GTK+ FAQ

Tony Gale

Shawn Amundson

Emmanuel Deloget

GTK+ FAQ

por Tony Gale, Shawn Amundson, y Emmanuel Deloget

La intención de este documento, es responder las preguntas que con seguridad serán hechas por programadores que utilizan GTK+, o por personas que simplemente están averiguando sobre como utilizar GTK+.

Tabla de contenidos

Capítulo 1. GTK+ FAQ

Información General

Antes que todo: los agradecimientos

Los autores del FAQ quieren dar las gracias a:

- Havoc Pennington
- Erik Mouw
- Owen Taylor
- Tim Janik
- Thomas Mailund Jensen
- Joe Pfeiffer
- Andy Kahn
- Federico Mena Quntero
- Damon Chaplin
- y a todos los miembros de la lista GTK+

¡Si nos olvidamos de tí, por favor envíans un correo electrónico! Gracias denuevo (Ya se, es muy cortito :)

Autores

Los autores origianles de GTK+ fueron:

- Peter Mattis
- Spencer Kimball
- Josh MacDonald

Desde entonces, mucho ha sido agregado por otros. Por favor, revisa el archivo AUTHORS en la distribución para averiguarl el Equipo GTK+.

¿Qué es GTK+?

GTK+ es un conjunto pequeño y eficiente de *widgets* [1], diseñado con el aspecto y sensación general de Motif. En realidad, luce mucho mejor que Motif. Contiene widgets comunes y algunos widgets más complejos como un selector de archivos, y widgets para seleccionar colores.

GTK+ provee algunas características únicas. (Al menos no conozco de otra librería de widgets que las provea). Por ejemplo, un botón no contiene una etiqueta, contiene un widget hijo, que en la mayoría de las instancias será una etiqueta. Sin embargo, el widget hijo también puede ser un pixmap, una imágen o cualquier combinación posible que el programador desee. Esta flexibilidad está adherida en toda la librería.

¿Qué es el + en GTK+?

Peter Mattis informó a la lista de correo gtk que:

“Originalmente escribí gtk que incluía las tres librerías, libglib, libgdk y libgtk. Se caracterizaba por una jerarquía plana de widget. Osea que, no podías derivar un nuevo widget de uno existente. Contenía un mecanismo de callbacks más estándar en lugar del mecanismo de señales ahora presente en gtk+. El + fue agregado para distinguir entre la versión original de gtk y la nueva versión. Puedes pensar de ello como una mejora al gtk original que agrega características orientadas a objetos.”

“GTK+ provee algunas características únicas. (Al menos no conozco de otra librería de widgets que las provea). Por ejemplo, un botón no contiene una etiqueta, contiene un widget hijo, que en la mayoría de las instancias será una etiqueta. Sin embargo, el widget hijo también puede ser un pixmap, una imágen o cualquier combinación posible que el programador desee. Esta flexibilidad está adherida en toda la librería.”

¿La G in GTK+, GDK y GLIB signfica?

GTK+ == Gimp Toolkit (Juego de Herramientas de Gimp)

GDK == GTK+ Drawing Kit (Equipo de Dibujo de GTK+)

GLib == G Libray (Librería G)

¿Dónde está la documentación para GTK+?

En el directorio doc/ de la distribución de GTK+ encontrarás el material de referencia para GTK y GDK, este FAQ y el Tutorial GTK.

Además, puedes encontrar enlaces a versiones HTML de estos documentos dirigiendote a http://www.gtk.org/[2]. Una versión empaquetada del Tutorial GTK con versiones SGML, HTML, Postscript, DVI y texto, puede ser encontrado en Postscript, DVI and text versions can be found in ftp://ftp.gtk.org/pub/gtk/tutorial [3]

Hay un par de libros disponibles, lo cuales lidian con programación GTK+, GDK y GNOME:

- El libro de Eric Harlow titulado "Developing Linux Applications with GTK+ and GDK". El ISBN es 0-7357-0021-4
- El código de ejemplo del libro de Eric está disponible en-línea en http://www.bcpl.net/~eharlow/book[4]
- Havoc Pennington ha sacado un libro llamado "GTK+/GNOME Application Development". El ISBN es 0-7357-0078-8

  La versión libre/gratuita del libro vive aquí: http://developer.gnome.org/doc/GGAD/[5]

  Havoc mantiene información del libro y errata aquí: http://pobox.com/~hp/gnome-app-devel.html[6]

¿Existe una lista de correo (o archivo de lista de correo) para GTK+?

Información sobre listas de correo relacionadas con GTK+ se puede encontrar en http://www.gtk.org/mailinglists.html[7]

Cómo obtener ayuda con GTK+

Primero, asegúrate que tu pregunta no está respondida en la documentación, este FAQ o el tutorial. ¿Ya lo hiciste? ¿Seguro que lo hiciste, correcto? En ese caso, el mejor lugar para enviar tus preguntas es a la lista de correo GTK+.

Cómo reportar *bugs* en GTK+

Los bugs [8] deben ser reportados al sistema de GNOME para rastreo de bugs (http://bugs.gnome.org[9]). Para reportar un problema sobre GTK+, envíe correo a submit@bugs.gnome.org.

El sujeto del correo debe describir su problema. En el cuerpo del correo, debe incluír de primero un "pseudo-encabezado" que informe el paquete y número de versión. Esto debe ser separado de los encabezados reales por una línea en blanco.

```
Paquete: gtk+
Versión: 1.2.0
```

Substituya 1.2.0 con la versión de GTK+ que tenga instalada.

Entonces describa el bug. Incluya:

- Información sobre su sistema. Por ejemplo:
  - Qué sistema operativo y la versión
  - Qué versión de X
  - Para Linux, qué versión de la librería de C

  Y cualquier otra cosa que crea que es relevante

- Cómo reproducir el bug.

  Si lo puede reproducir con el programa testgtk que se construye en el subdirectorio gtk/, sería lo más conveniente. De otro modo, por favor incluya un programa de prueba corto que muestre el comportamiento. Como último recurso, también puede guiarnos a una pieza de software mayor que pueda ser bajada.

  (Los bugs que pueden ser reproducidos dentro de GIMP son casi tan tan buenos como los que pueden ser reproducidos en testgtk. Si está reportando un bug encontrado con el GIMP, por favor incluya el número de versión de GIMP que está utilizando)

- Si el bug colgó el programa, el texto exacto que se imprimió al ocurrir.
- Información adicional, como rastreos de la pila, puede ser de utilidad, pero no es necesaria. En caso de que envía un rastreo de la pila, y el error es un error de X, será más útil si el rastreo de la pila es producido corriendo el programa de prueba con la opción de la línea de comando `–sync`.

¿Existe una versión de GTK+ para Windows?

Hay un traslado activo de GTK+ a la plataforma windows, el cual está progresando impresionantemente.

Ver http://www.iki.fi/tml/gimp/win32[10] para más información.

¿Qué aplicaciones se han escrito con GTK+?

Una lista de algunas aplicaciones basadas en GTK+ se puede encontrar en el servidor de web de GTK+ en http://www.gtk.org/apps/[11] y contiene más de 350 aplicaciones.

Si eso falla, busca un proyecto para trabajar para el proyecto GNOME, http://www.gnome.org/[12] Escribe un juego. Escribe halgo que sea útil.

Algunos de estos son:

- GIMP (http://www.gimp.org/[13]), un programa de manipulación de imágenes
- AbiWord (http://www.abisource.com/[14]), un procesador de palabras profesional
- Gzilla (http://www.levien.com/gzilla/[15]), un navegador de web
- XQF (http://www.botik.ru/~roma/quake/[16]), un navegador y lanzador de servidores QuakeWorld/Quake2
- GDK Imlib (http://www.rasterman.com/imlib.html[17]), una librería para GDK para manipulación y carga rápida de imágenes
- Glade (http://glade.pn.org/[18]), una herramienta RAD basada en GTK+ que produce aplicaciones GTK+

Estoy planeando escribir una aplicación en GTK+. ¿Qué les parece un cliente IRC?

Pregunta en la lista gtk para sugerencias. Hay por lo menos tres clientes IRC bajo desarrollo (probablemente más de hecho. El servidor en http://www.forcix.cx/irc-clients.html[19] lista un montón de ellos).

- X-Chat.
- girc. (Incluído con GNOME)
- gsirc. (En el árbol CVS de gnome)

Cómo encontrar, configurar, instalar e investigar fallas en GTK+

¿Qué necesito para correr GTK+?

Para compilar GTK+, todo lo que necesitas es un compilador de C (gcc) y el Sistema de Ventana X y librerías asociadas en tu sistema.

¿Dónde puedo obtener GTK+?

El sitio canónico esftp://ftp.gtk.org/pub/gtk[20].

Este sitio tiende a sobrecargarse alrededor del lanzamiento de un nuevo GTK+, así que intenta utilizar alguno de los sitios espejo listados en ftp://ftp.gtk.org/etc/mirrors[21]

Aquí hay algunos sitios espejos para que puedas iniciar:

- África - ftp://ftp.is.co.za/applications/gimp[22]

- Australia - ftp://ftp.au.gimp.org/pub/gimp[23]
- Finlandia - ftp://ftp.funet.fi/pub/sci/graphics/packages/gimp[24]
- Alemania - ftp://infosoc.uni-koeln.de/pub/ftp.gimp.org"[25]
- Japón - ftp://SunSITE.sut.ac.jp/pub/archives/packages/gimp[26]
- Reino Unido - ftp://ftp.flirble.org/pub/X/gimp[27]
- EUA - ftp://ftp.insync.net/pub/mirrors/ftp.gimp.org[28]

¿Cómo configurar/compilar GTK+?

Por lo general, todo lo que necesitas es ejecutar los comandos:

```
./configure
make
```

en el directorio gtk+-versión/.

Cuando compilo GTK+ obtengo un error como: `make: file ‘Makefile’ line 456: Syntax error`

Asegúrate que utilizas GNU make (usa `make -v` para verificar). Hay muchas versiones raras y maravillosas de make por ahí, y no todas ellas pueden manejar los *Makefiles* generados automáticamente.

¡He compilado e instalado GTK+, pero no puedo lograr enlazar programa alguno con él!

Este problema se encuentra cuando las librerías GTK+ no pueden ser localizadas o son de una versión incorrecta. Generalmente, el compilador se queja de un ’unresolved symbol’. Hay dos cosas que necesitas verificar:

- Asegúrate que las librerías se pueden encontrar. Quieres editar `/etc/ld.so.conf` para incluír los directorios que contienen las librerías GTK, de manera que luzca parecido a esto:

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

  Entonces debes correr /sbin/ldconfig como root. Puedes averiguar en que directorio se encuentra GTK utilizando

  ```
  gtk-config -libs
  ```

  Si tu sistema no utiliza ld.so para encontrar librerías (como Solaris), entonces deberás utilizar la variable de entorno LD_LIBRARY_PATH (o compilara la ruta en tu programa, lo cual no voy a cubrir aquí). Así que, con una línea de comandos tipo Bourne, puedes (si tus librerías GTK están en /usr/local/lib):

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

  y en csh, puedes hacer esto:

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

- Asegúrate que el enlazador está localizando el conjunto correcto de librerías. Si tienes una distribución Linux que instala GTK+ (ejemplo RedHat 5.0), entonces esta versión vieja puede ser utilizada. Ahora (asumiendo que tienes un sistema RedHat), utiliza el comando

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

  También puedes queres remover los paquetes que dependen de gtk (rpm te dirá cuales son). Si no tienes un sistema RedHat Linux, asegúrate que ni `/usr/lib` ni `/usr/local/lib` contienen algunas de las librerías libgtk, libgdk, libglib, o libgck. En caso de que existan, remuevelas (y cualquier archivo *include* de gtk, como `/usr/include/gtk` y `/usr/include/gdk`) y reinstala gtk.

Cuando compilo programas con GTK+ obtengo mensajes de error del compilador sobre no ser capaz de encontrar `glibconfig.h`.

El archivo de encabezado "glibconfig.h" fue movido al directorio $exec_prefix/lib/glib/include/. $exec_prefix es el directorio que era especificado al dar la bandera –exec-prefix a ./configure cuando se compiló GTK+. Su valor por omisión es $prefix, (especificado con –prefix), cuyo valor por omisión es /usr/local.

Esto fue hecho porque "glibconfig.h" incluye información dependiente de la arquitectura, y el resto de los archivos include son colocados en $prefix/include, que puede ser compartido entre distintas arquitecturas.

GTK+ incluye un archivo de comandos, `/gtk-config/`, que hace fácil el encontrar los caminos correctos de inclusión. El Tutorial GTK+ incluye un ejemplo de como utilizar `/gtk-config/` para compilaciones simples desde la línea de comandos. Para información sobre configuraciones más complicadas, vea el archivo docs/gtk-config.txt en la distribución GTK+.

Si está intentando compilar un programa viejo, puede resolver el problema configurándolo con una línea de comando como:

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

(Sustituya el valor apropiado de $exec_prefix para /usr/local.)

Cuando instalo una aplicación GTK+, *configure* reporta que no puede encontrar a GTK.

Hay varias razones comúnes para esto:

- Tienes una versión vieja de GTK instalada en algún lugar. RedHat 5.0, por ejemplo, instala una copia vieja de GTK que no va a funcionar con las aplicaciones nuevas. Debes remover esta copia vieja, pero ten en cuenta que estó hará fallar las aplicaciones del `control-panel`.
- `gtk-config` (o algún otro componente de GTK) no está en la ruta de búsqueda, o hay una versión vieja en el sistema. Teclea:

  ```
  gtk-config -version
  ```

para revisar ambos casos. Si regresa un valor diferente de lo que esperas, entonces tienes una versión vieja de GTK en tu sistema.

- El archivo de comandos ./configure no puede encontrar las librerías GTK. Ver la pregunta de arriba para obtener ayuda en esto.

Si ninguna de las anteriores te sirve de ayuda, entonces revisa en config.log, el cual es generado por ./configure a medida que corre. Al final encontrarás la última acción que tomó antes de fallar. Si es una sección de código fuente, copia el código fuente a un archivo y compilalo con la línea justo encima de él en config.log. Si la compilación es exitosa, intenta ejecutarlo.

Desarrollo de GTK+

¿Qué es esta cosa CVS de la cual todo el mundo habla, y cómo puedo accesarla?

CVS es el Sistema de Versión Concurrente, y es una manera muy popular para controlar las versiones de proyectos de software. Está diseñado para permitir que autores múltiples operen simultáneamente en el mismo árbol fuente. Este árbol fuente se mantiene de manera central, pero cada desarrollador tiene un espejo local de este repositorio en donde hacen los cambios.

Los desarrolladores de GTK+ utilizan un repositorio CVS para guardar la copia maestra de la versión actual en desarrollo de GTK+. Como tal, quien quiera contribuír parches a GTK+, los deben generar contra la versión CVS. La gente normal debe utilizar los lanzamientos en paquete.

El conjunto de herramientas de CVS está disponible como paquetes RPM, en los lugares usuales de RedHat. La última versión está disponible en http://download.cyclic.com/pub/[29]

Cualquiera puede bajar la última versión CVS de GTK+, utilizando el acceso anónimo siguiendo estos pasos:

- En una línea de comandos descendiente de bourne (como bash) teclea:

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

- A continuación, la primera vez que se saca el árbol de desarrollo, se necesita un login cvs.

```
cvs login
```

  Te preguntará por un password. No hay password para cvs.gimp.org, así que solo presiona retorno de carro.

- Para conseguir el árbol y colocarlo en un subdirectorio de tu directorio de trabajo actual, teclea el comando:

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

  Ten en cuenta que con el árbol GTK+ 1.1, glib se ha movido a un módulo CVS separado, de manera que si no tienes glib instalado, también deberás obtenerlo:

```
cvs -z3 get glib
```

¿Cómo puedo contribuír a GTK+?

Es simple. Si algo no trabajo como crees que debería en un programa, revisa la documentación para asegurarte que no te hace falta algo. Si es un verdadero bug o funcionalidad perdida, rastréalo en el fuente de GTK+, cámbialo, y entonces genera un parche en la forma de un 'context diff'. Esto puede ser hecho utilizando un comando como `diff -ru <oldfile> <newfile>`. Entonces sube el parche a:

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

junto con un archivo README. ¡Asegúrate que sigues las convenciones de nombramiento o tu parche simplemente será borrado! Los nombres de archivo deben ser de esta forma:

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

La "n" en la fecha indica un número único (empezando de 0) de parches que has subido ese día. Debe ser 0, a menos que subas más de un parche en el mismo día.

Ejemplo:

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

Una vez que subas *lo que sea*, envía el README a ftp-admin@gtk.org

¿Cómo puedo saber si mi parche fue aplicado, y si no, porqué razón no lo fue?

Los parches subidos serán movidos a `ftp://ftp.gtk.org/pub/gtk/patches`, en donde alguien del equipo de desarrollo de GTK+ los recogerá. Si se aplican, se moverán a `/pub/gtk/patches/old`.

Los parches que no se aplican, por cualquier razón, son movidos a `/pub/gtk/patches/unapplied` o `/pub/gtk/patches/outdated`. En este punto puedes preguntar en la lista de correo `gtk-list` el por qué tu parche no fue aplicado. Hay muchas razones posibles por las cuales un parche puede no ser aplicado, desde que no se aplica limpiamente, a no ser correcto. No te sientas mal si tu parche no lo logró la primera vez.

¿Cuál es la política sobre la incorporación de nuevos widgets a la librería?

Esto le compete a los autores, así que deberás preguntarles una vez que hayas terminado con tu widget. Como guía general, los widgets que son generalmente útiles, trabajan, y no son una desgracia al conjunto de widget, serán incluídos agradecidamente.

¿Hay alguien trabajando en *bindings* para otros lenguajes además de C?

La página hogar de GTK+ (http://www.gtk.org/[30]) presenta una lista de bindings para GTK+.

- Hay varios envoltorios de C++ para GTK+.
  - el paquete gtk--, el cual es un envoltorio bastante pequeño para GTK+. Puedes encontrar su página hogar en http://www.cs.tut.fi/~p150650/gtk/gtk--.html[31]. El sitio FTP es ftp://ftp.gtk.org/pub/gtk/gtk--[32].
  - el paquete VDK, el cual fue construido como el paquete base de un constructor de aplicaciones GTK+ parecido a Borland. La página hogar se encuentra en http://www.guest.net/homepages/mmotta/VDKHome[33].
  - El paquete wxWindows/Gtk, que es una librería C++ para desarrollo de interfaces gráficas de usuario en plataformas cruzadas. La página hogar de este paquete se encuentra en http://www.freiburg.linux.de/~wxxt/[34].
- Hay tres bindings conocidos para Objective-c actualmente en desarrollo:
  - El paquete seleccionado http://www.gnome.org/[35] es objgtk. Objgtk está basado en la clase Object y es mantenido por Elliot Lee[36]. Aparentemente, objgtk está siendo aceptado como el binding Objective-c 'estandard' para GTK+.
  - Si te inclinas más al proyecto GNUstep[37], puede que quieras revisar GTKKit por Helge Heß[38]. La intención es crear un binding GTK+ utilizando el FoundationKit. GTKKit incluye cosas agradables como escribir un archivo plantilla tipo XML para construír una interface GTK+.
  - El paquete GToolKit, que puede ser encontrado en ftp://ftp.gtk.org/pub/gtk/objc-gtoolkit/[39].
- Bindings para Perl ftp://ftp.gtk.org/pub/gtk/perl[40]
- Bindings para Guile. La página hogar se encuentra en http://www.ping.de/sites/zagadka/guile-gtk[41]. Es importante mencionar que Guile es la implementación de Scheme R4RS del Proyecto GNU (el estándard). Si te gusta Scheme, querrás darle un vistazo a esto.
- David Monniaux reporta: "Inicié un sistema de binding gtk-O'Caml. Lo básico del sistema, incluyendo *callbacks*, trabaja bien. El desarrollo actual se encuentra en http://www.ens-lyon.fr/~dmonniau/arcs[42]"
- Se han hecho varios bindings de Python:
  - pygtk está en http://www.daa.com.au/~james/pygtk[43] y ftp://ftp.gtk.org/pub/gtk/python[44]
  - python-gtk está en http://www.ucalgary.ca/~nascheme/python-gtk[45]
- Hay un par de widgets OpenGL/Mesa disponibles para GTK+. Sugiero que empieces en http://www.student.oulu.fi/~jlof/gtkglarea/index.html[46]
- Por último, hay varios bindings para otros lenguajes más como Eiffel, TOM, Pascal, Pike, etc.

Desarrollando con GTK+: el inicio

¿Cómo empiezo?

Después de haber instalado GTK+, hay un par de cosas que pueden ayudarte a empezar a desarrollar aplicaciones con él. Está el Tutorial GTK+ http://www.gtk.org/tutorial/[47], el cual está en desarrollo. Esto te introducirá a la escritura de aplicaciones utilizando C.

El Tutorial no contiene (todavía) información en todos los widgets que existen en GTK+. Para código de ejemplo en como utilizar lo básico de todos los widgets GTK+, debes ver el archivo gtk/testgtk.c (y archivos fuentes asociados) dentro de la distribución. Revisar la distribución te dará una buena base sobre que pueden hacer los widgets.

Traté de compilar un pequeño Hola Mundo propio, pero falló. ¿Alguna pista?

Ya que eres bueno codificando, no lidiaremos con errores en tiempo de compilación :)

La línea de comando clásica para compilar un programa basado en GTK+ es

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

Debes darte cuenta del caracter de comilla hacia atrás que es utilizado en esta línea de comando. Un error común cuando inicias un desarrollo basado en GTK+, es utilizar comillas en lugar de comillas hacia atrás. Si lo haces, el compilador se quejará sobre un archivo desconocido llamado `gtk-config -cflags -libs`. El texto en comillas hacia atrás es una instrucción a tu procesador de línea de comando para substituír la salida de la ejecución de este texto en la línea de comando.

La línea de comando de arriba se asegura de:

- se utilizarán las banderas correctas para el compilador C para compilar el programa (incluyendo la lista completa de encabezados C)
- tu programa será enlazado con las librerías necesaras.

¿Qué hay sobre utilizar la utilidad make?

Este es un archivo make de ejemplo que compila un programa basado en GTK+:

```
# archivo make para aplicación básica GTK+
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 $<

# fin del archivo
```

Para más información sobre la utilidad **make**, debes leer la página **man** o la página **info** relevante.

Utilizo la cosa de comillas hacia atrás en mis archivos make, pero mi proceso make falló.

La construcción de comillas hacia atrás parece no ser aceptada por algunas utilidades **make** viejas. Si utilizas alguna de estas, el proceso make probablemente fallará. Para poder tener la syntaxis de comillas hacia atrás funcionando, debes utilizar la utilidad GNU make (consíguela en el servidor ftp de GNU en ftp://ftp.gnu.org/[48]).

Quiero agregar algunas cosas de configure. ¿Cómo pudo hacer esto?

Para utilizar **autoconf/automake**, primero debes instalar los paquetes relevantes. Estos son:

- el preprocesado m4 v1.4 o mejor
- autoconf v2.13 o mejor
- automake v1.4 o mejor

Encontrarás estos paquetes en el servidor principal de ftp de GNU (ftp://ftp.gnu.org/[49]) o en cualquier espejo GNU.

Para poder utilizar el poderoso esquema autoconf/automake, debes crear un configure.in que pude lucir como:

```
dnl Procesa este archivo con autoconf para producir un config-
ure script.
dnl configure.in para un programa basado en GTK+

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

dnl Chequea programas
AC_PROG_CC dnl chequea el compilador c
dnl deberías agregar CFLAGS="" aquí, poruqe es seteado a -
g por PROG_CC

dnl Chequea librerías.
AM_PATH_GTK(1.2.0„AC_MSG_ERROR(mypkgname 0.1 needs GTK))dnl

AC_OUTPUT(
Makefile
)dnl
```

Debes agregar un archivo Makefile.am:

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

Si tu proyecto contiene más de un subdirectorio, deberás crear un Makefile.am en cada directorio además de un Makefile.am maestro que lucirá como:

```
SUBDIRS         = midir1 midir2 midir3
```

entonces, para utilizar estos, simplemente teclea lo siguientes comandos:

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

Para más información, deberías revisar la documentación de autoconf y automake (los archivos info incluídos son bastante fáciles de entender, y hay una gran cantidad de recursos en el web que lidian con autoconf y automake).

Trato de eliminar errores en mi aplicación GTK+ con gdb, pero cuelga mi servidor X cuando llego a algún punto de parada. ¿Alguna idea?

De Federico Mena Quintero:

“X no esta trancado. Lo más seguro es que estás llegando a un punto de parada dentro de un *callback* que es llamado de un lugar en Gtk que tiene el ratón agarrado.”

“Corre tu programa con la opción `–sync`; hará más fácil la eliminación de errores. También, puede que quieras utilizar la consola para correr el debugger, y simplemente dejes el programa corriendo en otra consola con el servidor X.”

Eric Mouw tiene otra solución:

“Una terminal vieja conectada a un puerto serial sin usar, también es grandioso para eliminar errores en programas X. Terminales viejas vt100/vt220 son baratísimas pero un poco difíciles de conseguir (aquí en los Países Bajos, YMMV[50])”

Desarrollo con GTK+: preguntas generales

¿Qué widgets hay en GTK+?

El Tutorial GTK+ lista los siguientes 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
```

¿Es GTK+ seguro con hilos? ¿Cómo escribo aplicaciones multi-hilo con GTK+?

La librería GLib puede ser utilizada en un modo seguro-con-hilos, al llamar g_thread_init() antes de hacer cualquier otra llamada GLib. En este modo GLib tranca automáticamente todas las estructuras de datos internas que necesite trancar. Esto no significa que dos hilos puedan accesar simultáneamente, por ejemplo, un tabla hash, pero pueden accesar dos tablas hash diferentes simultáneamente. Si dos hilos distintos necesitan accesar la misma tabla hash, la aplicación es responsable por asegurarse a sí misma.

Cuanco GLib se inicializa para ser seguro-con-hilos, GTK+ esta *atento a los hilos*. Hay una cerradura global que debes obtener con gdk_threads_enter() antes de hacer cualquier llamada GDK, y la sueltas con gdk_threads_leave().

Un programa main mínimo para una aplicación GTK+ con hilos luce como:

```
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);
}
```

Los callbacks requieren un poco de atención. Los callbacks de (señales) GTK+ se hacen dentro de la cerradura GTK+. Sin embargo, las llamadas desde GLib (timeouts, IO callbacks, y funciones inactivas) se hacen fuera de la cerradura GTK+. Así que, dentro de un manejador de señales no necesitas llamar a gdk_threads_enter(), pero desde los otros tipos de callbacks, si tienes que hacerlo.

Erik Mouw contribuyó el siguiente código de ejemplo para ilusrar cómo utilizar hilos dentro de programas GTK+.

```
/*-------------------------------------------------
 * Nombre de archivo: gtk-thread.c
 * Versión:           0.99.1
 * Copyright:         Copyright (C) 1999, Erik Mouw
 * Autor:             Erik Mouw <J.A.K.Mouw@its.tudelft.nl>
 * Descripción:       Ejemplo de hilos GTK.
 * Creado en:         Dom 17 Oct 21:27:09 1999
 * Modificado por:    Erik Mouw <J.A.K.Mouw@its.tudelft.nl>
 * Modificado en:     Dom 24 Oct 17:21:41 1999
 *-----------------------------------------------*/
/*
 * Compilar con:
 *
 * cc -o gtk-thread gtk-thread.c `gtk-config -cflags -libs gthread`
 *
 * Gracias a Sebastian Wilhelmi y Owen Taylor por señalar algunos
 * 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(;;)
    {
      /* duerme un rato */
      sleep(rand() / (RAND_MAX / 3) + 1);

      /* trancar la yes_or_no_variable */
      G_LOCK(yes_or_no);

      /* ¿tenemos que decir algo? */
      say_something = (yes_or_no != data->what);

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

      /* Soltar la variable yes_or_no */
      G_UNLOCK(yes_or_no);

      if(say_something)
{
  /* conseguir la cerradura GTK */
  gdk_threads_enter();

  /* asignar el texto de la etiqueta */
  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!");

  /* soltar la cerradura del hilo GTK */
  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;

  /* inicializar hilos */
  g_thread_init(NULL);

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

  /* inicializar generador de números aleatorios */
  srand((unsigned int)time(NULL));

  /* crear una ventana */
  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);

  /* crear una etiqueta */
  label = gtk_label_new("And now for something completely differ-
ent ...");
  gtk_container_add(GTK_CONTAINER(window), label);

  /* mostrar todo */
  gtk_widget_show(label);
  gtk_widget_show (window);

  /* crear los hilos */
  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);

  /* entrar al ciclo principal de GTK */
  gdk_threads_enter();
  gtk_main();
  gdk_threads_leave();

  return(0);
}
```

¿Por qué ocurre este extraño error ’x io error’ cuando hago un `fork()` en mi aplicación GTK+

Este no es en realidad un problema de GTK+, y el problema no está relacionado a `fork()` tampoco. Si ocurre 'x io error', entonces probablemente estás usando la función `exit()` para salir de un proceso hijo.

Cuando GDK abre una pantalla X, crea un descriptor de archivo de socket. Cuando utilizas la función `exit()`, implícitamente cierras todos los descriptores de archivos abiertos, y esto no le agrada a la librería X subyacente.

La función correcta a utilizar en este caso es `_exit()`.

Erik Mouw contribuyó el siguiente código de ejemplo para ilustrar el manejo de fork() y exit().

```
/*-------------------------------------------------
 * Nombre de archivo: gtk-fork.c
 * Versión:           0.99.1
 * Copyright:         Copyright (C) 1999, Erik Mouw
 * Autor:             Erik Mouw <J.A.K.Mouw@its.tudelft.nl>
 * Descripción:       Ejemplo de fork en GTK+
 * Creado en:         Jue 23 Sep 21:37:55 1999
 * Modificado por:    Erik Mouw <J.A.K.Mouw@its.tudelft.nl>
 * Modificado en:     Jue 23 Sep 22:39:39 1999
 *-----------------------------------------------*/
/*
 * Compilar con:
 *
 * 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;

  /* blockear otras señales SIGCHLD que vengan */
  sigemptyset(&set);
  sigaddset(&set, SIGCHLD);
  sigprocmask(SIG_BLOCK, &set, &oldset);

  /* esperar por el hijo */
  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-
instalar el manejador de señal (algunos sistemas necesitan esto) */
  signal(SIGCHLD, sigchld_handler);

  /* y desbloquearlo */
  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, falló el fork() */
      perror("fork");
      exit(-1);
    }
  else if(pid == 0)
    {
      /* hijo */
      fprintf(stderr, "Child: pid = %d\n", (int)getpid());

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

      /* si exec() retorna, hay algo malo */
      perror("execlp");
```

```
      /* salir del hijo. nota el uso de _exit() en lu-
gar de exit() */
      _exit(-1);
    }
  else
    {
      /* padre */
      fprintf(stderr, "Par-
ent: forked a child with pid = %d\n", (int)pid);
    }
}

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

  gtk_init(&argc, &argv);

  /* las cosas básicas: hacer una ventana y colocar call-
backs para destruír y
   * borrar eventos
   */
  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

  /* agregar un botón para hacer algo útil */
  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);

  /* mostrar todo */
  gtk_widget_show (button);
  gtk_widget_show (window);


  /* instalar un manejador de señal para señales SIGCHLD */
  signal(SIGCHLD, sigchld_handler);


  /* ciclo principal */
  gtk_main ();

  exit(0);
}
```

¿Por qué el contenido de un botón no se mueve cuando el botón se presiona? Aquí les tengo un parche para que funcione de esa manera...

De: Peter Mattis

"La razón por la cual los botones no mueven su hijo hacia abajo y la derecha cuando son presionados es porque no me parece que eso es lo que está ocurriendo visualmente. Mi visión de los botonos es que los estás mirando en línea recta. Osea que, la interface del usuario está en un plano y tú estás encima de este mirándolo recto. Cuando un botón es oprimido se mueve directamente lejos de tí. Para ser absolutamente correcto yo supongo que el hijo debería encojerse un poquito. Pero no veo por que el hijo se debería mover hacia abajo y a la izquierda. Recuerda, se supone que el hijo está pegado a la superficie del botón. No es bueno que aparezca como si el hijo se resbalara en la superficie del botón."

"En una nota más práctica, yo sí implemente esto en algún momento y decidí que no se veía bien y lo removí."

¿Cómo identifico la ventana superior de un widget u otro ancestro?

Hay un par de maneras de encontrar el padre superior de un widget. La más fácil es llamar la función `gtk_widget_top_level()` que regresa un puntero a GtkWidget que es la ventana superior.

Una manera más complicada de hacer esto (pero menos limitada, ya que permite al usuario obtener el ancestro más cercano de un tipo conocido) es utilizar `gtk_widget_get_ancestor()` como en:

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

Ya que virtualmente todos los GTK_TYPEs pueden ser usados como el segundo parámetro de esta función, puedes obtener cualquier widget padre de un widget en particular. Supón que tienes un hbox que contiene un vbox, que en su lugar contiene algún otro widget atómico (entry, label, etc. Para encontrar el hbox maestro utilizando el widget `entry` simplemente usa:

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

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);
```

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(GtkWiget *widget, GdkEvenBut-
ton *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\",
           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.)”

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[51]).

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."

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);
```

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.

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), sig-
nal_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.

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).

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)`.

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).

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 *ob-
ject, 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.

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)

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);
```

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.

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);
```

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 han-
dled this event; the buck
         * stops here. */
        return TRUE;
    }

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

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.

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.

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.

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);
```

Development with GTK+: widget specific questions

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");
}
```

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);
```

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.

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);
```

How can I define a separation line in a menu?

See the Tutorial[52] 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);
```

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);
```

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 (win-
dow1), "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, "acti-
vate_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);
```

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 */
  }
```

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_MISK(label), 0.0f, 0.0f);

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

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

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.

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);
```

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"
```

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
   * maxi-
mum 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);
```

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);
```

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 (ed-
itable), "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);
}
```

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.

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-*-*-
*-*-*-*");
```

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.

About GDK

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.

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, indicat-
ing the RGB triple
   * of the color we want to draw. Note that the val-
ues 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 col-
ormap 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 value is only truly valid on 24-
bit (TrueColor)
   * displays. Therefore, this call is re-
quired 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);

  ...
}
```

About GLib

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

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.

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);
```

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)."

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, "sym-
bol", 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 ta-
ble lookup
 |  |                                       turned a TO-
KEN_IDENTIFIER into a
 |  |                                       TO-
KEN_SYMBOL and took over the
 |  v                                       symbol's value)
 v  'i'  ('i' can be a valid to-
ken 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".

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>[53] Nathan Froyd <maestrox@geocities.com>[54], and Emmanuel Deloget <logout@free.fr>[55]. This FAQ was created by Shawn T. Amundson <amundson@gimp.org>[56] who continues to provide support. Contributions should be sent to Tony Gale <gale@gtk.org>[57]

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.

Notas

1. Nota del traductor: Widget significa Componente Gráfico Reutilizable.
2. http://www.gtk.org/
3. ftp://ftp.gtk.org/pub/gtk/tutorial
4. http://www.bcpl.net/~eharlow/book
5. http://developer.gnome.org/doc/GGAD/
6. http://pobox.com/~hp/gnome-app-devel.html
7. http://www.gtk.org/mailinglists.html
8. Nota del traductor: Bug significa Error de Programación
9. http://bugs.gnome.org
10. http://www.iki.fi/tml/gimp/win32
11. http://www.gtk.org/apps/
12. http://www.gnome.org/
13. http://www.gimp.org/
14. http://www.abisource.com/
15. http://www.levien.com/gzilla/
16. http://www.botik.ru/~roma/quake/
17. http://www.rasterman.com/imlib.html
18. http://glade.pn.org/
19. http://www.forcix.cx/irc-clients.html
20. ftp://ftp.gtk.org/pub/gtk

21. ftp://ftp.gtk.org/etc/mirrors
22. ftp://ftp.is.co.za/applications/gimp
23. ftp://ftp.au.gimp.org/pub/gimp
24. ftp://ftp.funet.fi/pub/sci/graphics/packages/gimp
25. ftp://infosoc.uni-koeln.de/pub/ftp.gimp.org
26. ftp://SunSITE.sut.ac.jp/pub/archives/packages/gimp
27. ftp://ftp.flirble.org/pub/X/gimp
28. ftp://ftp.insync.net/pub/mirrors/ftp.gimp.org
29. http://download.cyclic.com/pub/
30. http://www.gtk.org/
31. http://www.cs.tut.fi/~p150650/gtk/gtk–.html
32. ftp://ftp.gtk.org/pub/gtk/gtk–
33. http://www.guest.net/homepages/mmotta/VDKHome
34. http://www.freiburg.linux.de/~wxxt/
35. http://www.gnome.org/
36. mailto:sopwith@cuc.edu
37. http://www.gnustep.org/
38. mailto:helge@mdlink.de
39. ftp://ftp.gtk.org/pub/gtk/objc-gtoolkit/
40. ftp://ftp.gtk.org/pub/gtk/perl
41. http://www.ping.de/sites/zagadka/guile-gtk
42. http://www.ens-lyon.fr/~dmonniau/arcs
43. http://www.daa.com.au/~james/pygtk
44. ftp://ftp.gtk.org/pub/gtk/python
45. http://www.ucalgary.ca/~nascheme/python-gtk
46. http://www.student.oulu.fi/~jlof/gtkglarea/index.html
47. http://www.gtk.org/tutorial/
48. ftp://ftp.gnu.org/
49. ftp://ftp.gnu.org/
50. Nota del traductor: YMMV == Tu Millaje Puede Variar
51. http://www106.pair.com/rhp/sample_chapters.html
52. http://www.gtk.org/tutorial/
53. mailto:gale@gtk.org
54. mailto:maestrox@geocities.com
55. mailto:logout@free.fr
56. mailto:amundson@gimp.org
57. mailto:gale@gtk.org