the GNU GRUB developer manual

The GRand Unified Bootloader, version 2.06, 10 May 2021.

Yoshinori K. Okuji
Colin D Bennett
Vesa Jskelinen
Colin Watson
Robert Millan
Carles Pina

This developer manual is for GNU GRUB (version 2.06, 10 May 2021).

Table of Contents

1 Getting the source code

GRUB is maintained using the GIT revision control system. To fetch:

```
git clone git://git.sv.gnu.org/grub.git
```

Web access is available under

```
http://git.savannah.gnu.org/cgit/grub.git/
```

The branches available are:

'master' Main development branch.

'grub-legacy'
: GRUB 0.97 codebase. Kept for reference and legal reasons

'multiboot'
: Multiboot specfication

'multiboot2'
: Multiboot2 specfication

'developer branches'
: Prefixed with developer name. Every developer of a team manages his own branches. Developer branches do not need changelog entries.

Once you have used *git clone* to fetch an initial copy of a branch, you can use *git pull* to keep it up to date. If you have modified your local version, you may need to resolve conflicts when pulling.

2 Coding style

Basically we follow the GNU Coding Standards. We define additional conventions for GRUB here.

2.1 Naming Conventions

All global symbols (i.e. functions, variables, types, and macros) must have the prefix grub_ or GRUB_. The all capital form is used only by macros.

2.2 Functions

If a function is global, its name must be prefixed with grub_ and must consist of only small letters. If the function belongs to a specific function module, the name must also be prefixed with the module name. For example, if a function is for file systems, its name is prefixed with grub_fs_. If a function is for FAT file system but not for all file systems, its name is prefixed with grub_fs_fat_. The hierarchy is noted this way.

After a prefix, a function name must start with a verb (such as get or is). It must not be a noun. Some kind of abbreviation is permitted, as long as it wouldn't make code less readable (e.g. init).

If a function is local, its name may not start with any prefix. It must start with a verb.

2.3 Variables

The rule is mostly the same as functions, as noted above. If a variable is global, its name must be prefixed with grub_ and must consist of only small letters. If the variable belongs to a specific function module, the name must also be prefixed with the module name. For example, if a function is for dynamic loading, its name is prefixed with grub_dl_. If a variable is for ELF but not for all dynamic loading systems, its name is prefixed with grub_dl_elf_.

After a prefix, a variable name must start with a noun or an adjective (such as name or long) and it should end with a noun. Some kind of abbreviation is permitted, as long as it wouldn't make code less readable (e.g. i18n).

If a variable is global in the scope of a single file (i.e. it is declared with static), its name may not start with any prefix. It must start with a noun or an adjective.

If a variable is local, you may choose any shorter name, as long as it wouldn't make code less readable (e.g. i).

2.4 Types

The name of a type must be prefixed with grub_ and must consist of only small letters. If the type belongs to a specific function module, the name must also be prefixed with the module name. For example, if a type is for OS loaders, its name is prefixed with grub_loader_. If a type is for Multiboot but not for all OS loaders, its name is prefixed with grub_loader_linux_.

The name must be suffixed with _t, to emphasize the fact that it is a type but not a variable or a function.

2.5 Macros

If a macro is global, its name must be prefixed with GRUB_ and must consist of only large letters. Other rules are the same as functions or variables, depending on whether a macro is used like a function or a variable.

2.6 Comments

All comments shall be C-style comments, of the form '/* ... */'. A comment can be placed immediately preceding the entity it describes or it can be placed together with code, variable declarations, or other non-comment entities. However, it is recommended to not mix various forms especially in types/structs descriptions.

Acceptable:

```
/* The page # that is the front buffer. */
int displayed_page;
```

```
int render_page; /* The page # that is the back buffer. */
```

2.7 Multi-Line Comments

Comments spanning multiple lines shall be formatted with all lines after the first aligned with the first line. Asterisk characters should be repeated at the start of each subsequent line.

Acceptable:

```
/*
 * This is a comment
 * which spans multiple lines.
 * It is long.
 */
```

Unacceptable:

```
/* This is a comment
   which spans multiple lines.
   It is long. */
```

```
/*
 * This is a comment
 * which spans multiple lines.
 * It is long. */
```

```
/* This is a comment
 * which spans multiple lines.
 * It is long.
 */
```

In particular first unacceptable form makes comment difficult to distinguish from the code itself. Especially if it contains the code snippets and/or is long. So, its usage is disallowed.

3 Finding your way around

Here is a brief map of the GRUB code base.

GRUB uses Autoconf and Automake, with most of the Automake input generated by a Python script. The top-level build rules are in '`configure.ac`', '`grub-core/Makefile.core.def`', and '`Makefile.util.def`'. Each block in a '`*.def`' file represents a build target, and specifies the source files used to build it on various platforms. The '`*.def`' files are processed into Automake input by '`gentpl.py`' (which you only need to look at if you are extending the build system). If you are adding a new module which follows an existing pattern, such as a new command or a new filesystem implementation, it is usually easiest to grep '`grub-core/Makefile.core.def`' and '`Makefile.util.def`' for an existing example of that pattern to find out where it should be added.

In general, code that may be run at boot time is in a subdirectory of '`grub-core`', while code that is only run from within a full operating system is in a subdirectory of the top level.

Low-level boot code, such as the MBR implementation on PC BIOS systems, is in the '`grub-core/boot/`' directory.

The GRUB kernel is in '`grub-core/kern/`'. This contains core facilities such as the device, disk, and file frameworks, environment variable handling, list processing, and so on. The kernel should contain enough to get up to a rescue prompt. Header files for kernel facilities, among others, are in '`include/`'.

Terminal implementations are in '`grub-core/term/`'.

Disk access code is spread across '`grub-core/disk/`' (for accessing the disk devices themselves), '`grub-core/partmap/`' (for interpreting partition table data), and '`grub-core/fs/`' (for accessing filesystems). Note that, with the odd specialised exception, GRUB only contains code to *read* from filesystems and tries to avoid containing any code to *write* to filesystems; this lets us confidently assure users that GRUB cannot be responsible for filesystem corruption.

PCI and USB bus handling is in '`grub-core/bus/`'.

Video handling code is in '`grub-core/video/`'. The graphical menu system uses this heavily, but is in a separate directory, '`grub-core/gfxmenu/`'.

Most commands are implemented by files in '`grub-core/commands/`', with the following exceptions:

- A few core commands live in '`grub-core/kern/corecmd.c`'.
- Commands related to normal mode live under '`grub-core/normal/`'.
- Commands that load and boot kernels live under '`grub-core/loader/`'.
- The '`loopback`' command is really a disk device, and so lives in '`grub-core/disk/loopback.c`'.
- The '`gettext`' command lives under '`grub-core/gettext/`'.
- The '`loadfont`' and '`lsfonts`' commands live under '`grub-core/font/`'.
- The '`serial`', '`terminfo`', and '`background_image`' commands live under '`grub-core/term/`'.
- The '`efiemu_*`' commands live under '`grub-core/efiemu/`'.

- OS-dependent code should be under '`grub-core/osdep/`'
- Utility programs meant to be run from a full operating system (except OS-dependent code mentioned previously) are in '`util/`'.

There are a few other special-purpose exceptions; grep for them if they matter to you.

4 Contributing changes

Contributing changes to GRUB 2 is welcomed activity. However we have a bit of control what kind of changes will be accepted to GRUB 2. Therefore it is important to discuss your changes on grub-devel mailing list (see MailingLists). On this page there are some basic details on the development process and activities.

First of all you should come up with the idea yourself what you want to contribute. If you do not have that beforehand you are advised to study this manual and try GRUB 2 out to see what you think is missing from there.

Here are additional pointers:

- `https://savannah.gnu.org/task/?group=grub GRUB’s Task Tracker`
- `https://savannah.gnu.org/bugs/?group=grub GRUB’s Bug Tracker`

If you intended to make changes to GRUB Legacy (<=0.97) those are not accepted anymore.

4.1 Getting started

- Always use latest GRUB 2 source code. So get that first.

  For developers it is recommended always to use the newest development version of GRUB 2. If development takes a long period of time, please remember to keep in sync with newest developments regularly so it is much easier to integrate your change in the future. GRUB 2 is being developed in a GIT repository.

  Please check Savannah’s GRUB project page for details how to get newest git: GRUB 2 git Repository
- Compile it and try it out.

  It is always good idea to first see that things work somehow and after that to start to implement new features or develop fixes to bugs.
- Study the code.

  There are sometimes odd ways to do things in GRUB 2 code base. This is mainly related to limited environment where GRUB 2 is being executed. You usually do not need to understand it all so it is better to only try to look at places that relates to your work. Please do not hesitate to ask for help if there is something that you do not understand.
- Develop a new feature.

  Now that you know what to do and how it should work in GRUB 2 code base, please be free to develop it. If you have not so far announced your idea on grub-devel mailing list, please do it now. This is to make sure you are not wasting your time working on the solution that will not be integrated to GRUB 2 code base.

  You might want to study our coding style before starting development so you do not need to change much of the code when your patch is being reviewed. (see Chapter 2 [Coding style], page 3)

  For every accepted patch there has to exist a ChangeLog entry. Our ChangeLog consist of changes within source code and are not describing about what the change logically does. Please see examples from previous entries.

Also remember that GRUB 2 is licensed under GPLv3 license and that usually means that you are not allowed to copy pieces of code from other projects. Even if the source project's license would be compatible with GPLv3, please discuss it beforehand on grub-devel mailing list.

- Test your change.

  Test that your change works properly. Try it out a couple of times, preferably on different systems, and try to find problems with it.

- Publish your change.

  When you are happy with your change, first make sure it is compilable with latest development version of GRUB 2. After that please send a patch to grub-devel for review. Please describe in your email why you made the change, what it changes and so on. Please be prepared to receive even discouraging comments about your patch. There is usually at least something that needs to be improved in every patch.

  Please use unified diff to make your patch (good match of arguments for diff is '`-pruN`').

- Respond to received feedback.

  If you are asked to modify your patch, please do that and resubmit it for review. If your change is large you are required to submit a copyright agreement to FSF. Please keep in mind that if you are asked to submit for copyright agreement, process can take some time and is mandatory in order to get your changes integrated.

  If you are not on grub-devel to respond to questions, most likely your patch will not be accepted. Also if problems arise from your changes later on, it would be preferable that you also fix the problem. So stay around for a while.

- Your patch is accepted.

  Good job! Your patch will now be integrated into GRUB 2 mainline, and if it didn't break anything it will be publicly available in the next release.

  Now you are welcome to do further improvements :)

4.2 Typical Developer Experience

The typical experience for a developer in this project is the following:

1. You find yourself wanting to do something (e.g. fixing a bug).
2. You show some result in the mailing list or the IRC.
3. You are getting to be known to other developers.
4. You accumulate significant amount of contribution, so copyright assignment is processed.
5. You are free to check in your changes on your own, legally speaking.

At this point, it is rather annoying that you ought to ask somebody else every change to be checked in. For efficiency, it is far better, if you can commit it yourself. Therefore, our policy is to give you the write permission to our official repository, once you have shown your skill and will, and the FSF clerks have dealt with your copyright assignment.

4.3 When you are approved for write access to project's files

As you might know, GRUB is hosted on https://savannah.gnu.org/projects/grub Savannah, thus the membership is managed by Savannah. This means that, if you want to be a member of this project:

1. You need to create your own account on Savannah.
2. You can submit "Request for Inclusion" from "My Groups" on Savannah.

Then, one of the admins can approve your request, and you will be a member. If you don't want to use the Savannah interface to submit a request, you can simply notify the admins by email or something else, alternatively. But you still need to create an account beforehand.

NOTE: we sometimes receive a "Request for Inclusion" from an unknown person. In this case, the request would be just discarded, since it is too dangerous to allow a stranger to be a member, which automatically gives him a commit right to the repository, both for a legal reason and for a technical reason.

If your intention is to just get started, please do not submit a inclusion request. Instead, please subscribe to the mailing list, and communicate first (e.g. sending a patch, asking a question, commenting on another message...).

5 Updating external code

GRUB includes some code from other projects, and it is sometimes necessary to update it.

5.1 Gnulib

Gnulib is a source code library that provides basic functionality to programs and libraries. Many software packages make use of Gnulib to avoid reinventing the portability wheel.

GRUB imports Gnulib using its `bootstrap` utility, identifying a particular Git commit in '`bootstrap.conf`'. To upgrade to a new Gnulib commit, set `GNULIB_REVISION` in '`bootstrap.conf`' to the new commit ID, then run *`./bootstrap`* and whatever else you need to make sure it works. Check for changes to Gnulib's '`NEWS`' file between the old and new commits; in some cases it will be necessary to adjust GRUB to match. You may also need to update the patches in '`grub-core/lib/gnulib-patches/`'.

To add a new Gnulib module or remove one that is no longer needed, change `gnulib_modules` in '`bootstrap.conf`'. Again, run *`./bootstrap`* and whatever else you need to make sure it works.

Bootstrapping from an older distribution containing gettext version < 0.18.3, will require a patch similar to this to be applied first before running the `./bootstrap` utility:

```
diff --git a/bootstrap.conf b/bootstrap.conf
index 988dda0..a3193a9 100644
--- a/bootstrap.conf
+++ b/bootstrap.conf
@ -67,7 +67,7 @ SKIP_PO=t
buildreq="\
autoconf   2.63
automake   1.11
-gettext    0.18.3
+gettext    0.17
git        1.5.5
tar        -
"
diff --git a/configure.ac b/configure.ac
index 08b518f..99f5b36 100644
--- a/configure.ac
+++ b/configure.ac
@ -362,7 +362,7 @ AC_CHECK_PROG(HAVE_CXX, $CXX, yes, no)

AC_GNU_SOURCE
AM_GNU_GETTEXT([external])
-AM_GNU_GETTEXT_VERSION([0.18.3])
+AM_GNU_GETTEXT_VERSION([0.17])
AC_SYS_LARGEFILE

# Identify characteristics of the host architecture.
```

It will also be necessary to adjust the patches in 'po/gettext-patches/' to apply to an older version of gettext.

5.2 jsmn

jsmn is a minimalistic JSON parser which is implemented in a single header file 'jsmn.h'. To import a different version of the jsmn parser, you may simply download the 'jsmn.h' header from the desired tag or commit to the target directory:

```
curl -L https://raw.githubusercontent.com/zserge/jsmn/v1.1.0/jsmn.h \
    -o grub-core/lib/json/jsmn.h
```

5.3 minilzo

miniLZO is a very lightweight subset of the LZO library intended for easy inclusion in other projects. It is generated automatically from the LZO source code and contains the most important LZO functions.

To upgrade to a new version of the miniLZO library, download the release tarball and copy the files into the target directory:

```
curl -L -O http://www.oberhumer.com/opensource/lzo/download/minilzo-2.08.tar.gz
tar -zxf minilzo-2.08.tar.gz
rm minilzo-2.08/testmini.c
rm -r grub-core/lib/minilzo/*
cp minilzo-2.08/*.[hc] grub-core/lib/minilzo
rm -r minilzo-2.08*
```

6 Porting

GRUB2 is designed to be easily portable accross platforms. But because of the nature of bootloader every new port must be done separately. Here is how I did MIPS (loongson and ARC) and Xen ports. Note than this is more of suggestions, not absolute truth.

First of all grab any architecture specifications you can find in public (please avoid NDA).

First stage is "Hello world". I've done it outside of GRUB for simplicity. Your task is to have a small program which is loadable as bootloader and clearly shows its presence to you. If you have easily accessible console you can just print a message. If you have a mapped framebuffer you know address of, you can draw a square. If you have a debug facility, just hanging without crashing might be enough. For the first stage you can choose to load the bootloader across the network since format for network image is often easier than for local boot and it skips the need of small intermediary stages and nvram handling. Additionally you can often have a good idea of the needed format by running "file" on any netbootable executable for given platform.

This program should probably have 2 parts: an assembler and C one. Assembler one handles BSS cleaning and other needed setup (on some platforms you may need to switch modes or copy the executable to its definitive position). So your code may look like (x86 assembly for illustration purposes)

```
        .globl _start
_start:
movl $_bss_start, %edi
movl $_end, %ecx
subl %edi, %ecx
xorl %eax, %eax
cld
rep
stosb
        call main
```

```
static const char msg[] = "Hello, world";

void
putchar (int c)
{
  ...
}

void
main (void)
{
  const char *ptr = msg;
  while (*ptr)
    putchar (*ptr++);
  while (1);
```

```
}
```

Sometimes you need a third file: assembly stubs for ABI-compatibility.

Once this file is functional it's time to move it into GRUB2. The startup assembly file goes to grub-core/kern/$cpu/$platform/startup.S. You should also include grub/symbol.h and replace call to entry point with call to EXT_C(grub_main). The C file goes to grub-core/kern/$cpu/$platform/init.c and its entry point is renamed to void grub_machine_init (void). Keep final infinite loop for now. Stubs file if any goes to grub-core/kern/$cpu/$platform/callwrap.S. Sometimes either $cpu or $platform is dropped if file is used on several cpus respectivelyplatforms. Check those locations if they already have what you're looking for.

Then modify in configure.ac the following parts:

CPU names:

```
case "$target_cpu" in
  i[[3456]]86) target_cpu=i386 ;;
  amd64) target_cpu=x86_64 ;;
  sparc) target_cpu=sparc64 ;;
  s390x) target_cpu=s390 ;;
  ...
esac
```

Sometimes CPU have additional architecture names which don't influence booting. You might want to have some canonical name to avoid having bunch of identical platforms with different names.

NOTE: it doesn't influence compile optimisations which depend solely on chosen compiler and compile options.

```
if test "x$with_platform" = x; then
  case "$target_cpu"-"$target_vendor" in
    i386-apple) platform=efi ;;
    i386-*) platform=pc ;;
    x86_64-apple) platform=efi ;;
    x86_64-*) platform=pc ;;
    powerpc-*) platform=ieee1275 ;;
    ...
  esac
else
  ...
fi
```

This part deals with guessing the platform from CPU and vendor. Sometimes you need to use 32-bit mode for booting even if OS runs in 64-bit one. If so add your platform to:

```
case "$target_cpu"-"$platform" in
  x86_64-efi) ;;
  x86_64-emu) ;;
  x86_64-*) target_cpu=i386 ;;
  powerpc64-ieee1275) target_cpu=powerpc ;;
esac
```

Add your platform to the list of supported ones:

```
case "$target_cpu"-"$platform" in
  i386-efi) ;;
  x86_64-efi) ;;
  i386-pc) ;;
  i386-multiboot) ;;
  i386-coreboot) ;;
  ...
esac
```

If explicit -m32 or -m64 is needed add it to:

```
case "$target_cpu" in
  i386 | powerpc) target_m32=1 ;;
  x86_64 | sparc64) target_m64=1 ;;
esac
```

Finally you need to add a conditional to the following block:

```
AM_CONDITIONAL([COND_mips_arc], [test x$target_cpu = xmips -a x$platform = xarc])
AM_CONDITIONAL([COND_sparc64_ieee1275], [test x$target_cpu = xsparc64 -a x$platform =
AM_CONDITIONAL([COND_powerpc_ieee1275], [test x$target_cpu = xpowerpc -a x$platform =
```

Next stop is gentpl.py. You need to add your platform to the list of supported ones (sorry that this list is duplicated):

```
GRUB_PLATFORMS = [ "emu", "i386_pc", "i386_efi", "i386_qemu", "i386_coreboot",
                   "i386_multiboot", "i386_ieee1275", "x86_64_efi",
                   "mips_loongson", "sparc64_ieee1275",
                   "powerpc_ieee1275", "mips_arc", "ia64_efi",
                   "mips_qemu_mips", "s390_mainframe" ]
```

You may also want already to add new platform to one or several of available groups. In particular we always have a group for each CPU even when only one platform for given CPU is available.

Then comes grub-core/Makefile.core.def. In the block "kernel" you'll need to define ldflags for your platform ($cpu_$platform_ldflags). You also need to declare startup asm file ($cpu_$platform_startup) as well as any other files (e.g. init.c and callwrap.S) (e.g. $cpu_$platform = kern/$cpu/$platform/init.c). At this stage you will also need to add dummy dl.c and cache.S with functions grub_err_t grub_arch_dl_check_header (void *ehdr), grub_err_t grub_arch_dl_relocate_symbols (grub_dl_t mod, void *ehdr) (dl.c) and void grub_arch_sync_caches (void *address, grub_size_t len) (cache.S). They won't be used for now.

You will need to create directory include/$cpu/$platform and a file include/$cpu/types.h. The later folowing this template:

```
#ifndef GRUB_TYPES_CPU_HEADER
#define GRUB_TYPES_CPU_HEADER 1

/* The size of void *.  */
#define GRUB_TARGET_SIZEOF_VOID_P 4
```

```
/* The size of long.  */
#define GRUB_TARGET_SIZEOF_LONG 4

/* mycpu is big-endian.  */
#define GRUB_TARGET_WORDS_BIGENDIAN 1
/* Alternatively: mycpu is little-endian.  */
#undef GRUB_TARGET_WORDS_BIGENDIAN

#endif /* ! GRUB_TYPES_CPU_HEADER */
```

You will also need to add a dummy file to datetime and setjmp modules to avoid any of it having no files. It can be just completely empty at this stage.

You'll need to make grub-mkimage.c (util/grub_mkimage.c) aware of the needed format. For most commonly used formats like ELF, PE, aout or raw the support is already present and you'll need to make it follow the existant code paths for your platform adding adjustments if necessary. When done compile:

```
./bootstrap
./configure --target=$cpu --with-platform=$platform TARGET_CC=.. OBJCOPY=... STRIP=...
make > /dev/null
```

And create image

```
./grub-mkimage -d grub-core -O $format_id -o test.img
```

And it's time to test your test.img.

If it works next stage is to have heap, console and timer.

To have the heap working you need to determine which regions are suitable for heap usage, allocate them from firmware and map (if applicable). Then call grub_mm_init_region (vois *start, grub_size_t s) for every of this region. As a shortcut for early port you can allocate right after _end or have a big static array for heap. If you do you'll probably need to come back to this later. As for output console you should distinguish between an array of text, terminfo or graphics-based console. Many of real-world examples don't fit perfectly into any of these categories but one of the models is easier to be used as base. In second and third case you should add your platform to terminfokernel respectively videoinkernel group. A good example of array of text is i386-pc (kern/i386/pc/init.c and term/i386/pc/console.c). Of terminfo is ieee1275 (kern/ieee1275/init.c and term/ieee1275/console.c). Of video is loongson (kern/mips/loongson/init.c). Note that terminfo has to be inited in 2 stages: one before (to get at least rudimentary console as early as possible) and another after the heap (to get full-featured console). For the input there are string of keys, terminfo and direct hardware. For string of keys look at i386-pc (same files), for terminfo ieee1275 (same files) and for hardware loongson (kern/mips/loongson/init.c and term/at_keyboard.c).

For the timer you'll need to call grub_install_get_time_ms (...) with as sole argument a function returning a grub_uint64_t of a number of milliseconds elapsed since arbitrary point in the past.

Once these steps accomplished you can remove the inifinite loop and you should be able to get to the minimal console. Next step is to have module loading working. For this you'll need to fill kern/$cpu/dl.c and kern/$cpu/cache.S with real handling of relocations and respectively the real sync of I and D caches. Also you'll need to decide where in the image to store the modules. Usual way is to have it concatenated at the end. In this case you'll need to modify startup.S to copy modules out of bss to let's say ALIGN_UP (_end, 8) before cleaning out bss. You'll probably find useful to add total_module_size field to startup.S. In init.c you need to set grub_modbase to the address where modules can be found. You may need grub_modules_get_end () to avoid declaring the space occupied by modules as usable for heap. You can test modules with:

```
./grub-mkimage -d grub-core -O $format_id -o test.img hello
```

and then running "hello" in the shell.

Once this works, you should think of implementing disk access. Look around disk/ for examples.

Then, very importantly, you probably need to implement the actual loader (examples available in loader/)

Last step to have minimally usable port is to add support to grub-install to put GRUB in a place where firmware or platform will pick it up.

Next steps are: filling datetime.c, setjmp.S, network (net/drivers), video (video/), halt (lib/), reboot (lib/).

Please add your platform to Platform limitations and Supported kernels chapter in user documentation and mention any steps you skipped which result in reduced features or performance. Here is the quick checklist of features. Some of them are less important than others and skipping them is completely ok, just needs to be mentioned in user documentation.

Checklist:

- Is heap big enough?
- Which charset is supported by console?
- Does platform have disk driver?
- Do you have network card support?
- Are you able to retrieve datetime (with date)?
- Are you able to set datetime (with date)?
- Is serial supported?
- Do you have direct disk support?
- Do you have direct keyboard support?
- Do you have USB support?
- Do you support loading through network?
- Do you support loading from disk?
- Do you support chainloading?
- Do you support network chainloading?
- Does cpuid command supports checking all CPU features that the user might want conditionalise on (64-bit mode, hypervisor,...)
- Do you support hints? How reliable are they?
- Does platform have ACPI? If so do "acpi" and "lsacpi" modules work?
- Do any of platform-specific operations mentioned in the relevant section of user manual makes sense on your platform?

- Does your platform support PCI? If so is there an appropriate driver for GRUB?
- Do you support badram?

7 Error Handling

Error handling in GRUB 2 is based on exception handling model. As C language doesn't directly support exceptions, exception handling behavior is emulated in software.

When exception is raised, function must return to calling function. If calling function does not provide handling of the exception it must return back to its calling function and so on, until exception is handled. If exception is not handled before prompt is displayed, error message will be shown to user.

Exception information is stored on `grub_errno` global variable. If `grub_errno` variable contains value `GRUB_ERR_NONE`, there is no active exception and application can continue normal processing. When `grub_errno` has other value, it is required that application code either handles this error or returns instantly to caller. If function is with return type `grub_err_t` is about to return `GRUB_ERR_NONE`, it should not set `grub_errno` to that value. Only set `grub_errno` in cases where there is error situation.

Simple exception forwarder.

```
grub_err_t
forwarding_example (void)
{
  /* Call function that might cause exception.  */
  foobar ();

  /* No special exception handler, just forward possible exceptions.  */
  if (grub_errno != GRUB_ERR_NONE)
    {
      return grub_errno;
    }

  /* All is OK, do more processing.  */

  /* Return OK signal, to caller.  */
  return GRUB_ERR_NONE;
}
```

Error reporting has two components, the actual error code (of type `grub_err_t`) and textual message that will be displayed to user. List of valid error codes is listed in header file '`include/grub/err.h`'. Textual error message can contain any textual data. At time of writing, error message can contain up to 256 characters (including terminating NUL). To ease error reporting there is a helper function `grub_error` that allows easier formatting of error messages and should be used instead of writing directly to global variables.

Example of error reporting.

```
grub_err_t
failing_example ()
{
  return grub_error (GRUB_ERR_FILE_NOT_FOUND,
                     "Failed to read %s, tried %d times.",
                     "test.txt",
```

```
                         10);
}
```

If there is a special reason that error code does not need to be taken account, `grub_errno` can be zeroed back to `GRUB_ERR_NONE`. In cases like this all previous error codes should have been handled correctly. This makes sure that there are no unhandled exceptions.

Example of zeroing `grub_errno`.

```
grub_err_t
probe_example ()
{
  /* Try to probe device type 1.  */
  probe_for_device ();
  if (grub_errno == GRUB_ERR_NONE)
    {
      /* Device type 1 was found on system.  */
      register_device ();
      return GRUB_ERR_NONE;
    }
  /* Zero out error code.  */
  grub_errno = GRUB_ERR_NONE;

  /* No device type 1 found, try to probe device type 2.  */
  probe_for_device2 ();
  if (grub_errno == GRUB_ERR_NONE)
    {
      /* Device type 2 was found on system.  */
      register_device2 ();
      return GRUB_ERR_NONE;
    }
  /* Zero out error code.  */
  grub_errno = GRUB_ERR_NONE;

  /* Return custom error message.  */
  return grub_error (GRUB_ERR_UNKNOWN_DEVICE, "No device type 1 or 2 found.");
}
```

Some times there is a need to continue processing even if there is a error state in application. In situations like this, there is a needed to save old error state and then call other functions that might fail. To aid in this, there is a error stack implemented. Error state can be pushed to error stack by calling function `grub_error_push ()`. When processing has been completed, `grub_error_pop ()` can be used to pop error state from stack. Error stack contains predefined amount of error stack items. Error stack is protected for overflow and marks these situations so overflow error does not get unseen. If there is no space available to store error message, it is simply discarded and overflow will be marked as happened. When overflow happens, it most likely will corrupt error stack consistency as for pushed error there is no matching pop, but overflow message will be shown to inform user about the situation. Overflow message will be shown at time when prompt is about to be drawn.

Example usage of error stack.

```
/* Save possible old error message.  */
grub_error_push ();

/* Do your stuff here.  */
call_possibly_failing_function ();

if (grub_errno != GRUB_ERR_NONE)
  {
    /* Inform rest of the code that there is error (grub_errno
       is set). There is no pop here as we want both error states
       to be displayed.  */
    return;
  }

/* Restore old error state by popping previous item from stack. */
grub_error_pop ();
```

8 Stack and heap size

On emu stack and heap are just normal host OS stack and heap. Stack is typically 8 MiB although it's OS-dependent.

On i386-pc, i386-coreboot, i386-qemu and i386-multiboot the stack is 60KiB. All available space between 1MiB and 4GiB marks is part of heap.

On *-xen stack is 4MiB. If compiled for x86-64 with GCC 4.4 or later adressable space is unlimited. When compiled for x86-64 with older GCC version adressable space is limited to 2GiB. When compiling for i386 adressable space is limited to 4GiB. All adressable pages except the ones for stack, GRUB binary, special pages and page table are in the heap.

On *-efi GRUB uses same stack as EFI. If compiled for x86-64 with GCC 4.4 or later adressable space is unlimited. When compiled for x86-64 with older GCC version adressable space is limited to 2GiB. For all other platforms adressable space is limited to 4GiB. GRUB allocates pages from EFI for its heap, at most 1.6 GiB.

On i386-ieee1275 and powerpc-ieee1275 GRUB uses same stack as IEEE1275. It allocates at most 32MiB for its heap.

On sparc64-ieee1275 stack is 256KiB and heap is 2MiB.

On mips(el)-qemu_mips and mipsel-loongson stack is 2MiB (everything below GRUB image) and everything above GRUB image (from 2MiB + kernel size) until 256MiB is part of heap.

On mips-arc stack is 2MiB (everything below GRUB image) and everything above GRUB image(from 2MiB + kernel size) until 128MiB is part of heap.

On mipsel-arc stack is 2MiB (everything below GRUB image which is not part of ARC) and everything above GRUB image (from 7MiB + kernel size) until 256MiB is part of heap.

On arm-uboot stack is 256KiB and heap is 2MiB.

In short:

Platform	**Stack**	**Heap**
emu	8 MiB	?
i386-pc	60 KiB	< 4 GiB
i386-coreboot	60 KiB	< 4 GiB
i386-multiboot	60 KiB	< 4 GiB
i386-qemu	60 KiB	< 4 GiB
*-efi	?	< 1.6 GiB
i386-ieee1275	?	< 32 MiB
powerpc-ieee1275	?	< 32 MiB
sparc64-ieee1275	256KiB	2 MiB
arm-uboot	256KiB	2 MiB
mips(el)-qemu_mips	2MiB	253 MiB

mipsel-loongson	2MiB	253 MiB
mips-arc	2MiB	125 MiB
mipsel-arc	2MiB	248 MiB
x86_64-xen (GCC >= 4.4)	4MiB	unlimited
x86_64-xen (GCC < 4.4)	4MiB	< 2GiB
i386-xen	4MiB	< 4GiB

9 BIOS port memory map

Start	End	Usage
0	0x1000 - 1	BIOS and real mode interrupts
0x07BE	0x07FF	Partition table passed to another boot loader
?	0x2000 - 1	Real mode stack
0x7C00	0x7D00 - 1	Boot sector
0x8000	?	GRUB kernel
0x68000	0x71000 - 1	Disk buffer
?	0x80000 - 1	Protected mode stack
?	0xA0000 - 1	Extended BIOS Data Area
0xA0000	0xC0000 - 1	Video RAM
0xC0000	0x100000 - 1	BIOS
0x100000	?	Heap and module code

10 Video Subsystem

This document contains specification for Video Subsystem for GRUB2. Currently only the usage interface is described in this document. Internal structure of how video drivers are registering and how video driver manager works are not included here.

10.1 Video API

10.1.1 grub_video_setup

- Prototype:

```
grub_err_t
grub_video_setup (unsigned int width, unsigned int height, unsigned int mode_type)
```

- Description:

  Driver will use information provided to it to select best possible video mode and switch to it. Supported values for `mode_type` are `GRUB_VIDEO_MODE_TYPE_INDEX_COLOR` for index color modes, `GRUB_VIDEO_MODE_TYPE_RGB` for direct RGB color modes and `GRUB_VIDEO_MODE_TYPE_DOUBLE_BUFFERED` for double buffering. When requesting RGB mode, highest bits per pixel mode will be selected. When requesting Index color mode, mode with highest number of colors will be selected. If all parameters are specified as zero, video adapter will try to figure out best possible mode and initialize it, platform specific differences are allowed here. If there is no mode matching request, error X will be returned. If there are no problems, function returns `GRUB_ERR_NONE`.

  This function also performs following task upon succesful mode switch. Active rendering target is changed to screen and viewport is maximized to allow whole screen to be used when performing graphics operations. In RGB modes, emulated palette gets 16 entries containing default values for VGA palette, other colors are defined as black. When switching to Indexed Color mode, driver may set default VGA palette to screen if the video card allows the operation.

10.1.2 grub_video_restore

- Prototype:

```
grub_err_t
grub_video_restore (void);
```

- Description:

  Video subsystem will deinitialize activated video driver to restore old state of video device. This can be used to switch back to text mode.

10.1.3 grub_video_get_info

- Prototype:

```
grub_err_t
grub_video_get_info (struct grub_video_mode_info *mode_info);
```

```
struct grub_video_mode_info
{
  /* Width of the screen.  */
```

```
    unsigned int width;
    /* Height of the screen.  */
    unsigned int height;
    /* Mode type bitmask.  Contains information like is it Index color or
       RGB mode.  */
    unsigned int mode_type;
    /* Bits per pixel.  */
    unsigned int bpp;
    /* Bytes per pixel.  */
    unsigned int bytes_per_pixel;
    /* Pitch of one scanline.  How many bytes there are for scanline.  */
    unsigned int pitch;
    /* In index color mode, number of colors.  In RGB mode this is 256.  */
    unsigned int number_of_colors;
    /* Optimization hint how binary data is coded.  */
    enum grub_video_blit_format blit_format;
    /* How many bits are reserved for red color.  */
    unsigned int red_mask_size;
    /* What is location of red color bits.  In Index Color mode, this is 0.  */
    unsigned int red_field_pos;
    /* How many bits are reserved for green color.  */
    unsigned int green_mask_size;
    /* What is location of green color bits.  In Index Color mode, this is 0.  */
    unsigned int green_field_pos;
    /* How many bits are reserved for blue color.  */
    unsigned int blue_mask_size;
    /* What is location of blue color bits.  In Index Color mode, this is 0.  */
    unsigned int blue_field_pos;
    /* How many bits are reserved in color.  */
    unsigned int reserved_mask_size;
    /* What is location of reserved color bits.  In Index Color mode,
       this is 0.  */
    unsigned int reserved_field_pos;
  };
```

- Description:

  Software developer can use this function to query properties of active rendering taget. Information provided here can be used by other parts of GRUB, like image loaders to convert loaded images to correct screen format to allow more optimized blitters to be used. If there there is no configured video driver with active screen, error `GRUB_ERR_BAD_DEVICE` is returned, otherwise `mode_info` is filled with valid information and `GRUB_ERR_NONE` is returned.

10.1.4 grub_video_get_blit_format

- Prototype:

```
  enum grub_video_blit_format
  grub_video_get_blit_format (struct grub_video_mode_info *mode_info);
```

```
enum grub_video_blit_format
  {
    /* Follow exactly field & mask information.  */
    GRUB_VIDEO_BLIT_FORMAT_RGBA,
    /* Make optimization assumption.  */
    GRUB_VIDEO_BLIT_FORMAT_R8G8B8A8,
    /* Follow exactly field & mask information.  */
    GRUB_VIDEO_BLIT_FORMAT_RGB,
    /* Make optimization assumption.  */
    GRUB_VIDEO_BLIT_FORMAT_R8G8B8,
    /* When needed, decode color or just use value as is.  */
    GRUB_VIDEO_BLIT_FORMAT_INDEXCOLOR
  };
```

- Description:

  Used to query how data could be optimized to suit specified video mode. Returns exact video format type, or a generic one if there is no definition for the type. For generic formats, use `grub_video_get_info` to query video color coding settings.

10.1.5 grub_video_set_palette

- Prototype:

```
grub_err_t
grub_video_set_palette (unsigned int start, unsigned int count, struct grub_video_p
```

```
struct grub_video_palette_data
{
    grub_uint8_t r; /* Red color value (0-255). */
    grub_uint8_t g; /* Green color value (0-255). */
    grub_uint8_t b; /* Blue color value (0-255). */
    grub_uint8_t a; /* Reserved bits value (0-255). */
};
```

- Description:

  Used to setup indexed color palettes. If mode is RGB mode, colors will be set to emulated palette data. In Indexed Color modes, palettes will be set to hardware. Color values will be converted to suit requirements of the video mode. `start` will tell what hardware color index (or emulated color index) will be set to according information in first indice of `palette_data`, after that both hardware color index and `palette_data` index will be incremented until `count` number of colors have been set.

10.1.6 grub_video_get_palette

- Prototype:

```
grub_err_t
grub_video_get_palette (unsigned int start, unsigned int count, struct grub_video_p
```

```
struct grub_video_palette_data
{
    grub_uint8_t r; /* Red color value (0-255). */
    grub_uint8_t g; /* Green color value (0-255). */
```

```
    grub_uint8_t b; /* Blue color value (0-255). */
    grub_uint8_t a; /* Reserved bits value (0-255). */
};
```

- Description:

  Used to query indexed color palettes. If mode is RGB mode, colors will be copied from emulated palette data. In Indexed Color modes, palettes will be read from hardware. Color values will be converted to suit structure format. `start` will tell what hardware color index (or emulated color index) will be used as a source for first indice of `palette_data`, after that both hardware color index and `palette_data` index will be incremented until `count` number of colors have been read.

10.1.7 grub_video_set_area_status

- Prototype:

```
grub_err_t
grub_video_set_area_status (grub_video_area_status_t area_status);
```

```
enum grub_video_area_status_t
  {
    GRUB_VIDEO_AREA_DISABLED,
    GRUB_VIDEO_AREA_ENABLED
  };
```

- Description:

  Used to set area drawing mode for redrawing the specified region. Draw commands are performed in the intersection of the viewport and the region called area. Coordinates remain related to the viewport. If draw commands try to draw over the area, they are clipped. Set status to DISABLED if you need to draw everything. Set status to ENABLED and region to the desired rectangle to redraw everything inside the region leaving everything else intact. Should be used for redrawing of active elements.

10.1.8 grub_video_get_area_status

- Prototype:

```
grub_err_r
grub_video_get_area_status (grub_video_area_status_t *area_status);
```

- Description: Used to query the area status.

10.1.9 grub_video_set_viewport

- Prototype:

```
grub_err_t
grub_video_set_viewport (unsigned int x, unsigned int y, unsigned int width, unsig
```

- Description:

  Used to specify viewport where draw commands are performed. When viewport is set, all draw commands coordinates relate to those specified by `x` and `y`. If draw commands try to draw over viewport, they are clipped. If developer requests larger than possible viewport, width and height will be clamped to fit screen. If `x` and `y` are out of bounds, all functions drawing to screen will not be displayed. In order to maximize viewport, use `grub_video_get_info` to query actual screen dimensions and provide that information to this function.

10.1.10 grub_video_get_viewport

- Prototype:

```
grub_err_t
grub_video_get_viewport (unsigned int *x, unsigned int *y, unsigned int *width, uns
```

- Description:

  Used to query current viewport dimensions. Software developer can use this to choose best way to render contents of the viewport.

10.1.11 grub_video_set_region

- Prototype:

```
grub_err_t
grub_video_set_region (unsigned int x, unsigned int y, unsigned int width, unsigned
```

- Description:

  Used to specify the region of the screen which should be redrawn. Use absolute values. When the region is set and area status is ENABLE all draw commands will be performed inside the interseption of region and viewport named area. If draw commands try to draw over viewport, they are clipped. If developer requests larger than possible region, width and height will be clamped to fit screen. Should be used for redrawing of active elements.

10.1.12 grub_video_get_region

- Prototype:

```
grub_err_t
grub_video_get_region (unsigned int *x, unsigned int *y, unsigned int *width, unsig
```

- Description:

  Used to query current region dimensions.

10.1.13 grub_video_map_color

- Prototype:

```
grub_video_color_t
grub_video_map_color (grub_uint32_t color_name);
```

- Description:

  Map color can be used to support color themes in GRUB. There will be collection of color names that can be used to query actual screen mapped color data. Examples could be `GRUB_COLOR_CONSOLE_BACKGROUND`, `GRUB_COLOR_CONSOLE_TEXT`. The actual color defines are not specified at this point.

10.1.14 grub_video_map_rgb

- Prototype:

```
grub_video_color_t
grub_video_map_rgb (grub_uint8_t red, grub_uint8_t green, grub_uint8_t blue);
```

- Description:

  Map RGB values to compatible screen color data. Values are expected to be in range 0-255 and in RGB modes they will be converted to screen color data. In index color modes, index color palette will be searched for specified color and then index is returned.

10.1.15 grub_video_map_rgba

- Prototype:

```
grub_video_color_t
grub_video_map_rgba (grub_uint8_t red, grub_uint8_t green, grub_uint8_t blue, grub_
```

- Description:

  Map RGBA values to compatible screen color data. Values are expected to be in range 0-255. In RGBA modes they will be converted to screen color data. In index color modes, index color palette will be searched for best matching color and its index is returned.

10.1.16 grub_video_unmap_color

- Prototype:

```
grub_err_t
grub_video_unmap_color (grub_video_color_t color, grub_uint8_t *red, grub_uint8_t *
```

- Description:

  Unmap color value from `color` to color channels in `red`, `green`, `blue` and `alpha`. Values will be in range 0-255. Active rendering target will be used for color domain. In case alpha information is not available in rendering target, it is assumed to be opaque (having value 255).

10.1.17 grub_video_fill_rect

- Prototype:

```
grub_err_t
grub_video_fill_rect (grub_video_color_t color, int x, int y, unsigned int width, u
```

- Description:

  Fill specified area limited by given coordinates within specified viewport. Negative coordinates are accepted in order to allow easy moving of rectangle within viewport. If coordinates are negative, area of the rectangle will be shrinken to follow size limits of the viewport.

  Software developer should use either `grub_video_map_color`, `grub_video_map_rgb` or `grub_video_map_rgba` to map requested color to `color` parameter.

10.1.18 grub_video_blit_glyph

- Prototype:

```
grub_err_t
grub_video_blit_glyph (struct grub_font_glyph *glyph, grub_video_color_t color, int
```

```
struct grub_font_glyph {
    /* TBD. */
};
```

- Description:

  Used to blit glyph to viewport in specified coodinates. If glyph is at edge of viewport, pixels outside of viewport will be clipped out. Software developer should use either `grub_video_map_rgb` or `grub_video_map_rgba` to map requested color to `color` parameter.

10.1.19 grub_video_blit_bitmap

- Prototype:

```
grub_err_t
grub_video_blit_bitmap (struct grub_video_bitmap *bitmap, enum grub_video_blit_oper
struct grub_video_bitmap
{
    /* TBD. */
};

enum grub_video_blit_operators
  {
    GRUB_VIDEO_BLIT_REPLACE,
    GRUB_VIDEO_BLIT_BLEND
  };
```

- Description:

  Used to blit bitmap to viewport in specified coordinates. If part of bitmap is outside of viewport region, it will be clipped out. Offsets affect bitmap position where data will be copied from. Negative values for both viewport coordinates and bitmap offset coordinates are allowed. If data is looked out of bounds of bitmap, color value will be assumed to be transparent. If viewport coordinates are negative, area of the blitted rectangle will be shrinken to follow size limits of the viewport and bitmap. Blitting operator `oper` specifies should source pixel replace data in screen or blend with pixel alpha value.

  Software developer should use `grub_video_bitmap_create` or `grub_video_bitmap_load` to create or load bitmap data.

10.1.20 grub_video_blit_render_target

- Prototype:

```
grub_err_t
grub_video_blit_render_target (struct grub_video_render_target *source, enum grub_v
struct grub_video_render_target {
    /* This is private data for video driver. Should not be accessed from elsewhere
};

enum grub_video_blit_operators
  {
    GRUB_VIDEO_BLIT_REPLACE,
    GRUB_VIDEO_BLIT_BLEND
  };
```

- Description:

  Used to blit source render target to viewport in specified coordinates. If part of source render target is outside of viewport region, it will be clipped out. If blitting operator is specified and source contains alpha values, resulting pixel color components will be calculated using formula ((src_color * src_alpha) + (dst_color * (255 - src_alpha)) / 255, if target buffer has alpha, it will be set to src_alpha. Offsets affect render target position where data will be copied from. If data is looked out of bounds of render target, color value will be assumed to be transparent. Blitting operator `oper` specifies should source pixel replace data in screen or blend with pixel alpha value.

10.1.21 grub_video_scroll

- Prototype:

```
grub_err_t
grub_video_scroll (grub_video_color_t color, int dx, int dy);
```

- Description:

  Used to scroll viewport to specified direction. New areas are filled with specified color. This function is used when screen is scroller up in video terminal.

10.1.22 grub_video_swap_buffers

- Prototype:

```
grub_err_t
grub_video_swap_buffers (void);
```

- Description:

  If double buffering is enabled, this swaps frontbuffer and backbuffer, in order to show values drawn to back buffer. Video driver is free to choose how this operation is techincally done.

10.1.23 grub_video_create_render_target

- Prototype:

```
grub_err_t
grub_video_create_render_target (struct grub_video_render_target **result, unsigned
```

```
struct grub_video_render_target {
    /* This is private data for video driver. Should not be accessed from elsewhere
};
```

- Description:

  Driver will use information provided to it to create best fitting render target. `mode_type` will be used to guide on selecting what features are wanted for render target. Supported values for `mode_type` are `GRUB_VIDEO_MODE_TYPE_INDEX_COLOR` for index color modes, `GRUB_VIDEO_MODE_TYPE_RGB` for direct RGB color modes and `GRUB_VIDEO_MODE_TYPE_ALPHA` for alpha component.

10.1.24 grub_video_delete_render_target

- Prototype:

```
grub_err_t
grub_video_delete_render_target (struct grub_video_render_target *target);
```

- Description:

  Used to delete previously created render target. If `target` contains `NULL` pointer, nothing will be done. If render target is correctly destroyed, GRUB_ERR_NONE is returned.

10.1.25 grub_video_set_active_render_target

- Prototype:

```
grub_err_t
grub_video_set_active_render_target (struct grub_video_render_target *target);
```

- Description:

  Sets active render target. If this comand is successful all drawing commands will be done to specified `target`. There is also special values for target, `GRUB_VIDEO_RENDER_TARGET_DISPLAY` used to reference screen's front buffer, `GRUB_VIDEO_RENDER_TARGET_FRONT_BUFFER` used to reference screen's front buffer (alias for `GRUB_VIDEO_RENDER_TARGET_DISPLAY`) and `GRUB_VIDEO_RENDER_TARGET_BACK_BUFFER` used to reference back buffer (if double buffering is enabled). If render target is correclty switched GRUB_ERR_NONE is returned. In no any event shall there be non drawable active render target.

10.1.26 grub_video_get_active_render_target

- Prototype:

```
grub_err_t
grub_video_get_active_render_target (struct grub_video_render_target **target);
```

- Description:

  Returns currently active render target. It returns value in `target` that can be subsequently issued back to `grub_video_set_active_render_target`.

10.2 Example usage of Video API

10.2.1 Example of screen setup

```
grub_err_t rc;
/* Try to initialize video mode 1024 x 768 with direct RGB.  */
rc = grub_video_setup (1024, 768, GRUB_VIDEO_MODE_TYPE_RGB);
if (rc != GRUB_ERR_NONE)
{
  /* Fall back to standard VGA Index Color mode.  */
  rc = grub_video_setup (640, 480, GRUB_VIDEO_MODE_TYPE_INDEX);
  if (rc != GRUB_ERR_NONE)
  {
  /* Handle error.  */
  }
}
```

10.2.2 Example of setting up console viewport

```
grub_uint32_t x, y, width, height;
grub_video_color_t color;
struct grub_font_glyph glyph;
grub_err_t rc;
/* Query existing viewport.  */
grub_video_get_viewport (&x, &y, &width, &height);
/* Fill background.  */
color = grub_video_map_color (GRUB_COLOR_BACKGROUND);
grub_video_fill_rect (color, 0, 0, width, height);
/* Setup console viewport.  */
grub_video_set_viewport (x + 10, y + 10, width - 20, height - 20);
grub_video_get_viewport (&x, &y, &width, &height);
color = grub_video_map_color (GRUB_COLOR_CONSOLE_BACKGROUND);
grub_video_fill_rect (color, 0, 0, width, height);
/* Draw text to viewport.  */
color = grub_video_map_color (GRUB_COLOR_CONSOLE_TEXT);
grub_font_get_glyph ('X', &glyph);
grub_video_blit_glyph (&glyph, color, 0, 0);
```

10.3 Bitmap API

10.3.1 grub_video_bitmap_create

- Prototype:

  ```
  grub_err_t grub_video_bitmap_create (struct grub_video_bitmap **bitmap, unsigned in
  ```

- Description:

  Creates a new bitmap with given dimensions and blitting format. Allocated bitmap data can then be modified freely and finally blitted with `grub_video_blit_bitmap` to rendering target.

10.3.2 grub_video_bitmap_destroy

- Prototype:

  ```
  grub_err_t grub_video_bitmap_destroy (struct grub_video_bitmap *bitmap);
  ```

- Description:

  When bitmap is no longer needed, it can be freed from memory using this command. `bitmap` is previously allocated bitmap with `grub_video_bitmap_create` or loaded with `grub_video_bitmap_load`.

10.3.3 grub_video_bitmap_load

- Prototype:

  ```
  grub_err_t grub_video_bitmap_load (struct grub_video_bitmap **bitmap, const char *
  ```

- Description:

  Tries to load given bitmap (`filename`) using registered bitmap loaders. In case bitmap format is not recognized or supported error `GRUB_ERR_BAD_FILE_TYPE` is returned.

10.3.4 grub_video_bitmap_get_width

- Prototype:

```
unsigned int grub_video_bitmap_get_width (struct grub_video_bitmap *bitmap);
```

- Description:

  Returns bitmap width.

10.3.5 grub_video_bitmap_get_height

- Prototype:

```
unsigned int grub_video_bitmap_get_height (struct grub_video_bitmap *bitmap);
```

- Description:

  Return bitmap height.

10.3.6 grub_video_bitmap_get_mode_info

- Prototype:

```
void grub_video_bitmap_get_mode_info (struct grub_video_bitmap *bitmap, struct grul
```

- Description:

  Returns bitmap format details in form of `grub_video_mode_info`.

10.3.7 grub_video_bitmap_get_data

- Prototype:

```
void *grub_video_bitmap_get_data (struct grub_video_bitmap *bitmap);
```

- Description:

  Return pointer to bitmap data. Contents of the pointed data can be freely modified. There is no extra protection against going off the bounds so you have to be carefull how to access the data.

11 PFF2 Font File Format

11.1 Introduction

The goal of this format is to provide a bitmap font format that is simple to use, compact, and cleanly supports Unicode.

11.1.1 Goals of the GRUB Font Format

- Simple to read and use. Since GRUB will only be reading the font files, we are more concerned with making the code to read the font simple than we are with writing the font.
- Compact storage. The fonts will generally be stored in a small boot partition where GRUB is located, and this may be on a removable storage device such as a CD or USB flash drive where space is more limited than it is on most hard drives.
- Unicode. GRUB should not have to deal with multiple character encodings. The font should always use Unicode character codes for simple internationalization.

11.1.2 Why Another Font Format?

There are many existing bitmap font formats that GRUB could use. However, there are aspects of these formats that may make them less than suitable for use in GRUB at this time:

'`BDF`' Inefficient storage; uses ASCII to describe properties and hexadecimal numbers in ASCII for the bitmap rows.

'`PCF`' Many format variations such as byte order and bitmap padding (rows padded to byte, word, etc.) would result in more complex code to handle the font format.

11.2 File Structure

A file **section** consists of a 4-byte name, a 32-bit big-endian length (not including the name or length), and then *length* more section-type-specific bytes.

The standard file extension for PFF2 font files is '`.pf2`'.

11.2.1 Section Types

'`FILE`' **File type ID** (ASCII string). This must be the first section in the file. It has length 4 and the contents are the four bytes of the ASCII string '`PFF2`'.

'`NAME`' **Font name** (ASCII string). This is the full font name including family, weight, style, and point size. For instance, "Helvetica Bold Italic 14".

'`FAMI`' **Font family name** (ASCII string). For instance, "Helvetica". This should be included so that intelligent font substitution can take place.

'`WEIG`' **Font weight** (ASCII string). Valid values are '`bold`' and '`normal`'. This should be included so that intelligent font substitution can take place.

'`SLAN`' **Font slant** (ASCII string). Valid values are '`italic`' and '`normal`'. This should be included so that intelligent font substitution can take place.

'PTSZ' **Font point size** (uint16be).

'MAXW' **Maximum character width in pixels** (uint16be).

'MAXH' **Maximum character height in pixels** (uint16be).

'ASCE' **Ascent in pixels** (uint16be). See Section 11.3 [Font Metrics], page 41, for details.

'DESC' **Descent in pixels** (uint16be). See Section 11.3 [Font Metrics], page 41, for details.

'CHIX' **Character index.** The character index begins with a 32-bit big-endian unsigned integer indicating the total size of the section, not including this size value. For each character, there is an instance of the following entry structure:

- **Unicode code point.** (32-bit big-endian integer.)
- **Storage flags.** (byte.)
  - Bits 2..0:

    If equal to 000 binary, then the character data is stored uncompressed beginning at the offset indicated by the character's **offset** value.

    If equal to 001 binary, then the character data is stored within a compressed character definition block that begins at the offset within the file indicated by the character's **offset** value.
- **Offset.** (32-bit big-endian integer.)

  A marker that indicates the remainder of the file is data accessed via the character index (CHIX) section. When reading this font file, the rest of the file can be ignored when scanning the sections. The length should be set to -1 (0xFFFFFFFF).

  Supported data structures:

  Character definition Each character definition consists of:

  - **Width.** Width of the bitmap in pixels. The bitmap's extents represent the glyph's bounding box. `uint16be`.
  - **Height.** Height of the bitmap in pixels. The bitmap's extents represent the glyph's bounding box. `uint16be`.
  - **X offset.** The number of pixels to shift the bitmap by horizontally before drawing the character. `int16be`.
  - **Y offset.** The number of pixels to shift the bitmap by vertically before drawing the character. `int16be`.
  - **Device width.** The number of pixels to advance horizontally from this character's origin to the origin of the next character. `int16be`.
  - **Bitmap data.** This is encoded as a string of bits. It is organized as a row-major, top-down, left-to-right bitmap. The most significant bit of each byte is taken to be the leftmost or uppermost bit in the byte. For the sake of compact storage, rows are not padded to byte boundaries (i.e., a single byte may contain bits belonging to multiple rows). The last byte of the bitmap **is** padded with zero bits in the bits positions to the right of the last used bit if the bitmap data does not fill the last byte.

The length of the **bitmap data** field is (*width* * *height* + 7) / 8 using integer arithmetic, which is equivalent to ceil(*width* * *height* / 8) using real number arithmetic.

It remains to be determined whether bitmap fonts usually make all glyph bitmaps the same height, or if smaller glyphs are stored with bitmaps having a lesser height. In the latter case, the baseline would have to be used to calculate the location the bitmap should be anchored at on screen.

11.3 Font Metrics

- Ascent. The distance from the baseline to the top of most characters. Note that in some cases characters may extend above the ascent.
- Descent. The distance from the baseline to the bottom of most characters. Note that in some cases characters may extend below the descent.
- Leading. The amount of space, in pixels, to leave between the descent of one line of text and the ascent of the next line. This metrics is not specified in the current file format; instead, the font rendering engine calculates a reasonable leading value based on the other font metrics.
- Horizonal leading. The amount of space, in pixels, to leave horizontally between the left and right edges of two adjacent glyphs. The **device width** field determines the effective leading value that is used to render the font.

An illustration of how the various font metrics apply to characters.

12 Graphical Menu Software Design

12.1 Introduction

The 'gfxmenu' module provides a graphical menu interface for GRUB 2. It functions as an alternative to the menu interface provided by the 'normal' module, which uses the grub terminal interface to display a menu on a character-oriented terminal.

The graphical menu uses the GRUB video API, which is currently for the VESA BIOS extensions (VBE) 2.0+. This is supported on the i386-pc platform. However, the graphical menu itself does not depend on using VBE, so if another GRUB video driver were implemented, the 'gfxmenu' graphical menu would work on the new video driver as well.

12.2 Startup Sequence

- grub_enter_normal_mode [normal/main.c]
- grub_normal_execute [normal/main.c]
- read_config_file [normal/main.c]
- (When 'gfxmenu.mod' is loaded with `insmod`, it will call `grub_menu_viewer_register()` to register itself.)
- GRUB_MOD_INIT (gfxmenu) [gfxmenu/gfxmenu.c]
- grub_menu_viewer_register [kern/menu_viewer.c]
- grub_menu_viewer_show_menu [kern/menu_viewer.c]
- get_current_menu_viewer() [kern/menu_viewer.c]
- show_menu() [gfxmenu/gfxmenu.c]
- grub_gfxmenu_model_new [gfxmenu/model.c]
- grub_gfxmenu_view_new [gfxmenu/view.c]
- set_graphics_mode [gfxmenu/view.c]
- grub_gfxmenu_view_load_theme [gfxmenu/theme_loader.c]

12.3 GUI Components

The graphical menu implements a GUI component system that supports a container-based layout system. Components can be added to containers, and containers (which are a type of component) can then be added to other containers, to form a tree of components. Currently, the root component of this tree is a 'canvas' component, which allows manual layout of its child components.

Components (non-container):

- label
- image
- progress_bar
- circular_progress
- list (currently hard coded to be a boot menu list)

Containers:

- canvas
- hbox
- vbox

The GUI component instances are created by the theme loader in '`gfxmenu/theme_loader.c`' when a theme is loaded. Theme files specify statements such as '`+vbox{ +label { text="Hello" } +label{ text="World" } }`' to add components to the component tree root. By nesting the component creation statements in the theme file, the instantiated components are nested the same way.

When a component is added to a container, that new child is considered **owned** by the container. Great care should be taken if the caller retains a reference to the child component, since it will be destroyed if its parent container is destroyed. A better choice instead of storing a pointer to the child component is to use the component ID to find the desired component. Component IDs do not have to be unique (it is often useful to have multiple components with an ID of "__timeout__", for instance).

In order to access and use components in the component tree, there are two functions (defined in '`gfxmenu/gui_util.c`') that are particularly useful:

- `grub_gui_find_by_id (root, id, callback, userdata)`:

  This function ecursively traverses the component tree rooted at *root*, and for every component that has an ID equal to *id*, calls the function pointed to by *callback* with the matching component and the void pointer *userdata* as arguments. The callback function can do whatever is desired to use the component passed in.
- `grub_gui_iterate_recursively (root, callback, userdata)`:

  This function calls the function pointed to by *callback* for every component that is a descendant of *root* in the component tree. When the callback function is called, the component and the void pointer *userdata* as arguments. The callback function can do whatever is desired to use the component passed in.

12.4 Command Line Window

The terminal window used to provide command line access within the graphical menu is managed by '`gfxmenu/view.c`'. The '`gfxterm`' terminal is used, and it has been modified to allow rendering to an offscreen render target to allow it to be composed into the double buffering system that the graphical menu view uses. This is bad for performance, however, so it would probably be a good idea to make it possible to temporarily disable double buffering as long as the terminal window is visible. There are still unresolved problems that occur when commands are executed from the terminal window that change the graphics mode. It's possible that making `grub_video_restore()` return to the graphics mode that was in use before `grub_video_setup()` was called might fix some of the problems.

13 Verifiers framework

To register your own verifier call ‘`grub_verifier_register`’ with a structure pointing to your functions.

The interface is inspired by the hash interface with ‘`init`’/‘`write`’/‘`fini`’.

There are essentially 2 ways of using it, hashing and whole-file verification.

With the hashing approach: During ‘`init`’ you decide whether you want to check the given file and init context. In ‘`write`’ you update your hashing state. In ‘`fini`’ you check that the hash matches the expected value/passes some check/...

With whole-file verification: During ‘`init`’ you decide whether you want to check the given file and init context. In ‘`write`’ you verify the file and return an error if it fails. You don’t have ‘`fini`’.

Additional ‘`verify_string`’ receives various strings like kernel parameters to verify. Returning no error means successful verification and an error stops the current action.

Detailed description of the API:

Every time a file is opened your ‘`init`’ function is called with file descriptor and file type. Your function can have the following outcomes:

- returning no error and setting ‘`*flags`’ to ‘`GRUB_VERIFY_FLAGS_DEFER_AUTH`’. In this case verification is deferred to other active verifiers. Verification fails if nobody cares or selected verifier fails.
- returning no error and setting ‘`*flags`’ to ‘`GRUB_VERIFY_FLAGS_SKIP_VERIFICATION`’. In this case your verifier will not be called anymore and it is assumed to have skipped verification.
- returning no error and not setting ‘`*flags`’ to ‘`GRUB_VERIFY_FLAGS_SKIP_VERIFICATION`’ In this case verification is done as described in the following section.
- returning an error. Then opening of the file will fail due to failed verification.

In the third case your ‘`write`’ will be called with chunks of the file. If you need the whole file in a single chunk then during ‘`init`’ set the bit ‘`GRUB_VERIFY_FLAGS_SINGLE_CHUNK`’ in ‘`*flags`’. During ‘`init`’ you may set ‘`*context`’ if you need additional context. At every iteration you may return an error and the file will be considered as having failed the verification. If you return no error then verification continues.

Optionally at the end of the file ‘`fini`’, if it exists, is called with just the context. If you return no error during any of ‘`init`’, ‘`write`’ and ‘`fini`’ then the file is considered as having succeded verification.

14 Lockdown framework

The GRUB can be locked down, which is a restricted mode where some operations are not allowed. For instance, some commands cannot be used when the GRUB is locked down.

The function `grub_lockdown()` is used to lockdown GRUB and the function `grub_is_lockdown()` function can be used to check whether lockdown is enabled or not. When enabled, the function returns '`GRUB_LOCKDOWN_ENABLED`' and '`GRUB_LOCKDOWN_DISABLED`' when is not enabled.

The following functions can be used to register the commands that can only be used when lockdown is disabled:

- `grub_cmd_lockdown()` registers command which should not run when the GRUB is in lockdown mode.
- `grub_cmd_lockdown()` registers extended command which should not run when the GRUB is in lockdown mode.

Appendix A Copying This Manual

A.1 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 noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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

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

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

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.

A.1.1 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.

Index