GTK+ FAQ

Tony Gale

Shawn Amundson

Emmanuel Deloget

GTK+ FAQ

by Tony Gale, Shawn Amundson, and Emmanuel Deloget

This document is intended to answer questions that are likely to be frequently asked by programmers using GTK+ or people who are just looking at using GTK+.

Table of Contents

Chapter 1. General Information

1.1. Before anything else: the greetings

The FAQ authors want to thank:

- Havoc Pennington
- Erik Mouw
- Owen Taylor
- Tim Janik
- Thomas Mailund Jensen
- Joe Pfeiffer
- Andy Kahn
- Federico Mena Quntero
- Damon Chaplin
- and all the members of the GTK+ lists

If we forgot you, please email us! Thanks again (I know, it's really short :)

1.2. Authors

The original authors of GTK+ were:

- Peter Mattis
- Spencer Kimball
- Josh MacDonald

Since then, much has been added by others. Please see the AUTHORS file in the distribution for the GTK+ Team.

1.3. What is GTK+?

GTK+ is a small and efficient widget set designed with the general look and feel of Motif. In reality, it looks much better than Motif. It contains common widgets and some more complex widgets such as a file selection, and color selection widgets.

GTK+ provides some unique features. (At least, I know of no other widget library which provides them). For example, a button does not contain a label, it contains a child widget, which in most instances will be a label. However, the child widget can also be a pixmap, image or any combination possible the programmer desires. This flexibility is adhered to throughout the library.

1.4. What is the + in GTK+?

Peter Mattis informed the gtk mailing list that:

"I originally wrote gtk which included the three libraries, libglib, libgdk and libgtk. It featured a flat widget hierarchy. That is, you couldn't derive a new widget from an existing one. And it contained a more standard callback mechanism instead of the signal mechanism now present in gtk+. The + was added to distinguish between the original version of gtk and the new version. You can think of it as being an enhancement to the original gtk that adds object oriented features."

1.5. Does the G in GTK+, GDK and GLib stand for?

GTK+ == Gimp Toolkit

GDK == GTK+ Drawing Kit

GLib == G Libray

1.6. Where is the documentation for GTK+?

In the GTK+ distribution's doc/ directory you will find the reference material for both GTK and GDK, this FAQ and the GTK Tutorial.

In addition, you can find links to HTML versions of these documents by going to http://www.gtk.org/. A packaged version of the GTK Tutorial, with SGML, HTML, Postscript, DVI and text versions can be found in ftp://ftp.gtk.org/pub/gtk/tutorial

There are now a couple of books available that deal with programming GTK+, GDK and GNOME:

- Eric Harlows book entitled "Developing Linux Applications with GTK+ and GDK". The ISBN is 0-7357-0021-4
- The example code from Eric's book is available on-line at http://www.bcpl.net/~eharlow/book
- Havoc Pennington has released a book called "GTK+/GNOME Application Development". The ISBN is 0-7357-0078-8

  The free version of the book lives here: http://developer.gnome.org/doc/GGAD/

  And Havoc maintains information about it and errata here: http://pobox.com/~hp/gnome-app-devel.html

1.7. Is there a mailing list (or mailing list archive) for GTK+?

Information on mailing lists relating to GTK+ can be found at: http://www.gtk.org/mailinglists.html

1.8. How to get help with GTK+

First, make sure your question isn't answered in the documentation, this FAQ or the tutorial. Done that? You're sure you've done that, right? In that case, the best place to post questions is to the GTK+ mailing list.

1.9. How to report bugs in GTK+

Bugs should be reported to the GNOME bug tracking system (http://bugzilla.gnome.org). You will need to enter your email address and receive a password before you can use the system to register a new bug report.

There are a number of options to select and boxes to fill in when submitting a bug report. Please remember that the more information you give, the easier it will be to track the problem down. Extra information that may prove useful includes:

- How to reproduce the bug.

  If you can reproduce it with the testgtk program that is built in the gtk/ subdirectory, that will be most convenient. Otherwise, please include a short test program that exhibits the behavior. As a last resort, you can also provide a pointer to a larger piece of software that can be downloaded.

  (Bugs that can be reproduced within the GIMP are almost as good as bugs that can be reproduced in testgtk. If you are reporting a bug found with the GIMP, please include the version number of the GIMP you are using)

- If the bug was a crash, the exact text that was printed out when the crash occurred.
- Further information such as stack traces may be useful, but are not necessary. If you do send a stack trace, and the error is an X error, it will be more useful if the stacktrace is produced running the test program with the `--sync` command line option.

1.10. Is there a Windows version of GTK+?

There is an on going port of GTK+ to the Windows platform which is making impressive progress.

See http://www.iki.fi/tml/gimp/win32 for more information.

1.11. What applications have been written with GTK+?

A list of some GTK+ based application can be found on the GTK+ web server at http://www.gtk.org/apps/ and contains more than 350 applications.

Failing that, look for a project to work on for the GNOME project, http://www.gnome.org/ Write a game. Write something that is useful.

Some of these are:

- GIMP (http://www.gimp.org/), an image manipulation program
- AbiWord (http://www.abisource.com/), a professional word processor
- Gzilla (http://www.levien.com/gzilla/), a web browser
- XQF (http://www.botik.ru/~roma/quake/), a QuakeWorld/Quake2 server browser and launcher
- GDK Imlib (http://www.rasterman.com/imlib.html), a fast image loading and manipulation library for GDK
- Glade (http://glade.pn.org/), a GTK+ based RAD tool which produces GTK+ applications

1.12. I’m looking for an application to write in GTK+. How about an IRC client?

Ask on gtk-list for suggestions. There are at least three IRC clients already under development (probably more in fact. The server at http://www.forcix.cx/irc-clients.html list a bunch of them).

- X-Chat.
- girc. (Included with GNOME)
- gsirc. (In the gnome CVS tree)

Chapter 2. How to find, configure, install, and troubleshoot GTK+

2.1. What do I need to run GTK+?

To compile GTK+, all you need is a C compiler (gcc) and the X Window System and associated libraries on your system.

2.2. Where can I get GTK+?

The canonical site is ftp://ftp.gtk.org/pub/gtk.

This site tends to get busy around the time of a new GTK+ release so try and use one of the mirror sites that are listed in ftp://ftp.gtk.org/etc/mirrors

Here's a few mirror sites to get you started:

- Africa - ftp://ftp.is.co.za/applications/gimp
- Australia - ftp://ftp.au.gimp.org/pub/gimp
- Finland - ftp://ftp.funet.fi/pub/sci/graphics/packages/gimp
- Germany - ftp://infosoc.uni-koeln.de/pub/ftp.gimp.org" (ftp://infosoc.uni-koeln.de/pub/ftp.gimp.org)
- Japan - ftp://SunSITE.sut.ac.jp/pub/archives/packages/gimp
- UK - ftp://ftp.flirble.org/pub/X/gimp
- US - ftp://ftp.insync.net/pub/mirrors/ftp.gimp.org

2.3. How do I configure/compile GTK+?

Generally, all you will need to do is issue the commands:

```
./configure
make
```

in the gtk+-version/ directory.

2.4. When compiling GTK+ I get an error like: `make: file 'Makefile' line 456: Syntax error`

Make sure that you are using GNU make (use `make -v` to check). There are many weird and wonderful versions of make out there, and not all of them handle the automatically generated Makefiles.

2.5. I've compiled and installed GTK+, but I can't get any programs to link with it!

This problem is most often encountered when the GTK+ libraries can't be found or are the wrong version. Generally, the compiler will complain about an 'unresolved symbol'. There are two things you need to check:

- Make sure that the libraries can be found. You want to edit `/etc/ld.so.conf` to include the directories which contain the GTK libraries, so it looks something like:

```
/usr/X11R6/lib
/usr/local/lib
```

  Then you need to run /sbin/ldconfig as root. You can find what directory GTK is in using

```
gtk-config --libs
```

  If your system doesn't use ld.so to find libraries (such as Solaris), then you will have to use the LD_LIBRARY_PATH environment variable (or compile the path into your program, which I'm not going to cover here). So, with a Bourne type shell you can do (if your GTK libraries are in /usr/local/lib):

```
export LD_LIBRARY_PATH=/usr/local/lib
```

  and in a csh, you can do:

```
setenv LD_LIBRARY_PATH /usr/local/lib
```

- Make sure the linker is finding the correct set of libraries. If you have a Linux distribution that installs GTK+ (e.g. RedHat 5.0) then this older version may be used. Now (assuming you have a RedHat system), issue the command

  ```
  rpm -e gtk gtk-devel
  ```

  You may also want to remove the packages that depend on gtk (rpm will tell you which ones they are). If you don't have a RedHat Linux system, check to make sure that neither `/usr/lib` or `/usr/local/lib` contain any of the libraries libgtk, libgdk, libglib, or libgck. If they do exist, remove them (and any gtk include files, such as `/usr/include/gtk` and `/usr/include/gdk`) and reinstall gtk+.

2.6. When compiling programs with GTK+, I get compiler error messages about not being able to find `glibconfig.h`.

The header file "glibconfig.h" was moved to the directory $exec_prefix/lib/glib/include/. $exec_prefix is the directory that was specified by giving the --exec-prefix flags to ./configure when compiling GTK+. It defaults to $prefix, (specified with --prefix), which in turn defaults to /usr/local/.

This was done because "glibconfig.h" includes architecture dependent information, and the rest of the include files are put in $prefix/include, which can be shared between different architectures.

GTK+ includes a shell script, `/gtk-config/`, that makes it easy to find out the correct include paths. The GTK+ Tutorial includes an example of using `/gtk-config/` for simple compilation from the command line. For information about more complicated configuration, see the file docs/gtk-config.txt in the GTK+ distribution.

If you are trying to compile an old program, you may be able to work around the problem by configuring it with a command line like:

```
setenv CPPFLAGS "-I/usr/local/include/glib/include"
./configure
```

(Substitute the appropriate value of $exec_prefix for /usr/local.)

2.7. When installing a GTK+ application, configure reports that it can’t find GTK.

There are several common reasons for this:

- You have an old version of GTK installed somewhere. RedHat 5.0, for example, installs an older copy of GTK that may not work with the latest applications. You should remove this old copy, but note that in the case of RedHat 5.0 this will break the `control-panel` applications.
- `gtk-config` (or another component of GTK) isn’t in your path, or there is an old version on your system. Type:

  ```
  gtk-config --version
  ```

  to check for both of these. If it returns a value different from what you expect, then you have an old version of GTK on your system.

- The ./configure script can’t find the GTK libraries. As ./configure compiles various test programs, it needs to be able to find the GTK libraries. See the question above for help on this.

If none of the above help, then have a look in config.log, which is generated by ./configure as it runs. At the bottom will be the last action it took before failing. If it is a section of source code, copy the source code to a file and compile it with the line just above it in config.log. If the compilation is successful, try executing it.

Chapter 3. Development of GTK+

3.1. Whats this CVS thing that everyone keeps talking about, and how do I access it?

CVS is the Concurent Version System and is a very popular means of version control for software projects. It is designed to allow multiple authors to be able to simultanously operate on the same source tree. This source tree is centrally maintained, but each developer has a local mirror of this repository that they make there changes to.

The GTK+ developers use a CVS repository to store the master copy of the current development version of GTK+. As such, people wishing to contribute patches to GTK+ should generate them against the CVS version. Normal people should use the packaged releases.

The CVS toolset is available as RPM packages from the usual RedHat sites. The latest version is available at http://download.cyclic.com/pub/

Anyone can download the latest CVS version of GTK+ by using anonymous access using the following steps:

- In a bourne shell descendant (e.g. bash) type:

```
CVSROOT=':pserver:anonymous@anoncvs.gnome.org:/cvs/gnome'
export CVSROOT
```

- Next, the first time the source tree is checked out, a cvs login is needed.

```
cvs login
```

  This will ask you for a password. There is no password for cvs.gimp.org, so just enter a carriage return.

- To get the tree and place it in a subdir of your current working directory, issue the command:

```
cvs -z3 get gtk+
```

Note that with the GTK+ 1.1 tree, glib has been moved to a separate CVS module, so if you don't have glib installed you will need to get that as well:

```
cvs -z3 get glib
```

3.2. How can I contribute to GTK+?

It's simple. If something doesn't work like you think it should in a program, check the documentation to make sure you're not missing something. If it is a true bug or missing feature, track it down in the GTK+ source, change it, and then generate a patch in the form of a 'context diff'. This can be done using a command such as `diff -ru <oldfile> <newfile>`. Then upload the patchfile to:

```
ftp://ftp.gtk.org/incoming
```

along with a README file. Make sure you follow the naming conventions or your patch will just be deleted! The filenames should be of this form:

```
gtk<username>-<date yymmdd-n>.patch.gz
gtk-<username>-<date yymmdd-n>.patch.README
```

The "n" in the date indicates a unique number (starting from 0) of patches you uploaded that day. It should be 0, unless you upload more than one patch in the same day.

Example:

```
gtk-gale-982701-0.patch.gz
gtk-gale-982701-0.patch.README
```

Once you upload *anything*, send the README to ftp-admin@gtk.org

3.3. How do I know if my patch got applied, and if not, why not?

Uploaded patches will be moved to `ftp://ftp.gtk.org/pub/gtk/patches` where one of the GTK+ development team will pick them up. If applied, they will be moved to `/pub/gtk/patches/old`.

Patches that aren't applied, for whatever reason, are moved to `/pub/gtk/patches/unapplied` or `/pub/gtk/patches/outdated`. At this point you can ask on the `gtk-list` mailing list why your patch wasn't applied. There are many possible reasons why patches may not be applied, ranging from it doesn't apply cleanly, to it isn't right. Don't be put off if your patch didn't make it first time round.

3.4. What is the policy on incorporating new widgets into the library?

This is up to the authors, so you will have to ask them once you are done with your widget. As a general guideline, widgets that are generally useful, work, and are not a disgrace to the widget set will gladly be included.

3.5. Is anyone working on bindings for languages other than C?

The GTK+ home page (http://www.gtk.org/) presents a list of GTK+ bindings.

- There are several C++ wrappers for GTK+.
  - the gtk-- package, which is a very small wrapper for GTK+. You can find the home page at http://www.cs.tut.fi/~p150650/gtk/gtk--.html (http://www.cs.tut.fi/~p150650/gtk/gtk--.html). The FTP site is ftp://ftp.gtk.org/pub/gtk/gtk-- (ftp://ftp.gtk.org/pub/gtk/gtk--).
  - the VDK package, which was built as the base package of a GTK+ application Borland-like builder. The home page can be found at http://www.guest.net/homepages/mmotta/VDKHome (http://www.guest.net/homepages/mmotta/VDKHome).
  - The wxWindows/Gtk package, a free C++ library for cross-platform GUI development. The home page of this package is http://www.freiburg.linux.de/~wxxt/ (http://www.freiburg.linux.de/~wxxt/).

- There are three known Objective-c bindings currently in development:
  - The http://www.gnome.org/ package of choice is objgtk. Objgtk is based on the Object class and is maintained by Elliot Lee (mailto:sopwith@cuc.edu). Apparently, objgtk is being accepted as the 'standard' Objective-C binding for GTK+.
  - If you are more inclined towards the GNUstep project (http://www.gnustep.org/), you may want to check out GTKKit by Helge Heß (mailto:helge@mdlink.de). The intention is to setup a GTK+ binding using the FoundationKit. GTKKit includes nicities like writing a XML-type template file to construct a GTK+ interface.
  - The GToolKit package, which can be found at ftp://ftp.gtk.org/pub/gtk/objc-gtoolkit/ (ftp://ftp.gtk.org/pub/gtk/objc-gtoolkit/).

- Perl bindings ftp://ftp.gtk.org/pub/gtk/perl
- Guile bindings. The home page is at http://www.ping.de/sites/zagadka/guile-gtk. By the way, Guile is the GNU Project's implemention of R4RS Scheme (the standard). If you like Scheme, you may want to take a look at this.
- David Monniaux reports: "I've started a gtk-O'Caml binding system. The basics of the system, including callbacks, work fine. The current development is in http://www.ens-lyon.fr/~dmonniau/arcs"
- Several python bindings have been done:
  - pygtk is at http://www.daa.com.au/~james/pygtk (http://www.daa.com.au/~james/pygtk) and ftp://ftp.gtk.org/pub/gtk/python (ftp://ftp.gtk.org/pub/gtk/python)
  - python-gtk is at http://www.ucalgary.ca/~nascheme/python-gtk (http://www.ucalgary.ca/~nascheme/python-gtk)

- There's are a couple of OpenGL/Mesa widgets available for GTK+. I suggest you start at http://www.student.oulu.fi/~jlof/gtkglarea/index.html
- Last, there are a lot of other language bindings for languages such as Eiffel, TOM, Pascal, Pike, etc.

Chapter 4. Development with GTK+: the begining

4.1. How do I get started?

So, after you have installed GTK+ there are a couple of things that can ease you into developing applications with it. There is the GTK+ Tutorial http://www.gtk.org/tutorial/, which is undergoing development. This will introduce you to writing applications using C.

The Tutorial doesn't (yet) contain information on all of the widgets that are in GTK+. For example code on how to use the basics of all the GTK+ widgets you should look at the file gtk/testgtk.c (and associated source files) within the GTK+ distribution. Looking at these examples will give you a good grounding on what the widgets can do.

4.2. How do I write security sensitive/SUID/SGID programs with GTK+? Is GTK+ secure? What's this GTK_MODULES security hole I heard about?

The short answer to this question is: Don't write SUID/SGID programs with GTK+

For a more thorough explanation of the GTK+ Developers position on this issue see http://www.gtk.org/setuid.html.

4.3. I tried to compile a small Hello World of mine, but it failed. Any clue?

Since you are good at coding, we will not deal with compile time error here :)

The classic command line to compile a GTK+ based program is

```
gcc -o myprog [c files] ‘gtk-config --cflags --libs‘
```

You should notice the backquote character which is used in this command line. A common mistake when you start a GTK+ based development is to use quote instead of backquotes. If you do so, the compiler

will complain about an unknown file called `gtk-config --cflags --libs`. The text in backquotes is an instruction to your shell to substitute the output of executing this text into the commandline.

The command line above ensure that:

- the correct C compiler flags will be used to compile the program (including the complete C header directory list)
- your program will be linked with the needed libraries.

4.4. What about using the make utility?

This is a sample makefile which compile a GTK+ based program:

```
# basic GTK+ app makefile
SOURCES = myprg.c foo.c bar.c
OBJS    = ${SOURCES:.c=.o}
CFLAGS  = ‘gtk-config --cflags‘
LDADD   = ‘gtk-config --libs‘
CC      = gcc
PACKAGE = myprg

all : ${OBJS}
  ${CC} -o ${PACKAGE} ${OBJS} ${LDADD}

.c.o:
  ${CC} ${CFLAGS} -c $<

# end of file
```

For more information about the **make** utility, you should read either the related man page or the relevant info file.

4.5. I use the backquote stuff in my makefiles, but my make process failed.

The backquote construction seems to not be accepted by some old **make** utilities. If you use one of these, the make process will probably fail. In order to have the backquote syntax working again, you should use the GNU make utility (get it on the GNU ftp server at ftp://ftp.gnu.org/).

4.6. I want to add some configure stuff, how could I do this?

To use autoconf/automake, you must first install the relevant packages. These are:

- the m4 preprocessor v1.4 or better
- autoconf v2.13 or better
- automake v1.4 or better

You'll find these packages on the GNU main ftp server (ftp://ftp.gnu.org/) or on any GNU mirror.

In order to use the powerful autoconf/automake scheme, you must create a configure.in which may look like:

```
dnl Process this file with autoconf to produce a configure script.
dnl configure.in for a GTK+ based program

AC_INIT(myprg.c)dnl
AM_INIT_AUTOMAKE(mypkgname,0.0.1)dnl
AM_CONFIG_HEADER(config.h)dnl

dnl Checks for programs.
AC_PROG_CC dnl check for the c compiler
dnl you should add CFLAGS="" here, 'cos it is set to -g by PROG_CC

dnl Checks for libraries.
AM_PATH_GTK(1.2.0„AC_MSG_ERROR(mypkgname 0.1 needs GTK))dnl

AC_OUTPUT(
Makefile
)dnl
```

You must add a Makefile.am file:

```
bin_PROGRAMS    = myprg
myprg_SOURCES   = myprg.c foo.c bar.c
INCLUDES        = @GTK_CFLAGS@
LDADD           = @GTK_LIBS@
CLEANFILES      = *~
DISTCLEANFILES  = .deps/*.P
```

If your project contains more than one subdirectory, you'll have to create one Makefile.am in each directory plus a master Makefile.am which will look like:

```
SUBDIRS         = mydir1 mydir2 mydir3
```

then, to use these, simply type the following commands:

```
aclocal
autoheader
autoconf
automake --add-missing --include-deps --foreign
```

For further information, you should look at the autoconf and the automake documentation (the shipped info files are really easy to understand, and there are plenty of web resources that deal with autoconf and automake).

4.7. I try to debug my GTK+ application with gdb, but it hangs my X server when I hit some breakpoint. Any Idea?

From Federico Mena Quintero:

“X is not locked up. It is likely that you are hitting a breakpoint inside a callback that is called from a place in Gtk that has a mouse grab.”

“Run your program with the `--sync` option; it will make it easier to debug. Also, you may want to use the console for running the debugger, and just let the program run in another console with the X server.”

Eric Mouw had another solution:

“An old terminal connected to an otherwise unused serial port is also great for debugging X programs. Old vt100/vt220 terminals are dirt cheap but a bit hard to get (here in The Netherlands, YMMV).”

Chapter 5. Development with GTK+: general questions

5.1. What widgets are in GTK?

The GTK+ Tutorial lists the following widgets:

```
GtkObject
 +GtkData
 | +GtkAdjustment
 | ‘GtkTooltips
 ‘GtkWidget
   +GtkContainer
   | +GtkBin
   | | +GtkAlignment
   | | +GtkEventBox
   | | +GtkFrame
   | | | ‘GtkAspectFrame
   | | +GtkHandleBox
   | | +GtkItem
   | | | +GtkListItem
   | | | +GtkMenuItem
   | | | | ‘GtkCheckMenuItem
   | | | |   ‘GtkRadioMenuItem
   | | | ‘GtkTreeItem
   | | +GtkViewport
   | | ‘GtkWindow
   | |   +GtkColorSelectionDialog
   | |   +GtkDialog
   | |   | ‘GtkInputDialog
   | |   ‘GtkFileSelection
   | +GtkBox
   | | +GtkButtonBox
   | | | +GtkHButtonBox
   | | | ‘GtkVButtonBox
   | | +GtkHBox
   | | | +GtkCombo
   | | | ‘GtkStatusbar
   | | ‘GtkVBox
   | |   +GtkColorSelection
   | |   ‘GtkGammaCurve
   | +GtkButton
   | | +GtkOptionMenu
   | | ‘GtkToggleButton
   | |   ‘GtkCheckButton
   | |     ‘GtkRadioButton
   | +GtkCList
   |   ‘GtkCTree
   | +GtkFixed
```

```
| +GtkList
| +GtkMenuShell
| | +GtkMenuBar
| | ‘GtkMenu
| +GtkNotebook
| +GtkPaned
| | +GtkHPaned
| | ‘GtkVPaned
| +GtkScrolledWindow
| +GtkTable
| +GtkToolbar
| ‘GtkTree
+GtkDrawingArea
| ‘GtkCurve
+GtkEditable
| +GtkEntry
| | ‘GtkSpinButton
| ‘GtkText
+GtkMisc
| +GtkArrow
| +GtkImage
| +GtkLabel
| | ‘GtkTipsQuery
| ‘GtkPixmap
+GtkPreview
+GtkProgressBar
+GtkRange
| +GtkScale
| | +GtkHScale
| | ‘GtkVScale
| ‘GtkScrollbar
|   +GtkHScrollbar
|   ‘GtkVScrollbar
+GtkRuler
| +GtkHRuler
| ‘GtkVRuler
‘GtkSeparator
  +GtkHSeparator
  ‘GtkVSeparator
```

5.2. Is GTK+ thread safe? How do I write multi-threaded GTK+ applications?

The GLib library can be used in a thread-safe mode by calling g_thread_init() before making any other GLib calls. In this mode GLib automatically locks all internal data structures as needed. This does not mean that two threads can simultaneously access, for example, a single hash table, but they can access two different hash tables simultaneously. If two different threads need to access the same hash table, the application is responsible for locking itself.

When GLib is intialized to be thread-safe, GTK+ is *thread aware*. There is a single global lock that you must acquire with gdk_threads_enter() before making any GDK calls, and release with gdk_threads_leave() afterwards.

A minimal main program for a threaded GTK+ application looks like:

```
int
main (int argc, char *argv[])
{
  GtkWidget *window;

  g_thread_init(NULL);
  gtk_init(&argc, &argv);

  window = create_window();
  gtk_widget_show(window);

  gdk_threads_enter();
  gtk_main();
  gdk_threads_leave();

  return(0);
}
```

Callbacks require a bit of attention. Callbacks from GTK+ (signals) are made within the GTK+ lock. However callbacks from GLib (timeouts, IO callbacks, and idle functions) are made outside of the GTK+ lock. So, within a signal handler you do not need to call gdk_threads_enter(), but within the other types of callbacks, you do.

Erik Mouw contributed the following code example to illustrate how to use threads within GTK+ programs.

```
/*-----------------------------------------------------------------------
 * Filename:      gtk-thread.c
 * Version:       0.99.1
 * Copyright:     Copyright (C) 1999, Erik Mouw
 * Author:        Erik Mouw <J.A.K.Mouw@its.tudelft.nl>
 * Description:   GTK threads example.
 * Created at:    Sun Oct 17 21:27:09 1999
 * Modified by:   Erik Mouw <J.A.K.Mouw@its.tudelft.nl>
 * Modified at:   Sun Oct 24 17:21:41 1999
 *-----------------------------------------------------------------------*/
/*
 * Compile with:
 *
 * cc -o gtk-thread gtk-thread.c `gtk-config --cflags --libs gthread`
 *
 * Thanks to Sebastian Wilhelmi and Owen Taylor for pointing out some
 * bugs.
 *
```

```
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <gtk/gtk.h>
#include <glib.h>
#include <pthread.h>

#define YES_IT_IS    (1)
#define NO_IT_IS_NOT (0)

typedef struct
{
  GtkWidget *label;
  int what;
} yes_or_no_args;

G_LOCK_DEFINE_STATIC (yes_or_no);
static volatile int yes_or_no = YES_IT_IS;

void destroy(GtkWidget *widget, gpointer data)
{
  gtk_main_quit();
}

void *argument_thread(void *args)
{
  yes_or_no_args *data = (yes_or_no_args *)args;
  gboolean say_something;

  for(;;)
    {
      /* sleep a while */
      sleep(rand() / (RAND_MAX / 3) + 1);

      /* lock the yes_or_no_variable */
      G_LOCK(yes_or_no);

      /* do we have to say something? */
      say_something = (yes_or_no != data->what);

      if(say_something)
{
  /* set the variable */
  yes_or_no = data->what;
}

      /* Unlock the yes_or_no variable */
      G_UNLOCK(yes_or_no);

      if(say_something)
```

```
{
  /* get GTK thread lock */
  gdk_threads_enter();

  /* set label text */
  if(data->what == YES_IT_IS)
    gtk_label_set_text(GTK_LABEL(data->label), "O yes, it is!");
  else
    gtk_label_set_text(GTK_LABEL(data->label), "O no, it isn't!");

  /* release GTK thread lock */
  gdk_threads_leave();
}
    }

  return(NULL);
}

int main(int argc, char *argv[])
{
  GtkWidget *window;
  GtkWidget *label;
  yes_or_no_args yes_args, no_args;
  pthread_t no_tid, yes_tid;

  /* init threads */
  g_thread_init(NULL);

  /* init gtk */
  gtk_init(&argc, &argv);

  /* init random number generator */
  srand((unsigned int)time(NULL));

  /* create a window */
  window = gtk_window_new(GTK_WINDOW_TOPLEVEL);

  gtk_signal_connect(GTK_OBJECT (window), "destroy",
     GTK_SIGNAL_FUNC(destroy), NULL);

  gtk_container_set_border_width(GTK_CONTAINER (window), 10);

  /* create a label */
  label = gtk_label_new("And now for something completely different ...");
  gtk_container_add(GTK_CONTAINER(window), label);

  /* show everything */
  gtk_widget_show(label);
  gtk_widget_show (window);

  /* create the threads */
  yes_args.label = label;
  yes_args.what = YES_IT_IS;
```

```
  pthread_create(&yes_tid, NULL, argument_thread, &yes_args);

  no_args.label = label;
  no_args.what = NO_IT_IS_NOT;
  pthread_create(&no_tid, NULL, argument_thread, &no_args);

  /* enter the GTK main loop */
  gdk_threads_enter();
  gtk_main();
  gdk_threads_leave();

  return(0);
}
```

5.3. Why does this strange 'x io error' occur when I `fork()` in my GTK+ app?

This is not really a GTK+ problem, and the problem is not related to `fork()` either. If the 'x io error' occurs then you probably use the `exit()` function in order to exit from the child process.

When GDK opens an X display, it creates a socket file descriptor. When you use the `exit()` function, you implicitly close all the open file descriptors, and the underlying X library really doesn't like this.

The right function to use here is `_exit()`.

Erik Mouw contributed the following code example to illustrate handling fork() and exit().

```
/*-------------------------------------------------------------------------
 * Filename:      gtk-fork.c
 * Version:       0.99.1
 * Copyright:     Copyright (C) 1999, Erik Mouw
 * Author:        Erik Mouw <J.A.K.Mouw@its.tudelft.nl>
 * Description:   GTK+ fork example
 * Created at:    Thu Sep 23 21:37:55 1999
 * Modified by:   Erik Mouw <J.A.K.Mouw@its.tudelft.nl>
 * Modified at:   Thu Sep 23 22:39:39 1999
 *-----------------------------------------------------------------------*/
/*
 * Compile with:
 *
 * cc -o gtk-fork gtk-fork.c 'gtk-config --cflags --libs'
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
```

```
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <gtk/gtk.h>

void sigchld_handler(int num)
{
  sigset_t set, oldset;
  pid_t pid;
  int status, exitstatus;

  /* block other incoming SIGCHLD signals */
  sigemptyset(&set);
  sigaddset(&set, SIGCHLD);
  sigprocmask(SIG_BLOCK, &set, &oldset);

  /* wait for child */
  while((pid = waitpid((pid_t)-1, &status, WNOHANG)) > 0)
    {
      if(WIFEXITED(status))
{
  exitstatus = WEXITSTATUS(status);

  fprintf(stderr,
  "Parent: child exited, pid = %d, exit status = %d\n",
  (int)pid, exitstatus);
}
      else if(WIFSIGNALED(status))
{
  exitstatus = WTERMSIG(status);

  fprintf(stderr,
  "Parent: child terminated by signal %d, pid = %d\n",
  exitstatus, (int)pid);
}
      else if(WIFSTOPPED(status))
{
  exitstatus = WSTOPSIG(status);

  fprintf(stderr,
  "Parent: child stopped by signal %d, pid = %d\n",
  exitstatus, (int)pid);
}
      else
{
  fprintf(stderr,
  "Parent: child exited magically, pid = %d\n",
  (int)pid);
}
    }

  /* re-install the signal handler (some systems need this) */
  signal(SIGCHLD, sigchld_handler);
```

```
  /* and unblock it */
  sigemptyset(&set);
  sigaddset(&set, SIGCHLD);
  sigprocmask(SIG_UNBLOCK, &set, &oldset);
}

gint delete_event(GtkWidget *widget, GdkEvent *event, gpointer data)
{
  return(FALSE);
}

void destroy(GtkWidget *widget, gpointer data)
{
  gtk_main_quit();
}

void fork_me(GtkWidget *widget, gpointer data)
{
  pid_t pid;

  pid = fork();

  if(pid == -1)
    {
      /* ouch, fork() failed */
      perror("fork");
      exit(-1);
    }
  else if(pid == 0)
    {
      /* child */
      fprintf(stderr, "Child: pid = %d\n", (int)getpid());

      execlp("ls", "ls", "-CF", "/", NULL);

      /* if exec() returns, there is something wrong */
      perror("execlp");

      /* exit child. note the use of _exit() instead of exit() */
      _exit(-1);
    }
  else
    {
      /* parent */
      fprintf(stderr, "Parent: forked a child with pid = %d\n", (int)pid);
    }
}

int main(int argc, char *argv[])
{
  GtkWidget *window;
  GtkWidget *button;
```

```
  gtk_init(&argc, &argv);

  /* the basic stuff: make a window and set callbacks for destroy and
   * delete events
   */
  window = gtk_window_new(GTK_WINDOW_TOPLEVEL);

  gtk_signal_connect(GTK_OBJECT (window), "delete_event",
     GTK_SIGNAL_FUNC(delete_event), NULL);

  gtk_signal_connect(GTK_OBJECT (window), "destroy",
     GTK_SIGNAL_FUNC(destroy), NULL);

#if (GTK_MAJOR_VERSION == 1) && (GTK_MINOR_VERSION == 0)
  gtk_container_border_width(GTK_CONTAINER (window), 10);
#else
  gtk_container_set_border_width(GTK_CONTAINER (window), 10);
#endif

  /* add a button to do something usefull */
  button = gtk_button_new_with_label("Fork me!");

  gtk_signal_connect(GTK_OBJECT (button), "clicked",
     GTK_SIGNAL_FUNC(fork_me), NULL);

  gtk_container_add(GTK_CONTAINER(window), button);

  /* show everything */
  gtk_widget_show (button);
  gtk_widget_show (window);


  /* install a signal handler for SIGCHLD signals */
  signal(SIGCHLD, sigchld_handler);


  /* main loop */
  gtk_main ();

  exit(0);
}
```

5.4. Why don’t the contents of a button move when the button is pressed? Here’s a patch to make it work that way...

From: Peter Mattis

"The reason buttons don't move their child down and to the right when they are depressed is because I don't think that's what is happening visually. My view of buttons is that you are looking at them straight on. That is, the user interface lies in a plane and you're above it looking straight at it. When a button gets pressed it moves directly away from you. To be absolutely correct I guess the child should actually shrink a tiny amount. But I don't see why the child should shift down and to the left. Remember, the child is supposed to be attached to the buttons surface. Its not good for it to appear like the child is slipping on the surface of the button."

"On a more practical note, I did implement this at one point and determined it didn't look good and removed it."

5.5. How do I identifiy a widgets top level window or other ancestor?

There are a couple of ways to find the top level parent of a widget. The easier way is to call the `gtk_widget_top_level()` function that returns pointer to a GtkWidget that is the top level window.

A more complicated way to do this (but less limited, as it allows the user to get the closest ancestor of a known type) is to use `gtk_widget_get_ancestor()` as in:

```
GtkWidget       *widget;
widget = gtk_widget_get_ancestor(w, GTK_TYPE_WINDOW);
```

Since virtually all the GTK_TYPEs can be used as the second parameter of this function, you can get any parent widget of a particular widget. Suppose you have an hbox which contains a vbox, which in turn contains some other atomic widget (entry, label, etc. To find the master hbox using the `entry` widget simply use:

```
GtkWidget       *hbox;
hbox = gtk_widget_get_ancestor(w, GTK_TYPE_HBOX);
```

5.6. How do I get the Window ID of a GtkWindow?

The actual Gdk/X window will be created when the widget gets realized. You can get the Window ID with:

```
#include <gdk/gdkx.h>

Window xwin = GDK_WINDOW_XWINDOW (GTK_WIDGET (my_window)->window);
```

5.7. How do I catch a double click event (in a list widget, for example)?

Tim Janik wrote to gtk-list (slightly modified):

Define a signal handler:

```
gint
signal_handler_event(GtkWidget *widget, GdkEventButton *event, gpointer func_data)
{
  if (GTK_IS_LIST_ITEM(widget) &&
       (event->type==GDK_2BUTTON_PRESS ||
        event->type==GDK_3BUTTON_PRESS) ) {
    printf("I feel %s clicked on button %d\n",
           event->type==GDK_2BUTTON_PRESS ? "double" : "triple",
           event->button);
  }

  return FALSE;
}
```

And connect the handler to your object:

```
{
  /* list, list item init stuff */

  gtk_signal_connect(GTK_OBJECT(list_item),
                     "button_press_event",
                     GTK_SIGNAL_FUNC(signal_handler_event),
                     NULL);

  /* and/or */

  gtk_signal_connect(GTK_OBJECT(list_item),
                     "button_release_event",
                     GTK_SIGNAL_FUNC(signal_handler_event),
                     NULL);

  /* something else */
}
```

and, Owen Taylor wrote:

“Note that a single button press will be received beforehand, and if you are doing this for a button, you will therefore also get a "clicked" signal for the button. (This is going to be true for any toolkit, since computers aren’t good at reading one’s mind.)”

5.8. By the way, what are the differences between signals and events?

First of all, Havoc Pennington gives a rather complete description of the differences between events and signals in his free book (two chapters can be found at http://www106.pair.com/rhp/sample_chapters.html).

Moreover, Havoc posted this to the `gtk-list` “Events are a stream of messages received from the X server. They drive the Gtk main loop; which more or less amounts to "wait for events, process them" (not exactly, it is really more general than that and can wait on many different input streams at once). Events are a Gdk/Xlib concept.”

“Signals are a feature of GtkObject and its subclasses. They have nothing to do with any input stream; really a signal is just a way to keep a list of callbacks around and invoke them ("emit" the signal). There are lots of details and extra features of course. Signals are emitted by object instances, and are entirely unrelated to the Gtk main loop. Conventionally, signals are emitted "when something changes" about the object emitting the signal.”

“Signals and events only come together because GtkWidget happens to emit signals when it gets events. This is purely a convenience, so you can connect callbacks to be invoked when a particular widget receives a particular event. There is nothing about this that makes signals and events inherently related concepts, any more than emitting a signal when you click a button makes button clicking and signals related concepts.”

5.9. Data I pass to the `delete_event` (or other event) handler gets corrupted.

All event handlers take an additional argument which contains information about the event that triggered the handler. So, a `delete_event` handler must be declared as:

```
gint delete_event_handler (GtkWidget   *widget,
                           GdkEventAny *event,
                           gpointer     data);
```

5.10. I have my signal connected to the the (whatever) event, but it seems I don’t catch it. What’s wrong?

There is some special initialisation to do in order to catch some particular events. In fact, you must set the correct event mask bit of your widget before getting some particular events.

For example,

```
  gtk_widget_add_events(window, GDK_KEY_RELEASE_MASK);
```

lets you catch the key release events. If you want to catch every events, simply us the GDK_ALL_EVENTS_MASK event mask.

All the event masks are defined in the `gdktypes.h` file.

5.11. I need to add a new signal to a GTK+ widget. Any idea?

If the signal you want to add may be beneficial for other GTK+ users, you may want to submit a patch that presents your changes. Check the tutorial for more information about adding signals to a widget class.

If you don't think it is the case or if your patch is not applied you'll have to use the `gtk_object_class_user_signal_new` function. `gtk_object_class_user_signal_new` allows you to add a new signal to a predefined GTK+ widget without any modification of the GTK+ source code. The new signal can be emited with `gtk_signal_emit` and can be handled in the same way as other signals.

Tim Janik posted this code snippet:

```
static guint signal_user_action = 0;

signal_user_action =
  gtk_object_class_user_signal_new (gtk_type_class (GTK_TYPE_WIDGET),
                    "user_action",
                    GTK_RUN_LAST | GTK_RUN_ACTION,
                    gtk_marshal_NONE__POINTER,
                    GTK_TYPE_NONE, 1,
                    GTK_TYPE_POINTER);

void
gtk_widget_user_action (GtkWidget *widget,
                        gpointer   act_data)
{
  g_return_if_fail (GTK_IS_WIDGET (widget));

  gtk_signal_emit (GTK_OBJECT (widget), signal_user_action, act_data);
}
```

If you want your new signal to have more than the classical gpointer parameter, you'll have to play with GTK+ marshallers.

5.12. Is it possible to get some text displayed which is truncated to fit inside its allocation?

GTK's behavior (no clipping) is a consequence of its attempts to conserve X resources. Label widgets (among others) don't get their own X window - they just draw their contents on their parent's window. While it might be possible to have clipping occur by setting the clip mask before drawing the text, this would probably cause a substantial performance penalty.

Its possible that, in the long term, the best solution to such problems might be just to change gtk to give labels X windows. A short term workaround is to put the label widget inside another widget that does get its own window - one possible candidate would be the viewport widget.

```
viewport = gtk_viewport (NULL, NULL);
gtk_widget_set_usize (viewport, 50, 25);
gtk_viewport_set_shadow_type (GTK_VIEWPORT(viewport), GTK_SHADOW_NONE);
gtk_widget_show(viewport);

label = gtk_label ("a really long label that won't fit");
gtk_container_add (GTK_CONTAINER(viewport), label);
gtk_widget_show (label);
```

If you were doing this for a bunch of widgets, you might want to copy gtkviewport.c and strip out the adjustment and shadow functionality (perhaps you could call it GtkClipper).

5.13. How do I make my window modal? / How do I make a single window active?

After you create your window, do `gtk_grab_add(my_window)`. And after closing the window do `gtk_grab_remove(my_window)`.

5.14. Why doesn't my widget (e.g. progressbar) update?

You are probably doing all the changes within a function without returning control to `gtk_main()`. This may be the case if you do some lengthy calculation in your code. Most drawing updates are only placed on a queue, which is processed within `gtk_main()`. You can force the drawing queue to be processed using something like:

```
while (g_main_iteration(FALSE));
```

inside you're function that changes the widget.

What the above snippet does is run all pending events and high priority idle functions, then return immediately (the drawing is done in a high priority idle function).

5.15. How do I attach data to some GTK+ object/widget?

First of all, the attached data is stored in the object_data field of a GtkObject. The type of this field is GData, which is defined in glib.h. So you should read the gdataset.c file in your glib source directory very carefully.

There are two (easy) ways to attach some data to a gtk object. Using `gtk_object_set_data()` and `gtk_object_get_data()` seems to be the most common way to do this, as it provides a powerful interface to connect objects and data.

```
void gtk_object_set_data(GtkObject *object, const gchar *key, gpointer data);

gpointer gtk_object_get_data(GtkObject *object, const gchar *key);
```

Since a short example is better than any lengthy speech:

```
struct my_struct p1,p2,*result;
GtkWidget *w;

gtk_object_set_data(GTK_OBJECT(w),"p1 data",(gpointer)&p1);
gtk_object_set_data(GTK_OBJECT(w),"p2 data",(gpointer)&p2);

result = gtk_object_get_data(GTK_OBJECT(w),"p1 data");
```

The `gtk_object_set_user_data()` and `gtk_object_get_user_data()` functions does exactly the same thing as the functions above, but does not let you specify the "key" parameter.Instead, it uses a standard "user_data" key. Note that the use of these functions is deprecated in 1.2. They only provide a compatibility mode with some old gtk packages.

5.16. How do I remove the data I have attached to an object?

When attaching the data to the object, you can use the `gtk_object_set_data_full()` function. The three first arguments of the function are the same as in `gtk_object_set_data()`. The fourth one is a pointer to a callback function which is called when the data is destroyed. The data is destroyed when you:

- destroy the object
- replace the data with a new one (with the same key)
- replace the data with NULL (with the same key)

5.17. How do I reparent a widget?

The normal way to reparent (ie change the owner) of a widget should be to use the function:

```
void gtk_widget_reparent (GtkWidget *widget,
                          GtkWidget *new_parent)
```

But this is only a "should be" since this function does not correctly do its job on some specific widgets. The main goal of gtk_widget_reparent() is to avoid unrealizing widget if both widget and new_parent are realized (in this case, widget->window is successfully reparented). The problem here is that some widgets in the GTK+ hierarchy have multiple attached X subwindows and this is notably the case for the GtkSpinButton widget. For those, gtk_widget_reparent() will fail by leaving an unrealized child window where it should not.

To avoid this problem, simply use the following code snippet:

```
gtk_widget_ref(widget);
gtk_container_remove(GTK_CONTAINER(old_parent), widget);
gtk_container_add(GTK_CONTAINER(new_parent), widget);
gtk_widget_unref(widget);
```

5.18. How could I get any widgets position?

As Tim Janik pointed out, there are different cases, and each case requires a different solution.

- If you want the position of a widget relative to its parent, you should use `widget->allocation.x` and `widget->allocation.y`.
- If you want the position of a window relative to the X root window, you should use `gdk_window_get_geometry()` `gdk_window_get_position()` or `gdk_window_get_origin()`.
- If you want to get the position of the window (including the WM decorations), you should use `gdk_window_get_root_origin()`.
- Last but not least, if you want to get a Window Manager frame position, you should use `gdk_window_get_deskrelative_origin()`.

Your choice of Window Manager will have an effect of the results of the above functions. You should keep this in mind when writing your application. This is dependant upon how the Window Managers manage the decorations that they add around windows.

5.19. How do I set the size of a widget/window? How do I prevent the user resizing my window?

The `gtk_widget_set_uposition()` function is used to set the position of any widget.

The `gtk_widget_set_usize()` function is used to set the size of a widget. In order to use all the features that are provided by this function when it acts on a window, you may want to use the `gtk_window_set_policy` function. The definition of these functions are:

```
void gtk_widget_set_usize (GtkWidget *widget,
                           gint width,
                           gint height);

void gtk_window_set_policy (GtkWindow *window,
                            gint allow_shrink,
                            gint allow_grow,
                            gint auto_shrink);
```

`Auto_shrink` will automatically shrink the window when the requested size of the child widgets goes below the current size of the window. `Allow_shrink` will give the user the authorisation to make the window smaller that it should normally be. `Allow_grow` will give the user will have the ability to make the window bigger. The default values for these parameters are:

```
allow_shrink = FALSE
allow_grow   = TRUE
auto_shrink  = FALSE
```

The `gtk_widget_set_usize()` functions is not the easiest way to set a window size since you cannot decrease this window size with another call to this function unless you call it twice, as in:

```
gtk_widget_set_usize(your_widget, -1, -1);
gtk_widget_set_usize(your_widget, new_x_size, new_y_size);
```

Another way to set the size of and/or move a window is to use the `gdk_window_move_resize()` function which uses to work fine both to grow or to shrink the window:

```
gdk_window_move_resize(window->window,
                       x_pos, y_pos,
                       x_size, y_size);
```

5.20. How do I add a popup menu to my GTK+ application?

The `menu` example in the examples/menu directory of the GTK+ distribution implements a popup menu with this technique:

```
static gint button_press (GtkWidget *widget, GdkEvent *event)
{

    if (event->type == GDK_BUTTON_PRESS) {
        GdkEventButton *bevent = (GdkEventButton *) event;
        gtk_menu_popup (GTK_MENU(widget), NULL, NULL, NULL, NULL,
                        bevent->button, bevent->time);
        /* Tell calling code that we have handled this event; the buck
         * stops here. */
        return TRUE;
    }

    /* Tell calling code that we have not handled this event; pass it on. */
    return FALSE;
}
```

5.21. How do I disable or enable a widget, such as a button?

To disable (or to enable) a widget, use the `gtk_widget_set_sensitive()` function. The first parameter is you widget pointer. The second parameter is a boolean value: when this value is TRUE, the widget is enabled.

5.22. Shouldn’t the text argument in the gtk_clist_* functions be declared const?

For example:

```
gint gtk_clist_prepend (GtkCList *clist,
                        gchar    *text[]);
```

Answer: No, while a type "gchar*" (pointer to char) can automatically be cast into "const gchar*" (pointer to const char), this does not apply for "gchar *[]" (array of an unspecified number of pointers to char) into "const gchar *[]" (array of an unspecified number of pointers to const char).

The type qualifier "const" may be subject to automatic casting, but in the array case, it is not the array itself that needs the (const) qualified cast, but its members, thus changing the whole type.

5.23. How do I render pixels (image data) to the screen?

There are several ways to approach this. The simplest way is to use GdkRGB, see gdk/gdkrgb.h. You create an RGB buffer, render to your RGB buffer, then use GdkRGB routines to copy your RGB buffer to

a drawing area or custom widget. The book "GTK+/Gnome Application Development" gives some details; GdkRGB is also documented in the GTK+ reference documentation.

If you're writing a game or other graphics-intensive application, you might consider a more elaborate solution. OpenGL is the graphics standard that will let you access hardware accelaration in future versions of XFree86; so for maximum speed, you probably want to use OpenGL. A GtkGLArea widget is available for using OpenGL with GTK+ (but GtkGLArea does not come with GTK+ itself). There are also several open source game libraries, such as ClanLib and Loki's Simple DirectMedia Layer library (SDL).

You do NOT want to use `gdk_draw_point()`, that will be extremely slow.

5.24. How do I create a pixmap without having my window being realized/shown?

Functions such as `gdk_pixmap_create_from_xpm()` require a valid window as a parameter. During the initialisation phase of an application, a valid window may not be available without showing a window, which may be inappropriate. In order to avoid this, a function such as `gdk_pixmap_colormap_create_from_xpm` can be used, as in:

```
char *pixfile = "foo.xpm";
GtkWidget *top, *box, *pixw;
GdkPixmap *pixmap, *pixmap_mask;

top = gtk_window_new (GKT_WINDOW_TOPLEVEL);
box = gtk_hbox_new (FALSE, 4);
gtk_conainer_add (GTK_CONTAINER(top), box);

pixmap = gdk_pixmap_colormap_create_from_xpm (
             NULL, gtk_widget_get_colormap(top),
             &pixmap_mask, NULL, pixfile);
pixw = gtk_pixmap_new (pixmap, pixmap_mask);
gdk_pixmap_unref (pixmap);
gdk_pixmap_unref (pixmap_mask);
```

5.25. How do I do drag-and-drop?

GTK+ has a high level set of functions for doing inter-process communication via the drag-and-drop system. GTK+ can perform drag-and-drop on top of the low level Xdnd and Motif drag-and-drop protocols.

The documentation on GTK+ drag-and-drop isn't complete, but there is some information in the Tutorial (http://www.gtk.org/tutorial/). You should also look at the drag-and-drop example code that is part of the GTK+ source distribution, in the file `gtk/testdnd.c`.

5.26. Why does GTK+/GLib leak memory?

It doesn't. Both GLib and the C library (malloc implementation) will cache allocated memory on occasion, even if you free it with free().

So you can't generally use tools such as top to see if you are using free() properly (aside from the very roughest of estimations, i.e. if you are really, really screwing up top will show that, but you can't distinguish small mistakes from the GLib/malloc caches).

In order to find memory leaks, use proper memory profiling tools.

Chapter 6. Development with GTK+: widget specific questions

6.1. How do I find out about the selection of a GtkList?

Get the selection something like this:

```
GList *sel;
sel = GTK_LIST(list)->selection;
```

This is how GList is defined (quoting glist.h):

```
typedef struct _GList GList;

struct _GList
{
  gpointer data;
  GList *next;
  GList *prev;
};
```

A GList structure is just a simple structure for doubly linked lists. there exist several g_list_*() functions to modify a linked list in glib.h. However the GTK_LIST(MyGtkList)->selection is maintained by the gtk_list_*() functions and should not be modified.

The selection_mode of the GtkList determines the selection facilities of a GtkList and therefore the contents of GTK_LIST(AnyGtkList)->selection:

selection_mode	**GTK_LIST()->selection contents**
GTK_SELECTION_SINGLE	selection is either NULL or contains a GList* pointer for a single selected item.
GTK_SELECTION_BROWSE	selection is NULL if the list contains no widgets, otherwise it contains a GList* pointer for one GList structure.
GTK_SELECTION_MULTIPLE	selection is NULL if no listitems are selected or a a GList* pointer for the first selected item. that in turn points to a GList structure for the second selected item and so on.
GTK_SELECTION_EXTENDED	selection is NULL.

The data field of the GList structure GTK_LIST(MyGtkList)->selection points to the first GtkListItem that is selected. So if you would like to determine which listitems are selected you should go like this:

```
{
        gchar           *list_items[]={
                                "Item0",
                                "Item1",
                                "foo",
                                "last Item",
                        };
        guint           nlist_items=sizeof(list_items)/sizeof(list_items[0]);
        GtkWidget       *list_item;
        guint           i;

        list=gtk_list_new();
        gtk_list_set_selection_mode(GTK_LIST(list), GTK_SELECTION_MULTIPLE);
        gtk_container_add(GTK_CONTAINER(AnyGtkContainer), list);
        gtk_widget_show (list);

        for (i = 0; i < nlist_items; i++)
        {
                list_item=gtk_list_item_new_with_label(list_items[i]);
                gtk_object_set_user_data(GTK_OBJECT(list_item), (gpointer)i);
                gtk_container_add(GTK_CONTAINER(list), list_item);
                gtk_widget_show(list_item);
        }
}
```

To get known about the selection:

```
{
        GList   *items;

        items=GTK_LIST(list)->selection;

        printf("Selected Items: ");
        while (items) {
                if (GTK_IS_LIST_ITEM(items->data))
                        printf("%d ", (guint)
                gtk_object_get_user_data(items->data));
                items=items->next;
        }
        printf("\n");
}
```

6.2. How do I stop the column headings of a GtkCList disappearing when the list is scrolled?

This happens when a GtkCList is packed into a GtkScrolledWindow using the function `gtk_scroll_window_add_with_viewport()`. The prefered method of adding a CList to a scrolled window is to use the function `gtk_container_add`, as in:

```
GtkWidget *scrolled, *clist;
char *titles[] = { "Title1" , "Title2" };

scrolled = gtk_scrolled_window_new(NULL, NULL);

clist = gtk_clist_new_with_titles(2, titles);
gtk_container_add(GTK_CONTAINER(scrolled), clist);
```

6.3. I don’t want the user of my applications to enter text into a GtkCombo. Any idea?

A GtkCombo has an associated entry which can be accessed using the following expression:

```
GTK_COMBO(combo_widget)->entry
```

If you don’t want the user to be able to modify the content of this entry, you can use the gtk_entry_set_editable() function:

```
void gtk_entry_set_editable(GtkEntry *entry,
                            gboolean editable);
```

Set the editable parameter to FALSE to disable typing into the entry.

6.4. How do I catch a combo box change?

The entry which is associated to your GtkCombo send a "changed" signal when:

- some text is typed in
- the selection of the combo box is changed

To catch any combo box change, simply connect your signal handler with

```
gtk_signal_connect(GTK_COMBO(cb)->entry,
                   "changed",
                   GTK_SIGNAL_FUNC(my_cb_change_handler),
                   NULL);
```

6.5. How can I define a separation line in a menu?

See the Tutorial (http://www.gtk.org/tutorial/) for information on how to create menus. However, to create a separation line in a menu, just insert an empty menu item:

```
menuitem = gtk_menu_item_new();
gtk_menu_append(GTK_MENU(menu), menuitem);
gtk_widget_show(menuitem);
```

6.6. How can I right justify a menu, such as Help?

Depending on if you use the MenuFactory or not, there are two ways to proceed. With the MenuFactory, use something like the following:

```
menu_path = gtk_menu_factory_find (factory,  "<MyApp>/Help");
gtk_menu_item_right_justify(menu_path->widget);
```

If you do not use the MenuFactory, you should simply use:

```
gtk_menu_item_right_justify(my_menu_item);
```

6.7. How do I add some underlined accelerators to menu items?

Damon Chaplin, the technical force behind the Glade project, provided the following code sample (this code is an output from Glade). It creates a small File menu item with only one child (New). The F in File and the N in New are underlined, and the relevant accelerators are created.

```
menubar1 = gtk_menu_bar_new ();
gtk_object_set_data (GTK_OBJECT (window1), "menubar1", menubar1);
gtk_widget_show (menubar1);
gtk_box_pack_start (GTK_BOX (vbox1), menubar1, FALSE, FALSE, 0);

file1 = gtk_menu_item_new_with_label ("");
tmp_key = gtk_label_parse_uline (GTK_LABEL (GTK_BIN (file1)->child),
                                 _("_File"));
gtk_widget_add_accelerator (file1, "activate_item", accel_group,
                            tmp_key, GDK_MOD1_MASK, 0);
gtk_object_set_data (GTK_OBJECT (window1), "file1", file1);
gtk_widget_show (file1);
gtk_container_add (GTK_CONTAINER (menubar1), file1);

file1_menu = gtk_menu_new ();
file1_menu_accels = gtk_menu_ensure_uline_accel_group (GTK_MENU (file1_menu));
gtk_object_set_data (GTK_OBJECT (window1), "file1_menu", file1_menu);
gtk_menu_item_set_submenu (GTK_MENU_ITEM (file1), file1_menu);

new1 = gtk_menu_item_new_with_label ("");
tmp_key = gtk_label_parse_uline (GTK_LABEL (GTK_BIN (new1)->child),
                                 _("_New"));
gtk_widget_add_accelerator (new1, "activate_item", file1_menu_accels,
                            tmp_key, 0, 0);
```

```
gtk_object_set_data (GTK_OBJECT (window1), "new1", new1);
gtk_widget_show (new1);
gtk_container_add (GTK_CONTAINER (file1_menu), new1);
```

6.8. How can I retrieve the text from a GtkMenuItem?

You can usually retrieve the label of a specific GtkMenuItem with:

```
if (GTK_BIN (menu_item)->child)
{
  GtkWidget *child = GTK_BIN (menu_item)->child;

  /* do stuff with child */
  if (GTK_IS_LABEL (child))
  {
    gchar *text;

    gtk_label_get (GTK_LABEL (child), &text);
    g_print ("menu item text: %s\n", text);
  }
}
```

To get the active menu item from a GtkOptionMenu you can do:

```
if (GTK_OPTION_MENU (option_menu)->menu_item)
{
  GtkWidget *menu_item = GTK_OPTION_MENU (option_menu)->menu_item;
}
```

But, there's a catch. For this specific case, you can *not* get the label widget from `menu_item` with the above code, because the option menu reparents the menu_item's child temporarily to display the currently active contents. So to retrive the child of the currently active menu_item of an option menu, you'll have to do:

```
if (GTK_BIN (option_menu)->child)
{
  GtkWidget *child = GTK_BIN (option_menu)->child;

  /* do stuff with child */
}
```

6.9. How do I right (or otherwise) justify a GtkLabel?

Are you sure you want to *justify* the labels? The label class contains the `gtk_label_set_justify()` function that is used to control the justification of a multi-line label.

What you probably want is to set the *alignment* of the label, ie right align it, center it or left align it. If you want to do this, you should use:

```
void gtk_misc_set_alignment (GtkMisc *misc,
                             gfloat xalign,
                             gfloat yalign);
```

where the `xalign` and `yalign` values are floats in [0.00;1.00].

```
GtkWidget       *label;

/* horizontal : left align, vertical : top */
gtk_misc_set_alignment(GTK_MISC(label), 0.0f, 0.0f);

/* horizontal : centered, vertical : centered */
gtk_misc_set_alignment(GTK_MISC(label), 0.5f, 0.5f);

/* horizontal : right align, vertical : bottom */
gtk_misc_set_alignment(GTK_MISC(label), 1.0f, 1.0f);
```

6.10. How do I set the background color of a GtkLabel widget?

The Gtklabel widget is one of a few GTK+ widgets that don't create their own window to render themselves into. Instead, they draw themselves directly onto their parents window.

This means that in order to set the background color for a GtkLabel widget, you need to change the background color of its parent, i.e. the object that you pack it into.

6.11. How do I set the color and font of a GtkLabel using a Resource File?

The widget name path constructed for a Label consists of the widget names of its object hierarchy as well, e.g.

```
window (name: humphrey)
  hbox
     label (name: mylabel)
```

The widget path your pattern needs to match would be: `humphrey.GtkHBox.mylabel`

The resource file may look something like:

```
style "title"
{
      fg[NORMAL] = {1.0, 0.0, 0.0}
      font = "-adobe-helvetica-bold-r-normal--*-140-*-*-*-*-*-*"
}
widget "*mylabel" style "title"
```

In your program, you would also need to give a name to the Label widget, which can be done using:

```
  label = gtk_label_new("Some Label Text");
  gtk_widget_set_name(label, "mylabel");
  gtk_widget_show(label);
```

6.12. How do I configure Tooltips in a Resource File?

The tooltip's window is named "gtk-tooltips", GtkTooltips in itself is not a GtkWidget (though a GtkObject) and as such is not attempted to match any widget styles.

So, you resource file should look something like:

```
style "postie"
{
      bg[NORMAL] = {1.0, 1.0, 0.0}
}
widget "gtk-tooltips*" style "postie"
```

6.13. I can't add more than (something like) 2000 chars in a GtkEntry. What's wrong?

There is now a known problem in the GtkEntry widget. In the `gtk_entry_insert_text()` function, the following lines limit the number of chars in the entry to 2047.

```
  /* The algorithms here will work as long as, the text size (a
   * multiple of 2), fits into a guint16 but we specify a shorter
   * maximum length so that if the user pastes a very long text, there
   * is not a long hang from the slow X_LOCALE functions.  */

  if (entry->text_max_length == 0)
    max_length = 2047;
  else
    max_length = MIN (2047, entry->text_max_length);
```

6.14. How do I make a GtkEntry widget activate on pressing the Return key?

The Entry widget emits an 'activate' signal when you press return in it. Just attach to the activate signal on the entry and do whatever you want to do. Typical code would be:

```
  entry = gtk_entry_new();
  gtk_signal_connect (GTK_OBJECT(entry), "activate",
                      GTK_SIGNAL_FUNC(entry_callback),
                      NULL);
```

6.15. How do I validate/limit/filter the input to a GtkEntry?

If you want to validate the text that a user enters into a GtkEntry widget you can attach to the "insert_text" signal of the entry, and modify the text within the callback function. The example below forces all characters to uppercase, and limits the range of characters to A-Z. Note that the entry is cast to an object of type GtkEditable, from which GtkEntry is derived.

```
#include <ctype.h>
#include <gtk/gtk.h>

void insert_text_handler (GtkEntry    *entry,
                          const gchar *text,
                          gint         length,
                          gint        *position,
                          gpointer     data)
{
  GtkEditable *editable = GTK_EDITABLE(entry);
  int i, count=0;
  gchar *result = g_new (gchar, length);

  for (i=0; i < length; i++) {
    if (!isalpha(text[i]))
      continue;
    result[count++] = islower(text[i]) ? toupper(text[i]) : text[i];
  }

  if (count > 0) {
    gtk_signal_handler_block_by_func (GTK_OBJECT (editable),
      GTK_SIGNAL_FUNC (insert_text_handler),
      data);
    gtk_editable_insert_text (editable, result, count, position);
    gtk_signal_handler_unblock_by_func (GTK_OBJECT (editable),
GTK_SIGNAL_FUNC (insert_text_handler),
data);
  }
  gtk_signal_emit_stop_by_name (GTK_OBJECT (editable), "insert_text");
```

```
  g_free (result);
}

int main (int   argc,
          char *argv[])
{
  GtkWidget *window;
  GtkWidget *entry;

  gtk_init (&argc, &argv);

  /* create a new window */
  window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  gtk_window_set_title(GTK_WINDOW (window), "GTK Entry");
  gtk_signal_connect(GTK_OBJECT (window), "delete_event",
     (GtkSignalFunc) gtk_exit, NULL);

  entry = gtk_entry_new();
  gtk_signal_connect(GTK_OBJECT(entry), "insert_text",
     GTK_SIGNAL_FUNC(insert_text_handler),
     NULL);
  gtk_container_add(GTK_CONTAINER (window), entry);
  gtk_widget_show(entry);

  gtk_widget_show(window);

  gtk_main();
  return(0);
}
```

6.16. How do I use horizontal scrollbars with a GtkText widget?

The short answer is that you can’t. The current version of the GtkText widget does not support horizontal scrolling. There is an intention to completely rewrite the GtkText widget, at which time this limitation will be removed.

6.17. How do I change the font of a GtkText widget?

There are a couple of ways of doing this. As GTK+ allows the appearance of applications to be changed at run time using resources you can use something like the following in the appropriate file:

```
style "text"
{
  font = "-adobe-helvetica-medium-r-normal--*-100-*-*-*-*-*-*"
}
```

Another way to do this is to load a font within your program, and then use this in the functions for adding text to the text widget. You can load a font using, for example:

```
GdkFont *font;
font = gdk_font_load("-adobe-helvetica-medium-r-normal--*-140-*-*-*-*-*-*");
```

6.18. How do I set the cursor position in a GtkText object?

Notice that the response is valid for any object that inherits from the GtkEditable class.

Are you sure that you want to move the cursor position? Most of the time, while the cursor position is good, the insertion point does not match the cursor position. If this apply to what you really want, then you should use the `gtk_text_set_point()` function. If you want to set the insertion point at the current cursor position, use the following:

```
gtk_text_set_point(GTK_TEXT(text),
gtk_editable_get_position(GTK_EDITABLE(text)));
```

If you want the insertion point to follow the cursor at all time, you should probably catch the button press event, and then move the insertion point. Be careful : you'll have to catch it after the widget has changed the cursor position though. Thomas Mailund Jensen proposed the following code:

```
static void
insert_bar (GtkWidget *text)
{
  /* jump to cursor mark */
  gtk_text_set_point (GTK_TEXT (text),
  gtk_editable_get_position (GTK_EDITABLE  (text)));

  gtk_text_insert (GTK_TEXT (text), NULL, NULL, NULL,
     "bar", strlen ("bar"));
}

int
main (int argc, char *argv[])
{
  GtkWidget *window, *text;

  gtk_init (&argc, &argv);

  window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  text = gtk_text_new (NULL, NULL);
  gtk_text_set_editable (GTK_TEXT (text), TRUE);
  gtk_container_add (GTK_CONTAINER (window), text);

  /* connect after everything else */
  gtk_signal_connect_after (GTK_OBJECT(text), "button_press_event",
    GTK_SIGNAL_FUNC (insert_bar), NULL);
```

```
  gtk_widget_show_all(window);
  gtk_main();

  return 0;
}
```

Now, if you really want to change the cursor position, you should use the `gtk_editable_set_position()` function.

Chapter 7. About GDK

7.1. What is GDK?

GDK is basically a wrapper around the standard Xlib function calls. If you are at all familiar with Xlib, a lot of the functions in GDK will require little or no getting used to. All functions are written to provide an way to access Xlib functions in an easier and slightly more intuitive manner. In addition, since GDK uses GLib (see below), it will be more portable and safer to use on multiple platforms.

7.2. How do I use color allocation?

One of the nice things about GDK is that it's based on top of Xlib; this is also a problem, especially in the area of color management. If you want to use color in your program (drawing a rectangle or such, your code should look something like this:

```
{
  GdkColor *color;
  int width, height;
  GtkWidget *widget;
  GdkGC *gc;

  ...

  /* first, create a GC to draw on */
  gc = gdk_gc_new(widget->window);

  /* find proper dimensions for rectangle */
  gdk_window_get_size(widget->window, &width, &height);

  /* the color we want to use */
  color = (GdkColor *)malloc(sizeof(GdkColor));

  /* red, green, and blue are passed values, indicating the RGB triple
   * of the color we want to draw. Note that the values of the RGB components
   * within the GdkColor are taken from 0 to 65535, not 0 to 255.
   */
  color->red = red * (65535/255);
  color->green = green * (65535/255);
  color->blue = blue * (65535/255);

  /* the pixel value indicates the index in the colormap of the color.
   * it is simply a combination of the RGB values we set earlier
   */
  color->pixel = (gulong)(red*65536 + green*256 + blue);

  /* However, the pixel valule is only truly valid on 24-bit (TrueColor)
   * displays. Therefore, this call is required so that GDK and X can
```

```
   * give us the closest color available in the colormap
   */
  gdk_color_alloc(gtk_widget_get_colormap(widget), color);

  /* set the foreground to our color */
  gdk_gc_set_foreground(gc, color);

  /* draw the rectangle */
  gdk_draw_rectangle(widget->window, gc, 1, 0, 0, width, height);

  ...
}
```

Chapter 8. About GLib

8.1. What is GLib?

GLib is a library of useful functions and definitions available for use when creating GDK and GTK applications. It provides replacements for some standard libc functions, such as malloc, which are buggy on some systems.

It also provides routines for handling:

- Doubly Linked Lists
- Singly Linked Lists
- Timers
- String Handling
- A Lexical Scanner
- Error Functions

8.2. How can I use the doubly linked lists?

The GList object is defined as:

```
typedef struct _GList GList;

struct _GList
{
  gpointer data;
  GList *next;
  GList *prev;
};
```

To use the GList objects, simply:

```
GList   *list = NULL;
GList   *listrunner;
gint    array[] = { 1, 2, 3, 4, 5, 6 };
gint    pos;
gint    *value;

/* add data to the list */
for (pos=0;pos < sizeof array; pos++) {
  list = g_list_append(list, (gpointer)&array[pos]);
}

/* run through the list */
listrunner = g_list_first(list);
while (listrunner) {
```

```
  value = (gint *)listrunner->data;
  printf("%d\n", *value);
  listrunner = g_list_next(listrunner);
}

/* removing datas from the list */
listrunner = g_list_first(list);
list = g_list_remove_link(list, listrunner);
list = g_list_remove(list, &array[4]);
```

The same code is usable with singly linked lists (GSList objects) by replacing g_list_* functions with the relevant g_slist_* ones (g_slist_append, g_slist_remove, ...). Just remember that since you can't go backward in a singly linked list, there is no g_slist_first function - you'll need to keep a reference on the first node of the list.

8.3. Memory does not seem to be released when I free the list nodes I've allocated

GLib tries to be "intelligent" on this special issue: it assumes that you are likely to reuse the objects, so caches the allocated memory. If you do not want to use this behavior, you'll probably want to set up a special allocator.

To quote Tim Janik:

"If you have a certain portion of code that uses *lots* of GLists or GNodes, and you know you'd better want to release all of them after a short while, you'd want to use a GAllocator. Pushing an allocator into g_list will make all subsequent glist operations private to that allocator's memory pool (and thus you have to take care to pop the allocator again, before making any external calls): "

```
GAllocator *allocator;
GList *list = NULL;
guint i;

/* set a new allocation pool for GList nodes */
allocator = g_allocator_new ("list heap", 1024);
g_list_push_allocator (allocator);

/* do some list operations */
for (i = 0; i < 4096; i++)
  list = g_list_prepend (list, NULL);
list = g_list_reverse (list);

/* beware to pop allocator befor calling external functions */
g_list_pop_allocator ();
gtk_label_set_text (GTK_LABEL (some_label), "some text");

/* and set our private glist pool again */
```

```
g_list_push_allocator (allocator);

/* do some list operations */
g_list_free (list);
list = NULL;
for (i = 0; i < 4096; i++)
  list = g_list_prepend (list, NULL);

/* and back out (while freeing all of the list nodes in our pool) */
g_list_pop_allocator ();
g_allocator_free (allocator);
```

8.4. Why use g_print, g_malloc, g_strdup and fellow glib functions?

Thanks to Tim Janik who wrote to gtk-list: (slightly modified)

“Regarding g_malloc(), g_free() and siblings, these functions are much safer than their libc equivalents. For example, g_free() just returns if called with NULL. Also, if USE_DMALLOC is defined, the definition for these functions changes (in glib.h) to use MALLOC(), FREE() etc... If MEM_PROFILE or MEM_CHECK are defined, there are even small statistics made counting the used block sizes (shown by g_mem_profile() / g_mem_check()).”

“Considering the fact that glib provides an interface for memory chunks to save space if you have lots of blocks that are always the same size and to mark them ALLOC_ONLY if needed, it is just straight forward to create a small saver (debug able) wrapper around the normal malloc/free stuff as well - just like gdk covers Xlib. ;)”

“Using g_error() and g_warning() inside of applications like the GIMP that fully rely on gtk even gives the opportunity to pop up a window showing the messages inside of a gtk window with your own handler (by using g_set_error_handler()) along the lines of `gtk_print()` (inside of gtkmain.c).”

8.5. What’s a GScanner and how do I use one?

A GScanner will tokenize your text, that is, it’ll return an integer for every word or number that appears in its input stream, following certain (customizable) rules to perform this translation. You still need to write the parsing functions on your own though.

Here’s a little test program supplied by Tim Janik that will parse

```
<SYMBOL> = <OPTIONAL-MINUS> <NUMBER> ;
```

constructs, while skipping "#\n" and "/**/" style comments.

```
#include <glib.h>

/* some test text to be fed into the scanner */
static const gchar *test_text =
( "ping = 5;\n"
  "/* slide in some \n"
  " * comments, just for the\n"
  " * fun of it \n"
  " */\n"
  "pong = -6; \n"
  "\n"
  "# the next value is a float\n"
  "zonk = 0.7;\n"
  "# redefine ping\n"
  "ping = - 0.5;\n" );

/* define enumeration values to be returned for specific symbols */
enum {
  SYMBOL_PING = G_TOKEN_LAST + 1,
  SYMBOL_PONG = G_TOKEN_LAST + 2,
  SYMBOL_ZONK = G_TOKEN_LAST + 3
};

/* symbol array */
static const struct {
  gchar *symbol_name;
  guint  symbol_token;
} symbols[] = {
  { "ping", SYMBOL_PING, },
  { "pong", SYMBOL_PONG, },
  { "zonk", SYMBOL_ZONK, },
  { NULL, 0, },
}, *symbol_p = symbols;

static gfloat ping = 0;
static gfloat pong = 0;
static gfloat zonk = 0;

static guint
parse_symbol (GScanner *scanner)
{
  guint symbol;
  gboolean negate = FALSE;

  /* expect a valid symbol */
  g_scanner_get_next_token (scanner);
  symbol = scanner->token;
  if (symbol < SYMBOL_PING ||
```

```
      symbol > SYMBOL_ZONK)
    return G_TOKEN_SYMBOL;

  /* expect '=' */
  g_scanner_get_next_token (scanner);
  if (scanner->token != '=')
    return '=';

  /* feature optional '-' */
  g_scanner_peek_next_token (scanner);
  if (scanner->next_token == '-')
    {
      g_scanner_get_next_token (scanner);
      negate = !negate;
    }

  /* expect a float (ints are converted to floats on the fly) */
  g_scanner_get_next_token (scanner);
  if (scanner->token != G_TOKEN_FLOAT)
    return G_TOKEN_FLOAT;

  /* make sure the next token is a ';' */
  if (g_scanner_peek_next_token (scanner) != ';')
    {
      /* not so, eat up the non-semicolon and error out */
      g_scanner_get_next_token (scanner);
      return ';';
    }

  /* assign value, eat the semicolon and exit successfully */
  switch (symbol)
    {
    case SYMBOL_PING:
      ping = negate ? - scanner->value.v_float : scanner->value.v_float;
      break;
    case SYMBOL_PONG:
      pong = negate ? - scanner->value.v_float : scanner->value.v_float;
      break;
    case SYMBOL_ZONK:
      zonk = negate ? - scanner->value.v_float : scanner->value.v_float;
      break;
    }
  g_scanner_get_next_token (scanner);

  return G_TOKEN_NONE;
}

int
main (int argc, char *argv[])
{
  GScanner *scanner;
  guint expected_token;
```

```
scanner = g_scanner_new (NULL);

/* adjust lexing behaviour to suit our needs
 */
/* convert non-floats (octal values, hex values...) to G_TOKEN_INT */
scanner->config->numbers_2_int = TRUE;
/* convert G_TOKEN_INT to G_TOKEN_FLOAT */
scanner->config->int_2_float = TRUE;
/* don't return G_TOKEN_SYMBOL, but the symbol's value */
scanner->config->symbol_2_token = TRUE;

/* load symbols into the scanner */
while (symbol_p->symbol_name)
  {
    g_scanner_add_symbol (scanner,
                          symbol_p->symbol_name,
                          GINT_TO_POINTER (symbol_p->symbol_token));
    symbol_p++;
  }

/* feed in the text */
g_scanner_input_text (scanner, test_text, strlen (test_text));

/* give the error handler an idea on how the input is named */
scanner->input_name = "test text";

/* scanning loop, we parse the input until its end is reached,
 * the scanner encountered a lexing error, or our sub routine came
 * across invalid syntax
 */
do
  {
    expected_token = parse_symbol (scanner);

    g_scanner_peek_next_token (scanner);
  }
while (expected_token == G_TOKEN_NONE &&
       scanner->next_token != G_TOKEN_EOF &&
       scanner->next_token != G_TOKEN_ERROR);

/* give an error message upon syntax errors */
if (expected_token != G_TOKEN_NONE)
  g_scanner_unexp_token (scanner, expected_token, NULL, "symbol", NULL, NULL, TRUE);

/* finsish parsing */
g_scanner_destroy (scanner);

/* print results */
g_print ("ping: %f\n", ping);
g_print ("pong: %f\n", pong);
g_print ("zonk: %f\n", zonk);

return 0;
```

```
}
```

You need to understand that the scanner will parse its input and tokenize it, it is up to you to interpret these tokens, not define their types before they get parsed, e.g. watch gscanner parse a string:

```
"hi i am 17"
 |  | |  |
 |  | |  v
 |  | v  TOKEN_INT, value: 17
 |  v TOKEN_IDENTIFIER, value: "am"
 v  TOKEN_CHAR, value: 'i'
TOKEN_IDENTIFIER, value: "hi"
```

If you configure the scanner with:

```
scanner->config->int_2_float = TRUE;
scanner->config->char_2_token = TRUE;
scanner->config->scan_symbols = TRUE;
```

and add "am" as a symbol with

```
g_scanner_add_symbol (scanner, "am", "symbol value");
```

GScanner will parse it as

```
"hi i am 17"
 |  | |  |
 |  | |  v
 |  | v  TOKEN_FLOAT, value: 17.0  (automatic int->float conversion)
 |  | TOKEN_SYMBOL, value: "symbol value"  (a successfull hash table lookup
 |  |                                       turned a TOKEN_IDENTIFIER into a
 |  |                                       TOKEN_SYMBOL and took over the
 |  v                                       symbol's value)
 v  'i'  ('i' can be a valid token as well, as all chars >0 and <256)
TOKEN_IDENTIFIER, value: "hi"
```

You need to match the token sequence with your code, and if you encounter something that you don't want, you error out:

```
/* expect an identifier ("hi") */
```

```
g_scanner_get_next_token (scanner);
if (scanner->token != G_TOKEN_IDENTIFIER)
  return G_TOKEN_IDENTIFIER;
/* expect a token 'i' */
g_scanner_get_next_token (scanner);
if (scanner->token != 'i')
  return 'i';
/* expect a symbol ("am") */
g_scanner_get_next_token (scanner);
if (scanner->token != G_TOKEN_SYMBOL)
  return G_TOKEN_SYMBOL;
/* expect a float (17.0) */
g_scanner_get_next_token (scanner);
if (scanner->token != G_TOKEN_FLOAT)
  return G_TOKEN_FLOAT;
```

If you got past here, you have parsed "hi i am 17" and would have accepted "dooh i am 42" and "bah i am 0.75" as well, but you would have not accepted "hi 7 am 17" or "hi i hi 17".

Chapter 9. GTK+ FAQ Contributions, Maintainers and Copyright

If you would like to make a contribution to the FAQ, send either one of us an e-mail message with the exact text you think should be included (question and answer). With your help, this document can grow and become more useful!

This document is maintained by Tony Gale <gale@gtk.org> (mailto:gale@gtk.org) Nathan Froyd <maestrox@geocities.com> (mailto:maestrox@geocities.com), and Emmanuel Deloget <logout@free.fr> (mailto:logout@free.fr). This FAQ was created by Shawn T. Amundson <amundson@gimp.org> (mailto:amundson@gimp.org) who continues to provide support. Contributions should be sent to Tony Gale <gale@gtk.org> (mailto:gale@gtk.org)

If you are intending to incorporate this document into a published work, please contact one of the maintainers, and we will make an effort to ensure that you have the most up to date information available.

There is no guarentee that this document lives up to its intended purpose. This is simply provided as a free resource. As such, the authors and maintainers of the information provided within can not make any guarentee that the information is even accurate.