing (i.e. free_func was not NULL when creating the list) then it's the responsibility of the caller to free it when popping it. One popped it's not freed, but it's out of the list scope. Of course the lw6sys_list_t is freed, but not the data. If you happen to store non-NULL data in your list, you can call this function without bothering calling `lw6sys_list_is_empty` and assume that when you get NULL, there's no data left. At this stage, the list won't exist anymore BTW, you won't even need to free it. The idea is: popping a list which has no elements left (empty list) destroys the list and returns NULL.

Return value: a pointer on the popped data, whatever you pushed.

`lw6sys_list_t * lw6sys_list_dup (`*lw6sys_list_t * `list`*, *lw6sys_dup_func_t `dup_func`*`)` [Function]

list: the list to duplicate, can be NULL

dup_func: the function which will be called to duplicate data

Duplicates a list. All data will be copied so that if the first list is deleted, the duplicated one is fine. Addtionnally, dup_func will be called to filter all data, and possibly allocated new pointers if needed, for instance. If dup_func is NULL, then data values will simply be copied. This is likely to be usefull when data is not dynamically allocated.

Returned value: a newly allocated list.

`void lw6sys_log_set_file (`*char * `filename`*`)` [Function]

filename: the name of the log file.

Sets up the log file. Until you call this function, messages all logged to the default log file, as returned by the `lw6sys_get_default_log_file` function.

Return value: void

`void lw6sys_log_clear (`*char * `filename`*`)` [Function]

filename: the name of the log file.

Clears the log file, this function would typically be called at the beginning of the program. If filename is NULL, then the default log file is cleared.

Return value: void

`void lw6sys_log (`*int `level_id`, char * `file`, int `line`, char * `fmt`,* `...`*`)` [Function]

level_id: the log level to use. Possible values are, by order, LW6SYS_LOG_ERROR_ID (0), LW6SYS_LOG_WARNING_ID (1), LW6SYS_LOG_NOTICE_ID (2), LW6SYS_LOG_INFO_ID (3) and LW6SYS_LOG_DEBUG_ID (4).

file: the name of the source file where the function is called, one can use __FILE__

line: the line in the source file where the function is called, one can use __LINE__

fmt: a printf-like format string ...: printf-like arguments, corresponding to `fmt`.

This function is usually called with the first three arguments packed into a single macro. A typical use is "lw6sys_log(LW6SYS_LOG_WARNING,"sys","problem *s d*","foo",1)". The `LW6SYS_LOG_WARNING` macro expands and fills the first 3 args, so there's no need to type __FILE__ and __LINE__ again and again. Note that this function will reset errno. The idea is to call it whenever there's something to do with errno (if you deal with errno, it's a good habit to log it) then errno is cleared so that it won't interfere with next log messages.

`void lw6sys_log_critical (`*char * **fmt***, `...`)` [Function]

fmt: a printf-like format string ...: printf-like arguments, corresponding to `fmt`.

This function is a special log function which will dump informations on the console only, without opening any log file whatsoever. The idea is that it's a "never fail" function. Additionnally, it will never return but quit the program. This can be used as an ultimate emergency function, use it when the program won't run for sure, and displaying an immediate error message is the only issue.

`int lw6sys_log_get_level ()` [Function]

`void lw6sys_log_set_level (`*int **level***`)` [Function]

level: the log level, integer between 0 & 4. 4 is very verbose (debug), 0 displays errors only.

`void * lw6sys_malloc (`*int **size***, *char * **file***, *int **line***`)` [Function]

size: number of bytes to allocate.

file: name of the file calling the function, use `__FILE__`

line: line in the file calling the function, use `__LINE__`

This is a wrapper over the standard `malloc` function. Additionnally it will keep track of the call with an internal program-wide counter, thus enabling memory leak checks. You should not use this function directly but use the macro `LW6SYS_MALLOC` which has the same syntax, without the last two parameters, which are automatically provided by macro expansion.

Return value: the newly allocated pointer. Data is not initialized.

`void * lw6sys_calloc (`*int **size***, *char * **file***, *int **line***`)` [Function]

size: number of bytes to allocate.

file: name of the file calling the function, use `__FILE__`

line: line in the file calling the function, use `__LINE__`

This is a wrapper over the standard `calloc` function. Additionnally it will keep track of the call with an internal program-wide counter, thus enabling memory leak checks. You should not use this function directly but use the macro `LW6SYS_CALLOC` which has the same syntax, without the last two parameters, which are automatically provided by macro expansion.

Return value: the newly allocated pointer. Data is filled with zeros.

`void * lw6sys_realloc (void * ptr, int size, char * file, int line)` [Function]

ptr: the pointer to reallocate.

size: number of bytes to allocate.

file: name of the file calling the function, use `__FILE__`

line: line in the file calling the function, use `__LINE__`

This is a wrapper over the standard `realloc` function. You should not use this function directly but use the macro `LW6SYS_REALLOC` which has the same syntax, without the last two parameters, which are automatically provided by macro expansion.

Return value: the newly allocated pointer.

`void lw6sys_free (void * ptr, char * file, int line)` [Function]

ptr: the pointer to free.

file: name of the file calling the function, use `__FILE__`

line: line in the file calling the function, use `__LINE__`

This is a wrapper over the standard `free` function. Additionnally it will keep track of the call with an internal program-wide counter, thus enabling memory leak checks. You should not use this function directly but use the macro `LW6SYS_FREE` which has the same syntax, without the last two parameters, which are automatically provided by macro expansion.

Return value: none.

`void lw6sys_free_callback (void * ptr)` [Function]

ptr: the pointer to free.

This is a callback to be used when the `lw6sys_free` does not fit. A good example is a list, which, to free its elements, requires you to provide a callback that only takes 1 arg, the pointer to free. Problem, `lw6sys_free` takes 3 args. And the `LW6SYS_FREE` macro is not usable in such a context. And you can't use standard `free` either for it would mess up the `malloc` / `free` automatical count which is so convenient to track memory leaks. So this callback is here, it's only drawback is that in case of an error, the error will not be reported with the real file and line parameters. It's still better than nothing.

Return value: none.

`int lw6sys_megabytes_available ()` [Function]

Gives a raw approximation of available memory, in megabytes. Value is to be taken with distance, but it can give good hints when system is running short of ressources.

Return value: number of megabytes (physical memory) available.

`int lw6sys_is_big_endian ()` [Function]

Checks the endianess of the machine. PPC is big endian, for instance.

Return value: 1 if system is big endian, 0 if little endian.

`int lw6sys_is_little_endian ()` [Function]

Checks the endianess of the machine. x86 is little endian, for instance.

Return value: 1 if system is little endian, 0 if big endian.

`int lw6sys_check_types_size ()` [Function]
Checks of common types and usefull structures, this is a debugging function which helps finding compiler strange behaviors and programmer's bad intuitions.

Return value: 1 if everything is OK, 0 if error.

`void * lw6sys_mutex_create ()` [Function]
Creates a mutex object.

Return value: newly allocated pointer.

`void lw6sys_mutex_destroy (`*void * mutex*`)` [Function]
mutex: the mutex to destroy.

Destroys a mutex object.

Return value: none.

`int lw6sys_mutex_lock (`*void * mutex*`)` [Function]
mutex: the mutex to use

Locks the mutex. Note that this should never fail unless there's a serious initialization problem, instead, function will wait forever until mutex is released.

Return value: 1 if success, 0 if failure.

`int lw6sys_mutex_trylock (`*void * mutex*`)` [Function]
mutex: the mutex to use

Tries to locks the mutex. That is, tells wether mutex can be locked immediately or not. Note that this does not mean there's 100% chance next call to lock will terminated immediately, since lock can still be acquired by another thread.

Return value: 1 if mutex unlocked, 0 if locked or error.

`int lw6sys_mutex_unlock (`*void * mutex*`)` [Function]
mutex: the mutex to use

Unlocks a mutex.

Return value: 1 if sucess, 0 if error.

`int lw6sys_get_mutex_lock_count ()` [Function]
Returns how many mutexes have been locked since program start. Usefull for sanity checking when debugging.

Return value: number of calls to lock

`int lw6sys_get_mutex_unlock_count ()` [Function]
Returns how many mutexes have been unlocked since program start. Usefull for sanity checking when debugging.

Return value: number of calls to unlock

`int lw6sys_check_mutex_count ()` [Function]
Checks wether unlock has been called as many times as lock. Usefull for sanity checking when debugging.

Return value: 1 if OK, 0 if inconsistency.

`int lw6sys_true ()` [Function]
Function which returns always true, that is, something different than 0.

`int lw6sys_false ()` [Function]
Function which returns always false, that is, 0. This can seem totally useless but it does have some utility. It's used for instance to "fool" the compiler and force it to compile and link functions in binaries, so that, afterwards, dynamically loaded .so files can find in the main binary some functions which would otherwise be stripped during the final link.

`char * lw6sys_get_default_user_dir ()` [Function]
Returns the default user directory. Note that this value is not static, it can depend, for instance, of the environment variable `HOME`.

Return value: a newly allocated string.

`char * lw6sys_get_default_config_file ()` [Function]
Returns the default config file. Note that this value is not static, it can depend, for instance, of the environment variable `HOME`.

Return value: a newly allocated string.

`char * lw6sys_get_default_log_file ()` [Function]
Returns the default log file. Note that this value is not static, it can depend, for instance, of the environment variable `HOME`.

Return value: a newly allocated string.

`char * lw6sys_get_default_prefix ()` [Function]
Returns the default prefix, could be /usr/local for instance.

Return value: a newly allocated string.

`char * lw6sys_get_default_mod_dir ()` [Function]
Returns the default module directory (dynamically loaded libraries).

Return value: a newly allocated string.

`char * lw6sys_get_default_data_dir ()` [Function]
Returns the default data directory.

Return value: a newly allocated string.

`char * lw6sys_get_default_map_dir ()` [Function]
Returns the default map directory.

Return value: a newly allocated string.

`char * lw6sys_get_default_map_path ()` [Function]
Returns the default map path, which can be composed of several directories.

Return value: a newly allocated string.

`char * lw6sys_get_default_script_file ()` [Function]
Returns the default script file.

Return value: a newly allocated string.

`void lw6sys_options_log_defaults ()` [Function]

Logs all default values to log file. Usefull for debugging, to know where the program is searching for its informations.

`char * lw6sys_get_cwd ()` [Function]

Returns the current working directory (absolute path).

Return value: a newly allocated string.

`char * lw6sys_get_run_dir (`*int* `argc`*, char * []* `argv`*)* [Function]

argc: argc, number of arguments, as given to `main`

argv: argv, pointer to arguments, as given to `main`

Returns the binary directory, that is, the directory the binary is stored in. This is calculated dynamically, by interpreting command-line arguments.

Return value: a newly allocated string.

`char * lw6sys_get_user_dir (`*int* `argc`*, char * []* `argv`*)* [Function]

argc: argc, number of arguments, as given to `main`

argv: argv, pointer to arguments, as given to `main`

Returns the user dir, taking in account command-line and environment variables. However config file content has no impact on the result.

Return value: a newly allocated string.

`char * lw6sys_get_config_file (`*int* `argc`*, char * []* `argv`*)* [Function]

argc: argc, number of arguments, as given to `main`

argv: argv, pointer to arguments, as given to `main`

Returns the config file, taking in account command-line and environment variables. However config file content has no impact on the result.

Return value: a newly allocated string.

`char * lw6sys_get_log_file (`*int* `argc`*, char * []* `argv`*)* [Function]

argc: argc, number of arguments, as given to `main`

argv: argv, pointer to arguments, as given to `main`

Returns the log file, taking in account command-line and environment variables. However config file content has no impact on the result.

Return value: a newly allocated string.

`char * lw6sys_get_prefix (`*int* `argc`*, char * []* `argv`*)* [Function]

argc: argc, number of arguments, as given to `main`

argv: argv, pointer to arguments, as given to `main`

Returns the prefix, taking in account command-line and environment variables. However config file content has no impact on the result.

Return value: a newly allocated string.

`char * lw6sys_get_mod_dir` (*int* `argc`, *char * []* `argv`) [Function]

argc: argc, number of arguments, as given to `main`

argv: argv, pointer to arguments, as given to `main`

Returns the mod dir (modules, shared .so), taking in account command-line and environment variables. However config file content has no impact on the result.

Return value: a newly allocated string.

`char * lw6sys_get_data_dir` (*int* `argc`, *char * []* `argv`) [Function]

argc: argc, number of arguments, as given to `main`

argv: argv, pointer to arguments, as given to `main`

Returns the data dir, taking in account command-line and environment variables. However config file content has no impact on the result.

Return value: a newly allocated string.

`char * lw6sys_get_map_dir` (*int* `argc`, *char * []* `argv`) [Function]

argc: argc, number of arguments, as given to `main`

argv: argv, pointer to arguments, as given to `main`

Returns the map dir, taking in account command-line and environment variables. However config file content has no impact on the result.

Return value: a newly allocated string.

`char * lw6sys_get_map_path` (*int* `argc`, *char * []* `argv`) [Function]

argc: argc, number of arguments, as given to `main`

argv: argv, pointer to arguments, as given to `main`

Returns the map path, taking in account command-line and environment variables. However config file content has no impact on the result. Map path can contain several directories.

Return value: a newly allocated string.

`char * lw6sys_get_script_file` (*int* `argc`, *char * []* `argv`) [Function]

argc: argc, number of arguments, as given to `main`

argv: argv, pointer to arguments, as given to `main`

Returns the script file, taking in account command-line and environment variables. However config file content has no impact on the result.

Return value: a newly allocated string.

`void lw6sys_options_log` (*int* `argc`, *char * []* `argv`) [Function]

argc: argc, number of arguments, as given to `main`

argv: argv, pointer to arguments, as given to `main`

Logs all the main options values which are not config-file dependant but depend on built-in defaults, command-line arguments and environment variables. Usefull to debug and know where the program is searching for things.

`int lw6sys_file_exists (char * filename)` [Function]

filename: the file to test

Tests the existence of a file on the filesystem. File is considered to exists if it's at least readable.

Return value: 1 if OK, 0 if file doesn't exist or can't be read.

`int lw6sys_dir_exists (char * dirname)` [Function]

dirname: the directory to test

Tests the existence of a directory on the filesystem.

Return value: 1 if OK, 0 if directory doesn't exist.

`int lw6sys_create_dir (char * dirname)` [Function]

dirname: the directory to create

Creates a directory, performing sanity checks such as verifying the directory really exists after being created.

Return value: 1 if OK, 0 if error.

`int lw6sys_create_dir_silent (char * dirname)` [Function]

dirname: the directory to create

Creates a directory like `lw6sys_create_dir` but this function is silent in the sense that it won't log any error. Usefull to create the log directory itself, for instance, and avoid infinite loops on error.

Return value: 1 if OK, 0 if error.

`char * lw6sys_path_add_slash (char * path)` [Function]

path: a path

Adds a slash, or in a general manner, a directory separator, at the end of a path, if needed. So /foo/bar will become /foo/bar/ but /bar/foo/ will remain /bar/foo/.

Return value: a newly allocated string, must be freed.

`char * lw6sys_path_strip_slash (char * path)` [Function]

path: a path

Strips the slash, or in a general manner, the directory separator, at the end of a path, if needed. So /foo/bar/ will become /foo/bar but /bar/foo will remain /bar/foo.

Return value: a newly allocated string, must be freed.

`char * lw6sys_path_concat (char * path1, char * path2)` [Function]

path1: left part of the path

path2: right part of the path

Concatenates 2 parts of a path. Function will try to avoid stupid "double-slash" when concatenating /foo/ with /bar/ and conversely insert a directory separator when concatenating /foo with bar/.

Return value: a newly allocated string, must be freed.

`lw6sys_list_t * lw6sys_path_split` (*char * path*) [Function]

path: a path

Splits a path into all its parts. For instance /boo/bar/foo2/bar2 returns a 4 elements list. This is more than a plain split, for heading and tailing slashes will be ignored, and various path separators will be interpreted (depends on platform).

Return value: a list containing 0-terminated strings.

`int lw6sys_path_is_relative` (*char * path*) [Function]

path: a path

Checks wether a path is relative or absolute.

Return value: 1 if relative, 0 if absolute.

`int lw6sys_path_is_cwd` (*char * path*) [Function]

path: a path

Checks wether a path is "." or not. Will also trap "" and "./".

Return value: 1 if relative, 0 if absolute.

`char * lw6sys_path_parent` (*char * path*) [Function]

path: a path

Returns the parent path. That will return /foo when given /foo/bar in input.

Return value: a newly allocated string, must be freed.

`char * lw6sys_path_unparent` (*char * path*) [Function]

path: a path

Given the ../foo/bar path, will return foo/bar. Usefull to get rid of heading ../ when a path is known to start with it.

Return value: a newly allocated string, must be freed.

`char * lw6sys_path_unparent_no_malloc` (*char * path*) [Function]

path: a path

Given the ../foo/bar path, will return foo/bar. Usefull to get rid of heading ../ when a path is known to start with it. This is different from `lw6sys_path_unparent` just because the result is not dynamically allocated and copied from source.

Return value: a pointer which points somewhere within the string passed as an argument.

`void lw6sys_print_xml_header` (*FILE * f*, *char * comment*) [Function]

f: file to output content to

Prints a standard Liquid War compliant XML header in the given file.

Return value: none.

`void lw6sys_print_xml_footer` (*FILE * f*) [Function]

f: file to output content to

Prints a standard Liquid War 6 compliant XML footer in the given file.

Return value: none.

`void lw6sys_progress_default (`*lw6sys_progress_t * **progress**, float * **value**`)` [Function]

progress: the progress struct to initialize

value: the value to point to

Sets a progress struct to default values, that is, ranging from 0.0f to 1.0f.

Return value: none.

`void lw6sys_progress_update (`*lw6sys_progress_t * **progress**, int **min**, int **max**, int **value**`)` [Function]

progress: the progress struct to update

min: the min value

max: the max value

value: the current value

Updates a progress struct. This is typically the function used by a callback to show the progress of a process. Note that this is note an initializer. Rather, the progress struct was initialized before, and this call is done in a loop with min being 0, max being the last value in the loop, and value the current index in the loop. NULL pointers correctly handled internally, so call this with any parameters, it's safe.

Return value: none.

`void lw6sys_progress_split (`*lw6sys_progress_t * **progress1**, lw6sys_progress_t * **progress2**, lw6sys_progress_t * **progress_src**`)` [Function]

progress1: the first part of the splitted progress progress2: the second part of the splitted progress progress_src: the progress to split

Utility function to split a progress struct, that is, if a progress was ranging from a to b, make 2 progress structs, ranging from a to c and from c to b, c being between a and b.

Return value: none

`void lw6sys_progress_split3 (`*lw6sys_progress_t * **progress1**, lw6sys_progress_t * **progress2**, lw6sys_progress_t * **progress3**, lw6sys_progress_t * **progress_src**`)` [Function]

progress1: the first part of the splitted progress progress2: the second part of the splitted progress progress3: the third part of the splitted progress progress_src: the progress to split

Utility function to split a progress struct, this one will split it into 3 equal parts.

Return value: none

`void lw6sys_progress_split4 (`*lw6sys_progress_t * **progress1**, lw6sys_progress_t * **progress2**, lw6sys_progress_t * **progress3**, lw6sys_progress_t * **progress4**, lw6sys_progress_t * **progress_src**`)` [Function]

progress1: the first part of the splitted progress progress2: the second part of the splitted progress progress3: the third part of the splitted progress progress4: the fourth part of the splitted progress progress_src: the progress to split

Utility function to split a progress struct, this one will split it into 4 equal parts.

Return value: none

`void lw6sys_progress_begin (`*lw6sys_progress_t * **progress***`)` [Function]
progress: the progress to update

Sets the progress to its min value, NULL values correctly handled.

Return value: none

`void lw6sys_progress_half (`*lw6sys_progress_t * **progress***`)` [Function]
progress: the progress to update

Sets the progress to the average between min and max, NULL values correctly handled.

Return value: none

`void lw6sys_progress_end (`*lw6sys_progress_t * **progress***`)` [Function]
progress: the progress to update

Sets the progress to its max value, NULL values correctly handled.

Return value: none

`u_int32_t lw6sys_random (`*u_int32_t* ***range***`)` [Function]
range: the high limit for random generated numbers. If you want random numbers between 0 and 5, set this to 6.

Wrapper over standard random function. This one is thread safe. This idea is not to provide cryptographic-proof random numbers, rather generate sequences which are random enough to generate unique server ids and such things. The function is initialized on its first call, and results depend on timestamp, host name, user name, and memory available.

`float lw6sys_random_float (`*float* ***min***, *float* ***max***`)` [Function]
min: the min value, as a float

max: the max value, as a float

Returns a random float number between min & max. Can be equal to min or max.

`int lw6sys_sdl_register ()` [Function]
Function used to avoid initializing SDL several times in a program. AFAIK Allegro has a `was_init` function, but SDL doesn't. With this function - which every LW6 sub-module should use - one can know globally, for the whole program, wether SDL has been initialized or not.

`int lw6sys_sdl_unregister ()` [Function]
Call this whenever you are done with SDL and exit it, so that the `lw6sys_sdl_register` function works correctly.

Return value: 1 if SDL needs to be unregistered, that is, if it has already been initialized, else 0.

`void lw6sys_serialize_int64 (`*unsigned char * **data**, int64_t* `value`*`)`* [Function]
data: pointer to the data, must contain at least 8 bytes of writable space

value: the integer to serialize

Serializes a 64-bit integer in a byte buffer. Result is not dependant on machine endianess. Typically used for checksums or high-level serializations.

`int64_t lw6sys_unserialize_int64` (*unsigned char* * `data`) [Function]

data: pointer to the data, must contain at least 8 bytes

Recovers a 64-bit integer from a byte buffer created, for instance, with `lw6sys_serialize_int64`.

`void lw6sys_serialize_int32` (*unsigned char* * `data`, *int32_t* `value`) [Function]

data: pointer to the data, must contain at least 4 bytes of writable space

value: the integer to serialize

Serializes a 32-bit integer in a byte buffer. Result is not dependant on machine endianess. Typically used for checksums or high-level serializations.

`int32_t lw6sys_unserialize_int32` (*unsigned char* * `data`) [Function]

data: pointer to the data, must contain at least 4 bytes

Recovers a 32-bit integer from a byte buffer created, for instance, with `lw6sys_serialize_int32`.

`void lw6sys_serialize_int16` (*unsigned char* * `data`, *int16_t* `value`) [Function]

data: pointer to the data, must contain at least 2 bytes of writable space

value: the integer to serialize

Serializes a 16-bit integer in a byte buffer. Result is not dependant on machine endianess. Typically used for checksums or high-level serializations.

`int16_t lw6sys_unserialize_int16` (*unsigned char* * `data`) [Function]

data: pointer to the data, must contain at least 2 bytes

Recovers a 16-bit integer from a byte buffer created, for instance, with `lw6sys_serialize_int16`.

`int lw6sys_shape_check_min_max_whd` (*lw6sys_whd_t* * `shape`, *int* `min_wh`, *int* `max_wh`, *int* `max_d`) [Function]

shape: the dimensions to control

min_wh: the min value for w and h

max_wh: the max value for w and h

max_d: the max value for d

Will check wether the given shape respects some basic constraints, being not to small and not too big.

Return value: 1 if OK, 0 if not.

`int lw6sys_shape_check_pos` (*lw6sys_whd_t* * `shape`, *lw6sys_xyz_t* * `pos`) [Function]

shape: the boundary box

pos: the position

Checks wether position is within the given boundary box.

Return value: 1 if OK, 0 if not.

`int lw6sys_sort_int_callback (`*lw6sys_list_t ** `list_a`, lw6sys_list_t ** `list_b`*`)`` [Function]

list_a: pointer to a list of int item

list_b: pointer to a list of int item

A typicall sort callback function, can be passed to `lw6sys_sort` to sort a list of integers.

Return value: -1 if `list_a < list_b` , 0 if `list_a == list_b`, 1 if `list_a > list_b`

`int lw6sys_sort_int_desc_callback (`*lw6sys_list_t ** `list_a`, lw6sys_list_t ** `list_b`*`)`` [Function]

list_a: pointer to a list of int item

list_b: pointer to a list of int item

A typicall sort callback function, can be passed to `lw6sys_sort` to sort a list of integers. This one will sort in reverse mode.

Return value: 1 if `list_a < list_b` , 0 if `list_a == list_b`, -1 if `list_a > list_b`

`int lw6sys_sort_float_callback (`*lw6sys_list_t ** `list_a`, lw6sys_list_t ** `list_b`*`)`` [Function]

list_a: pointer to a list of float item

list_b: pointer to a list of float item

A typicall sort callback function, can be passed to `lw6sys_sort` to sort a list of floating point numbers.

Return value: -1 if `list_a < list_b` , 0 if `list_a == list_b`, 1 if `list_a > list_b`

`int lw6sys_sort_float_desc_callback (`*lw6sys_list_t ** `list_a`, lw6sys_list_t ** `list_b`*`)`` [Function]

list_a: pointer to a list of float item

list_b: pointer to a list of float item

A typicall sort callback function, can be passed to `lw6sys_sort` to sort a list of floating point numbers. This one will sort in reverse mode.

Return value: 1 if `list_a < list_b` , 0 if `list_a == list_b`, -1 if `list_a > list_b`

`int lw6sys_sort_str_callback (`*lw6sys_list_t ** `list_a`, lw6sys_list_t ** `list_b`*`)`` [Function]

list_a: pointer to a list of string item

list_b: pointer to a list of string item

A typicall sort callback function, can be passed to `lw6sys_sort` to sort a list of 0-terminated strings.

Return value: -1 if `list_a < list_b` , 0 if `list_a == list_b`, 1 if `list_a > list_b`

`int lw6sys_sort_str_desc_callback (`*lw6sys_list_t ** `list_a`, lw6sys_list_t ** `list_b`*`)`` [Function]

list_a: pointer to a list of string item

list_b: pointer to a list of string item

A typicall sort callback function, can be passed to `lw6sys_sort` to sort a list of 0-terminated strings. This one will sort in reverse mode.

Return value: 1 if `list_a < list_b` , 0 if `list_a == list_b`, -1 if `list_a > list_b`

`void lw6sys_sort (`*lw6sys_list_t ** `list`, lw6sys_sort_callback_func_t `sort_func*`)` [Function]

list: the list to sort, might be modified by the function

sort_func: the callback function used to sort

A general sorting function. Internally, will use the glibc `qsort` function, but this one is adapted to the LW6 specific data structures, more exactly, the `lw6sys_list` structure. Several default sort callbacks are defined, but one is free to use any callback, provided it has the right prototype.

`char * lw6sys_str_copy (`*char * `src`*`)` [Function]

src: the string to copy

Duplicate a string, creating a new pointer on it, which must be freed afterwards. The main difference with `strdup` is that here we use the LW6SYS_MALLOC macro to track down possible memory leaks.

Return value: a newly allocated pointer, must be freed.

`char * lw6sys_str_concat (`*char * `str1`, char * `str2`*`)` [Function]

str1: the left part to be concatenated

str2: the right part to be concatenated

Concatenate 2 strings, and put the result in a newly allocated string. Unlike `strcat` which uses the same pointer.

Return value: a newly allocated pointer, must be freed.

`char * lw6sys_new_sprintf (`*char * `fmt`, `...`*`)` [Function]

fmt: a format string, like the one you would pass to `printf` ...: optional arguments, like the ones you would pass to `printf`

An sprintf like function, except it allocates a new string automatically, with "enough space". This is not a highly optimized function, it will allocate plenty of memory, possibly several times, and thus consume time and resources. But it has the great advantage of freeing the programmer of the dirty work of guessing "how log will the sprintf'ed string be?" before even generating it. So it's a time saver for the programmer. Additionnally, helps avoiding memory leaks and buffer overflows.

Return value: a new allocated string, must be freed.

`int lw6sys_str_is_blank (`*char * `str`*`)` [Function]

str: the string to test

Tests wether a string is blank, that is, if it's composed of space, tabs, or carriage returns only.

Return value: 1 if blank, 0 if not.

`int lw6sys_skip_blanks (`*char ** **`str_ptr`*`)` [Function]

str_ptr: a pointer to a string pointer (read/write parameter).

Skips blanks at the beginning of a string. The passed parameter is modifed in place. Usefull for parsing.

Return value: 1 if blanks were found, else 0.

`void lw6sys_str_cleanup (`*char * **`str`*`)` [Function]

str: a pointer to the string, which will be modified in-place.

Used to clean up some strings, for instance if they come from the network, we don't necessarly want system chars to be displayed on the console. Basically it removes all characters with an ASCII code inferior to 32, that is, all system characters. This way, there won't be any tab, linefeed, or any of such characters left.

Return value: none.

`char * lw6sys_str_reformat (`*char * **`str`**, *char * **`prefix`**, *int* **`nb_columns`**`)` [Function]

str: a pointer to the string we want to modify

prefix: a prefix to put before each line

Reformats a string, that is, insert newline characters in the right places to that it fits in a given number of columns. A prefix is appended at the beginning of each line. Will not handle strings which already contain newline characters perfectly.

Return value: a newly allocated string, must be freed.

`char * lw6sys_eol ()` [Function]

Returns the value of EOL, that is, the "end of line" sequence. Will simply return "\n" on UNIX and "\r\n" on Microsoft platforms. Note that while this is convenient to write config and example files, for instance, it's a bad idea to use this to generate network messages, because this kind of message needs to be platform independant. Thus any network protocol oriented string would use chr(10) and char(13) directly.

Return value: the EOL string, must not be freed.

`lw6sys_list_t * lw6sys_str_split (`*char * **`str`**, *char* **`c`**`)` [Function]

str: a string

c: the delimiter to split with

Splits a string, for instance 'foo,bar' splited with 'o' will return 'f', '' and ',bar'.

Return value: a list containing 0-terminated strings.

`lw6sys_list_t * lw6sys_str_split_no_0 (`*char * **`str`**, *char* **`c`**`)` [Function]

str: a string

c: the delimiter to split with

Splits a string, ignoring empty '0-length' members. For instance 'foo,bar' splited with 'o' will return 'f' and ',bar'.

Return value: a list containing 0-terminated strings.

`int lw6sys_test ()` [Function]

Runs the `sys` module test suite, testing most (if not all...) functions. Note that some tests perform file system operations and might therefore fail on a read-only filesystem, or if user permissions are not sufficient.

Return value: 1 if test is successfull, 0 on error.

`void * lw6sys_thread_create (`*lw6sys_thread_callback_func_t* `callback_func`*, lw6sys_thread_callback_func_t* `callback_join`*, void ** `callback_data`*, int* `flag``)` [Function]

callback_func: the main callback, the function that will run the thread

callback_join: function which will be called when joining, at the end

callback_data: data which will be passed to the callback

flag: a user flag which will be associated to the thread

Creates a thread. All threads must be joined. This is because we really do not want the game to leak, and detached threads are typically the kind of thing that leaves stuff in the heap. Note that callback_func is just something which will be called when joining it can be NULL. The idea is to put in it free & delete functions, which you can't call before joining when you want the main thread to get the results of the callback_func.

Return value: an opaque pointer on the thread. Can be NULL if failed.

`int lw6sys_thread_is_callback_done (`*void ** `thread_handler``)` [Function]

thread_handler: thread to work on

Tells wether the callback is done, that is to say, wether the results are available, and we can join.

Return value: 1 if done, else 0.

`int lw6sys_thread_get_id (`*void ** `thread_handler``)` [Function]

thread_handler: thread to query

Returns the id of the thread, this is an internal value, unique for each process, which can help identifying the thread.

Return value: the id, should be >0.

`void * lw6sys_thread_get_data (`*void ** `thread_handler``)` [Function]

thread_handler: thread to query

Returns the data associated to the thread, that is, the pointer which was passed to the callback function.

Return value: a pointer.

`int lw6sys_thread_get_flag (`*void ** `thread_handler``)` [Function]

thread_handler: thread to query

Returns the flag associated to the thread, that is, the integer which was given when creating the thread. This can be used in any way you want to decide what to do when a thread is over, for instance.

Return value: a pointer.

`void lw6sys_thread_join (`*`void * thread_handler`*`)` [Function]
thread_handler: thread to end

Joins the thread, that's to say wait until the thread is over, and destroys the ressources associated to it. Note that if the thread is looping forever, this function will just wait forever. This is the only way to end a thread.

Return value: none.

`int lw6sys_get_thread_create_count ()` [Function]
Utility function used to check how many threads where created and joined.

Return value: how many threads were created.

`int lw6sys_get_thread_join_count ()` [Function]
Utility function used to check how many threads where created and joined.

Return value: how many threads were joined.

`int lw6sys_check_thread_count ()` [Function]
Utility function used to check how many threads where created and joined. This one will compare the results of `lw6sys_get_thread_create_count` and `lw6sys_get_thread_join_count`.

Return value: 1 if both are equals, 0 if not (error...).

`int64_t lw6sys_timestamp ()` [Function]
Returns a 64-bit timestamp, for general purpose, but no precise timing. Precision is only of 1 second, see the `gfx` module for more accurate timing, for animations for instance.

Return value: the timestamp.

`int32_t lw6sys_uptime ()` [Function]
Returns the uptime, in seconds, since program startup. Based on timestamp.

Return value: the uptime, in seconds.

`void lw6sys_sleep (`*`float seconds`*`)` [Function]
seconds: the number of seconds to wait, fractions allowed

Will sleep for the given amount of seconds. Provides accurate timing and has "about-millisecond" precision, since it uses `select` internally. Might however be interrupted in some cases, so consider function can always return quicker than specified. A common usage of this function is polling loops, where you don't care if 2 polls are very close, but simply want to avoid polling continuously, therefore consumming 100% of the CPU for nothing.

`void lw6sys_time_init ()` [Function]
Global initializations required to handle time properly.

4.15.19 libtsk

`lw6tsk_loader_t * lw6tsk_loader_new` (*float* ***sleep***, *float* * ***progress***) [Function]

sleep: how many seconds to wait between every poll

Creates a new loader. This object is used to do some reputed slow calculus in the background, in a separated thread. Typical example is map loading. This is a high-level objects which encapsulates threads and other wizardry.

Return value: a pointer to the loader, NULL if failed.

`void lw6tsk_loader_free` (*lw6tsk_loader_t* * ***loader***) [Function]

loader: the loader to free.

Deletes a loader. Will automatically stop the child thread, free data, and so on.

Return value: none.

`char * lw6tsk_loader_repr` (*lw6tsk_loader_t* * ***loader***) [Function]

loader: the loader to represent.

Creates a string which briefly describes the loader.

Return value: a dynamically allocated pointer, must be freed.

`int lw6tsk_loader_get_stage` (*lw6tsk_loader_t* * ***loader***) [Function]

loader: the loader to query.

Returns the current stage of the loader.

Return value: 0 if idle, 1 if loading the map from disk, 2 if build dynamic stuff such as game_state.

Appendix A 2005 .plan

Here's my .plan file, which describes what I (Christian Mauduit) have planned for Liquid War 6. There's no garantee that what's written here is a precise description of the real future, however it should give a good idea of what I have in mind.

Note that the information here was written in summer 2005, it might or not be accurate now, as the main reason for plans to exist is that people never follow them. I'm no exception.

A.1 Complete rewrite

Liquid War 6 will be an almost complete rewrite. I mean that common code between branches 5 and 6 might end up in representing 0% of the total code. I think this is a wise decision, for the current code is really hard to maintain, and would not survive any serious cleanup. LW5 was first written in 1998, for DOS, when I had much less experience in programming. In 7 years I - and other people as well - hacked major enhancements in it such as cross-platform support, network games, and if you compare release 5.0 with the latest 5.x.x release, you'll see that a bunch of things have changed. I had never expected I would patch and fix this game for so long, and it's no surprise that it's bloated today.

FYI, here's a list of what makes LW5 unsuitable for major improvements without a complete rewrite:

- global variable hell. Lots of things are stored in globals.
- hard-coded C GUI. Read src/level.c to get an idea of how horrible it is.
- hard-coded 256 colors paletted mode. A clever bet in 1998 (performance...). Not anymore.
- generally bloated code. Makes bug-finding very tricky.

A.2 Technologies

Liquid War 6 will use a different technical framework than Liquid War 5.

A.2.1 Script + standard C + assembly

It happens that coding a large project in pure C is a waist of time, if possible at all.

If one applies the standard 80/20 rule to a computer game, one might state that 80% of the code eat up 20% of the CPU and the other 20% of the code eat up 80% of the CPU, the former being high-level glue code and the latter being low-level algorithmic code.

With Liquid War, one could speak of the 99/01 rule. I mean that 99% of the CPU time concerns only 1% of the code, and vice-versa. Basically, Liquid War has a very CPU-greedy core algorithm, still spends a fair amount of CPU displaying stuff (but this is delegated to the low-level game programming library) and the rest is totally unsignificant, in terms of CPU. Point is this "rest" represents the vast majority of the code, and also represents the very same buggy code I spend nights to patch on Liquid War 5. I'm talking about network code, GUI, and other high-level glue-code which are currently being written in C.

This idea is to write all this in a convenient scripting language. There won't be any impact on performances. I can't garantee Liquid War 6 will be blazingly fast, but for sure it won't be the scripting language fault. And of course if, as in Liquid War 3 and 5, I feel the need to implement some stuff in assembly for performances issues, I will do it.

We end up with a multi-language architecture: script + C + assembly.

My guess is that I'll use Scheme as an extension language. Python would be a good choice too. Let's say I'll give Scheme a chance, and if it's really not adapted, I'll switch back to Python. The point is that today I know Python and don't really know Scheme, but, well, it's always a pleasure for me to learn new things. It's fun.

So what is planned today is that Liquid War 6 will be a Scheme program, which will call callbacks functions written in C and/or assembly. These functions will do all the low-level time consuming algorithmic and graphical stuff. The rest of the code being entirely scripted.

A.2.2 OpenGL

Liquid War is not a 3D game, so why use OpenGL?

- it's a very convenient way to access video hardware acceleration with XFree86.
- low-end computers and/or computers without 3D acceleration can still run Liquid War 5.
- I'm interested in learning/using this API 8-)

This choice implies that I won't use Allegro anymore. Allegro stays a very convenient library and I would recommend it for it's excellent, easy to learn, powerfull, and stable. But for the needs of Liquid War 6 I'll use something else (because of OpenGL). I first thought of using GLUT but I might end up simply using SDL. The idea is just fo have an OpenGL wrapper which sets up OpenGL in a similar manner on all platforms, and handles basic things such as mouse or keyboard.

A.2.3 CSound

I've got two excellent books on Csound, and the will to learn how to use this tool.

I'll probably use Csound for a number of things, ranging from "bubbling sounds" to full blown music. Stay tuned 8-)

A.3 Functionnalities

A.3.1 Visual enhancements

Of course Liquid War 6 will look nicer than Liquid War 5, blah blah blah. What do you think?

Maybe I'll try to use some OpenGL features to make it possible to play on a ball, on a Moebius ring, or other fancy things. I have zillion of ideas, future will decide which ones will be implemented first.

To make it clear, visual enhancements aren't my top-level priority. However I'll try and make room for these enhancements, and prepare the terrain correctly. So it's possible that the first releases of Liquid War 6 won't be that much better than Liquid War 5, but at least Liquid War 6 will have the possibility to evolve. Something Liquid War 5 doesn't have.

A.3.2 Rules enhancements

There are many things that could be done easily:

- several cursors for one team

- alliances between teams
- deep places on a map, where more liquid can reside
- circular maps which "connect" the left border to the right one
- ...

As for graphical improvements, this is not my top-level priority. Simply, I'll make the game ready-to-improve. Again, all these enhancements are very hard to code in Liquid War 5, else I would already have coded them. Network enhancements

That's my top-level prioriry.

Why is that? Well, think of Liquid War in terms of "what makes it a good game?" and "what makes it a poor game?".

It's a good game because:

- the idea is original
- the gameplay is addictive
- you can play on a LAN
- all the family can play
- it's cross-platform
- it's Free Software

It's a poor game because:

- it's somewhat ugly and has a retro "back in the eighties" look
- network games are slow on Internet
- there are not enough active Internet servers

For the ugliness, well, OpenGL and some artwork should make it. But for the network, what's the real problem?

The real problem is that in the current situation, the server needs to have all "keystrokes" before doing anything, and all players must be connected before a game starts. Here's what I plan to do to fix this:

- players will be able to connect on a game "on the fly". This is done by most online games, and it's IMHO a required features for a network mode to work on Internet (not speaking of local networks, but real wide online gaming). How this will fit with Liquid War's rules is not totally decided, but I already know of several way to achieve this.
- I'll implement an "anticipation" system "a la" U61. This means that no matter if a remote player has a poor network connection, things will behave as if everything was fine. Internally, the system keeps 2 images of the game. One which is "anticipated" and displayed to the player, and one which is validated but outdated, kept internally. It's a little hard to explain, consumes twice as much CPU and memory, but it works. It happens that today the lacking ressource for playing Liquid War online is more on the network side than on the local CPU and memory aspects.
- I'll take it to the next level and implement a "peer-to-peer-like" network model, in which any client can become a server. The idea behind is that if a server quits the game, then a client takes its role, letting the game continue for hours. This way one could virtually have a never ending Liquid War game which would last weeks. I believe

this could be really cool. I also believe no proprietary game will ever implement that, for in this model there's no way to force people to access a centralized server, this server usually being the major key in the business model of a company which sells proprietary software.

This third point will be the real enhancement of Liquid War with version 6. It's one of the very points which drives me to rewrite it completely. First because it's impossible to implement it without some heavy work. Then because I find it very motivating.

A.3.3 Hey, you forgot my idea!!!

Many gamers submitted suggestions, either by mail or by posting messages on the mailing list.

Don't worry, I keep them. Not reading them here does not mean I won't implement them. It simply means I won't implement them first. I first need the game basically function before enhancing it with fancy stuff.

A.4 Road map

As I stated on the mailing list, when thinking about Liquid War 6, think of years rather than months (unless I get fired, jobless, or spend several months in a hospital with a laptop).

Note that this road map takes it for granted that I'll be the lone coder on the project. It's unlikely that someone is going to help me for the first stages, until there's at least something real, something playable. Something that proves that the concept is valid. Besides, (real) team work implies a significant overhead, especially at project start. It's hard to figure out how to distribute tasks when the tasks themselves are not clearly identified. But for the rest (starting in 2007 or 2008), it's possible that external help might greatly... ...help!

- 2005 : Project framework should be done. This implies that the scripting engine is up and running, graphical mode works, config and data loading work, basic menus are available. Nothing playable.
- 2006 : Import the core algorithm from Liquid War 5, make the game playable in "demo mode" (" la" Liquid War 2), implement the network "peer-to-peer-like" mode. At this stage, it will be possible to know wether Liquid War 6 is true vaporware or not.
- 2007 : glue all this together to make something usable by anyone, heavy work on the GUI, on the options, on error checking, many bug fixes. The goal is to have a game which is equivalent to Liquid War 5, with the network aspects pushed to the next level.
- 2008 : tadaaaaaaaaaaa! Release the game "publicly" - inform Freshmeat 8-) - and enhance it with all the feedback from gamers (bug reports and suggestions received since 1998). Work on artwork (both graphics and musics). Write documentation.
- 2009 : stabilize the game, patch it for all those things which had been forgotten back then in 2005, optimize for speed, bug-fix bug-fix bug-fix.
- 2010 : stop maintaining Liquid War 5, invite Liquid War fans and coders to a hudge party in my garden, sing all night, drink beers and wine, teach Liquid War strategies to my 5 and 6 year old daughters, remember the old times when Liquid War wasn't so cool 8-)

Appendix B Fanfic

Quoting Gavin: “I wrote a liquid war fanfic some time ago [...] I wrote it after a friend claimed that there wasn’t any liquid war fanfic because it wasn’t possible.”

So here it is, a Liquid War fanfic. It was initially written for Liquid War 5, but applies to Liquid War 6 as well. Enjoy!

B.1 The Battle of Emberlificoted

...

The General presided over his massing army in his seat, or rather hovering ring, of power. It dipped slightly as he flew low over his troops marching through the viscous marsh-like terrain. They were like children: obedient, loyal, and they ate a lot.

Glancing at the status panel mounted in front of him he grimaced; the other five armies: Yellow, Green, Orange, Turquoise, and, of course, Red, were also readying armies of a similar size to his own. His violet clones would have to fight hard and eat well to win this day.

Today would not be a battle of luck, the General mused, it would be a battle of tactics, of alliances, and of betrayal. Every clone was identical - that was the general idea behind clones - and the terrain seemed strangely symmetrical; it would not give advantage to any of the six armies amassed today. Glancing at the hologram of the battlefield projected in front of him the General noted that he would have to move quickly, Orange and Yellow were too close for comfort, though fortunately Baron Red’s army of eponymous coloured clones was the furthest.

General Violet’s fingertips were sweaty even before they touched the four main control keys in front of him. They were labeled ’W’, ’A’, ’D’, and, of course, the full retreat button - very useful for misleading foes and ambushing them as they pursued - ’S’. The keys were arrange in a roughly equilateral triangular pattern; with ’S’ forming the base and being adjacent to both ’A’ and ’D’, ’W’ formed the tip of the triangle.

A long breath left his parched lips as at last he made his move.

...

“Dammit!” he screamed moments later. He had misjudged Captain Yellow and Commander Orange; he had expected one at least to attack immediately, one he could have handled. They were working together - foiling his attempt to shoot between them to near the center of the battlefield to gain a better vantage point. Yellow had shot down towards him, cutting off his advance, and now Orange had sealed his escape route. “It’s not over yet” muttered the General. He opened a voice channel with Commander Orange:

“Very clever. Flawed, but still clever.”

“Flawed?” came the reply.

“Yes flawed, when the good Captain is finished devouring my army who do you think he will turn to next?”, bluffed the General - his hands worked quickly as he manoeuvred

his hovering control ring, all that his troops ever saw of him, carefully towards the weakest section of his attackers. If he could just break out a few units he could soon turn the tide against both Yellow and Orange.

"We have an alliance..." Orange's voice was unsure now.

Time for some sarcasm to through her even more off balance, thought the General,

"I gathered", he spoke softly, slowly, and with too much meaning. Then closing the channel he turned his attention back to his escape.

...

"Yes!" wooped the ecstatic figure of the General. Fifty or so of his troops had broken free undetected and were even now working their way cautiously towards the camps of the Yellow army, only the front lines were still actively fighting; this opening gambit of Yellow and Orange had turned into a stale siege and Yellow's army had pitched tent.

General Violet steered his hovering guidance ring to the center of the Yellow camp. His troops struck, both those who had got behind the lines and those who were still besieged. Yellow reacted too slowly and suddenly found that her army, was shrinking back from the onslaught. There was nowhere to run to, and bye now her only ally - Commander Orange - had abandoned her to her fate; he was too busy engaging Sir. Turquoise, who had managed to escape from the slaughter that the Baron had caused to the Turquoise ranks and was even now valiantly attacking the flanks of the Orange troops.

A glance at the status panel showed that Yellow's life force was fading quickly: 8%, 3%, 1%, Gone.

The General smiled, he always enjoyed getting the first kill, and by now his armies life force had grown and his clones had replicated. With his, now, formidable fighting force it was no problem to engulf both Sir. Turquoise and Commander Orange's brawling armies and annihilate them. Once again his army grew in size and power. Now if only the Baron didn't notice that..., thought the General.

...

"Too late!" yelped the General, now thrown into panic, as he saw the approaching Baron. His army had also grown in size and power - having fatally injured the Turquoise army within the opening moments of the battle, and having finally managed to catch the elusive fleeing form of, or what remained of, Emperor Green.

Gripping the controls harder the General thought quickly, his army doesn't so completely outnumber me that this is already over, however unless I can cause him to make a mistake that allows me to take the upper hand then I will inevitably lose. Maybe I can...

This thought was terminated and replaced by another as the Baron's angry red troops broke through the undergrowth that had covered their movements and started to surround the General's army. The thought that now throbbed through the panic-stricken mind of General Violet was simply 'Run!'.

Even as he signaled the retreat and made for what seemed to be the only possible means of escape the Baron's blood red control ring appeared at the opening. The General knew it was over, even before the host of red beings appeared at the opening.

There was no escape. His life force was almost depleted and he was surrounded. Then it was that the Baron decided to communicate:

"Too bad. It was a good game"

The General blinked, gaped, and was generally gobsmacked. Just before his life force completely failed and his own weary eyes closed in defeat he snarled,

"What!? This is not a game!" were the General's dying words.

Appendix C Links

This section lists various Internet Liquid War related links.

- Liquid War 6 homepage
- Liquid War 6 on ufoot.org
- Online manual
- Savannah downloads
- ufoot.org downloads
- GNU Arch repositoty
- Project on Savannah
- Mailing-list archives
- Liquid War entry on Wikipedia
- Liquid War entry on Wikipedia (French)
- Liquid War 6 entry on LGDB
- Liquid War 6 entry on Libregamewiki
- Liquid War 5

Appendix D GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. 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 them 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 prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. 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.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

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

TERMS AND CONDITIONS

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey 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; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a. The work must carry prominent notices stating that you modified it, and giving a relevant date.

   b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

   You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source.

The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

   a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
   b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
   c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d. Limiting the use for publicity purposes of names of licensors or authors of the material; or

e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

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

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

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance.

However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

    A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

    Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

    In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

    If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so

available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If 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 convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU 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 that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    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.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS 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.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

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 state 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) year 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 3 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, see http://www.gnu.org/licenses/.
```

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

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

```
program Copyright (C) year name of author
This program 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, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/.

The GNU 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. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html.

Appendix E GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released

under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

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

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

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

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

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

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

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and

that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

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

3. COPYING IN QUANTITY

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

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

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

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

4. MODIFICATIONS

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

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called

an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

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

8. TRANSLATION

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

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

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document 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.

10. FUTURE REVISIONS OF THIS LICENSE

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

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

E.1 ADDENDUM: How to use this License for your documents

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

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

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

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

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

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

Appendix F Indexes

F.1 Concept index

F.2 Function and keyword index

A

B

C

M

N

P

R

S

T

U

V

W

X

Y

Z