在 Windows 下以 GTK+撰寫視窗程式

作者：徐天送、陳孟哲
email：s2598003@ntut.edu.tw
網址：http://www.ntut.edu.tw/~s2598003/
日期：2004/11/2（0.01 版）

前言

在 Win32 平台下，若是要開發視窗程式，並且使撰寫出來的程式能夠在 Linux 下也能編譯並執行，除了 Qt 之外，我們還可以有另一個選擇，就是 GTK+。GTK+這套 Graphical Library 已完整移植至 Windows 平台上，我們可以用 MinGW 編譯 GTK+開發的視窗程式，而產生出來的執行程式則是以 Win32 Native API 的方式執行。在這份文件，我們將以 Eclipse 整合環境，加上 GTK+的 libraries 來一個 HelloGTK 視窗程式。此外，我們也假設在 Windows 上已安裝 MinGW，且可編輯、執行 C++程式。

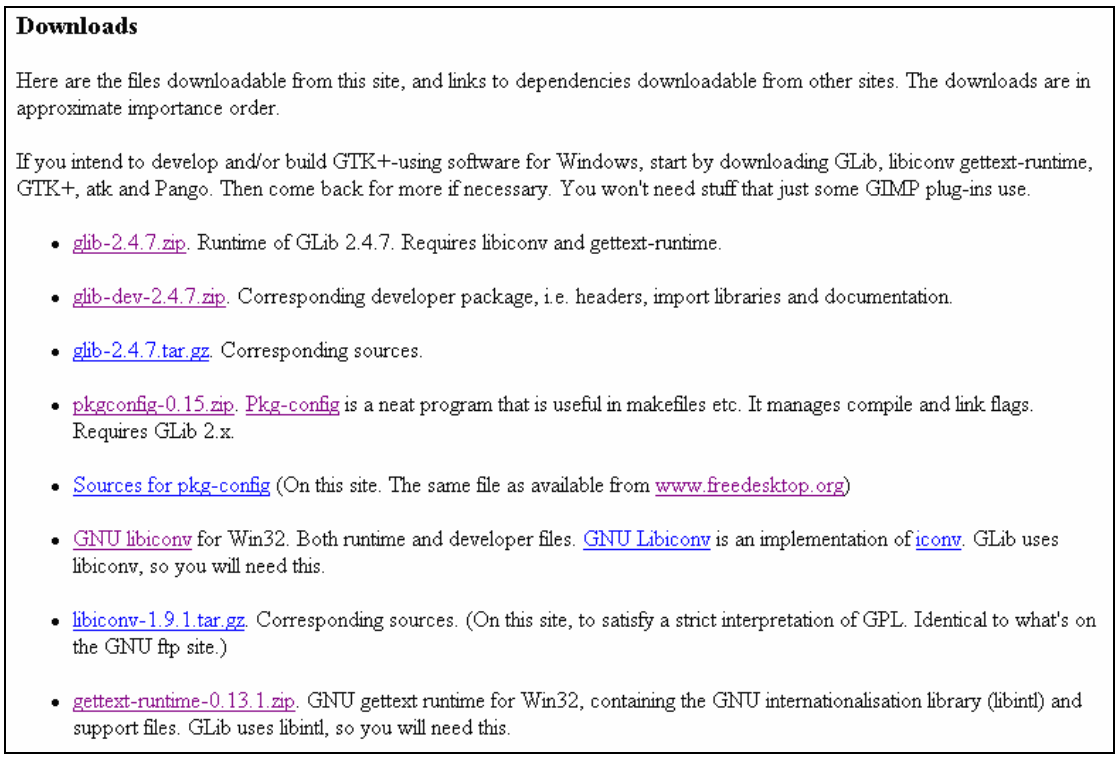
安裝 GTK+

GTK+最主要分成三個主要的 libraries 來互相合作：

- **GLib** is the low-level core library that forms the basis of GTK+ and GNOME. It provides data structure handling for C, portability wrappers, and interfaces for such runtime functionality as an event loop, threads, dynamic loading, and an object system.
- **Pango** is a library for layout and rendering of text, with an emphasis on internationalization. It forms the core of text and font handling for GTK+-2.0.
- **ATK** library provides a set of interfaces for accessibility. By supporting the ATK interfaces, an application or toolkit can be used with such tools as screen readers, magnifiers, and alternative input devices.

然而，由於 GTK 是由許多 libraries 組成，在 http://www.gtk.org/網站，我們可以看到如圖 1 的畫面，而這裡只是一部份而已，因此，我們將必須安裝的 libraries 整理成表 1，你可以根據表 1 將這些必要的 library 一一下載下來[1]。(可以直接選擇 library file 的名稱下載)

[1] 爲了下載方便，我將這些檔案再整理成一個 zip 檔，你可以到 http://www.ntut.edu.tw/~s2598003/gtk2.4.7_all.zip 下載，下載後解壓縮後，即是表 1 所列的 11 個檔案。



Downloads

Here are the files downloadable from this site, and links to dependencies downloadable from other sites. The downloads are in approximate importance order.

If you intend to develop and/or build GTK+-using software for Windows, start by downloading GLib, libiconv gettext-runtime, GTK+, atk and Pango. Then come back for more if necessary. You won't need stuff that just some GIMP plug-ins use.

- glib-2.4.7.zip. Runtime of GLib 2.4.7. Requires libiconv and gettext-runtime.
- glib-dev-2.4.7.zip. Corresponding developer package, i.e. headers, import libraries and documentation.
- glib-2.4.7.tar.gz. Corresponding sources.
- pkgconfig-0.15.zip. Pkg-config is a neat program that is useful in makefiles etc. It manages compile and link flags. Requires GLib 2.x.
- Sources for pkg-config (On this site. The same file as available from www.freedesktop.org)
- GNU libiconv for Win32. Both runtime and developer files. GNU Libiconv is an implementation of iconv. GLib uses libiconv, so you will need this.
- libiconv-1.9.1.tar.gz. Corresponding sources. (On this site, to satisfy a strict interpretation of GPL. Identical to what's on the GNU ftp site.)
- gettext-runtime-0.13.1.zip. GNU gettext runtime for Win32, containing the GNU internationalisation library (libintl) and support files. GLib uses libintl, so you will need this.

圖 1：GTK+下載畫面

表 1：GTK+下載 library 一覽表

Library File	Notes
gettext-runtime-0.13.1.zip	GNU gettext runtime for Win32, containing the GNU internationalisation library (libintl) and support files. GLib uses libintl, so you will need this.
glib-2.4.7.zip	Runtime of GLib 2.4.7. Requires libiconv and gettext-runtime.
libiconv-1.9.1.bin.woe32.zip	Both runtime and developer files. GNU Libiconv is an implementation of iconv. GLib uses libiconv, so you will need this.
gtk+-2.4.13.zip	Runtime of GTK+ 2.4.13. Requires glib, atk and pango. In addition, if you want to be able to load PNG, JPEG or TIFF images with gdk-pixbuf, you will need libpng and zlib, libjpeg and libtiff respectively.
pkgconfig-0.15.zip	Pkg-config is a neat program that is useful in makefiles etc. It manages compile and link flags. Requires GLib 2.x.
glib-dev-2.4.7.zip	Corresponding developer package, i.e. headers, import libraries and documentation.

gtk+-dev-2.4.13.zip	Corresponding developer package.
pango-1.4.1.zip	Pango 1.4.1. Pango is used by GTK+ 2 and GIMP 2. Note that Pango 1.6.0 has a bug that affects the Win32 backend, and should not be used on Windows.
pango-dev-1.4.1.zip	Corresponding developer package.
atk-1.6.0.zip	Atk is used by GTK+ 2.
atk-dev-1.6.0.zip	Corresponding developer package.

下找完畢後，將所有的檔案解壓縮至 MinGW 的根目錄下(例如 C:\MinGW)，總共 11 個壓縮檔，解壓縮過程中，會有少許檔案會重複，此時只要擇擇全部取代即可。

取得 GTK+環境變數

由於我們在編譯 c++程式，必須 Include Header File 的路徑並且指定要連結的 libraries 路徑，然而 GTK+其 library 的數量眾多，我們難以一一加入所有的 paths。所幸的是 GTK+提供一個名爲 pkg-config 的程式，幫我們取得目前 GTK+相關的 Include 和 libraries 的路徑及其參數，我們先將取得的所有路徑及參數複製下來，待撰寫 Makefile 時再使用這些值。**第一個指令是取得 GTK+所有 Include 的路徑，而第二個指令是取得 GTK+所有 libraries 的路徑。**

我們首先在 Windows 下執行 Command Mode，開啓後輸入以下指令

```
pkg-config --cflags gtk+-2.0
```

之後會出現如下訊息的畫面，如圖 2 所示，我們再將這些字串複製下來。

```
D:\OOP\Workspace\HelloGTK>pkg-config --cflags gtk+-2.0
-IC:/MinGW/include/gtk-2.0 -IC:/MinGW/lib/gtk-2.0/include -IC:/MinGW/include/atk
-1.0 -IC:/MinGW/include/pango-1.0 -IC:/MinGW/include/glib-2.0 -IC:/MinGW/lib/gli
b-2.0/include
```

圖 2：取得 cflags 的參數及 Include 的 Path

接下來我們再輸入，其結果如圖 3 所示：

```
pkg-config --libs gtk+-2.0
```

```
D:\OOP\Workspace\HelloGTK>pkg-config --libs gtk+-2.0
-LC:/MinGW/lib -lgtk-win32-2.0 -lgdk-win32-2.0 -latk-1.0 -lgdk_pixbuf-2.0 -lpang
owin32-1.0 -lgdi32 -lpango-1.0 -lgobject-2.0 -lgmodule-2.0 -lglib-2.0
```

圖 3：取得 libs 的變數

同樣地，我們也將這一字串複製下來，待未來編譯時使用。(若是 MinGW 安裝的路徑與我們相同，則上述所記錄的值也應該同上兩張圖中的參數、路徑)

撰寫 HelloGTK 視窗程式

我們現在實作一 HelloGTK 範例來說明編譯、執行的過程即結果。至於關於如何撰寫 GTK+的程式，你可以參考 http://www.gtk.org/tutorial/這個網站的 GTK+的 Tutorial。而此 HelloGTK 則是改寫 Tutorial 中的 Helloworld 程式，其程式列表如下：

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

static gboolean delete event( GtkWidget *widget,
                          GdkEvent  *event,
                          gpointer   data )
{
    /* If you return FALSE in the "delete event" signal handler,
     * GTK will emit the "destroy" signal. Returning TRUE means
     * you don't want the window to be destroyed.
     * This is useful for popping up 'are you sure you want to quit?'
     * type dialogs. */

    g_print ("delete event occurred\n");

    /* Change TRUE to FALSE and the main window will be destroyed with
     * a "delete_event". */

    return FALSE;
}

/* Another callback */
static void destroy( GtkWidget *widget,
                     gpointer   data )
{
    gtk_main_quit ();
}

int main( int   argc,
          char *argv[] )
{
    /* GtkWidget is the storage type for widgets */
    GtkWidget *window;
    GtkWidget *label;

    /* This is called in all GTK applications. Arguments are parsed
     * from the command line and are returned to the application. */
    gtk_init (&argc, &argv);

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

    /* When the window is given the "delete event" signal (this is given
     * by the window manager, usually by the "close" option, or on the
     * titlebar), we ask it to call the delete event () function
     * as defined above. The data passed to the callback
     * function is NULL and is ignored in the callback function. */
    g_signal_connect (G_OBJECT (window), "delete_event",
```

```
                G_CALLBACK (delete_event), NULL);

    /* Here we connect the "destroy" event to a signal handler.
     * This event occurs when we call gtk widget destroy() on the window,
     * or if we return FALSE in the "delete event" callback. */
    g signal connect (G OBJECT (window), "destroy",
                G_CALLBACK (destroy), NULL);

    /* create a label with "HelloGTK" text */
    label = gtk_label_new ("Hello GTK");

    /* This packs the label into the window (a gtk container). */
    gtk_container_add (GTK_CONTAINER (window), label);

    /* Sets the border width of the window. */
    gtk_container_set_border_width (GTK_CONTAINER (window), 10);

    /* The final step is to display this newly created widget. */
    gtk_widget_show (label);

    /* and the window */
    gtk_widget_show (window);

    /* All GTK applications must have a gtk main(). Control ends here
     * and waits for an event to occur (like a key press or
     * mouse event). */
    gtk_main ();

    return 0;
}
```

接下來我們得撰寫一 Makefile 來編譯我們的視窗程式：

```
gtk include=-IC:/MinGW/include/gtk-2.0
-IC:/MinGW/lib/gtk-2.0/include -IC:/MinGW/include/atk-1.0
-IC:/MinGW/include/pango-1.0 -IC:/MinGW/include/glib-2.0
-IC:/MinGW/lib/glib-2.0/include


gtk lib=-LC:/MinGW/lib -lgtk-win32-2.0 -lgdk-win32-2.0 -latk-1.0
-lgdk pixbuf-2.0 -lpangowin32-1.0 -lgdi32 -lpango-1.0 -lgobject-2.0
-lgmodule-2.0 -lglib-2.0


target_dir=Debug

main: HelloGTK.cpp
    gcc -mms-bitfields -Wall -g HelloGTK.cpp -o HelloGTK \
    ${gtk include} ${gtk_lib}
    mkdir ${target dir}
    mv *.exe ${target_dir}/

all:
```

注意：沒有換行

GTK+的 Include 路徑，由 pkg-config --cflags gtk+-2.0 取得

GTK+的 Libraries 路徑，由 pkg-config --libs gtk+-2.0 取得

```
    ${MAKE} main

.PHONY : clean
clean :
    -rm ${target dir}/*.exe
    rmdir ${target_dir}
```

接下來我們可以開始準備 Build 我們的程式了，如同一般 Build 方式，我們選擇【Project】→【Build Project】即開始編譯，編譯結果如圖 4 所示，並產生一個 HelloGTK.exe 的執行檔，最後，我們選擇【Run】→【Run…】開啓一個 Run Wizard 來執行程式。選擇執行檔，且設定完後，按下【Run】即可執行我們的應用程式，執行結果如圖 5。總算大功告成了！

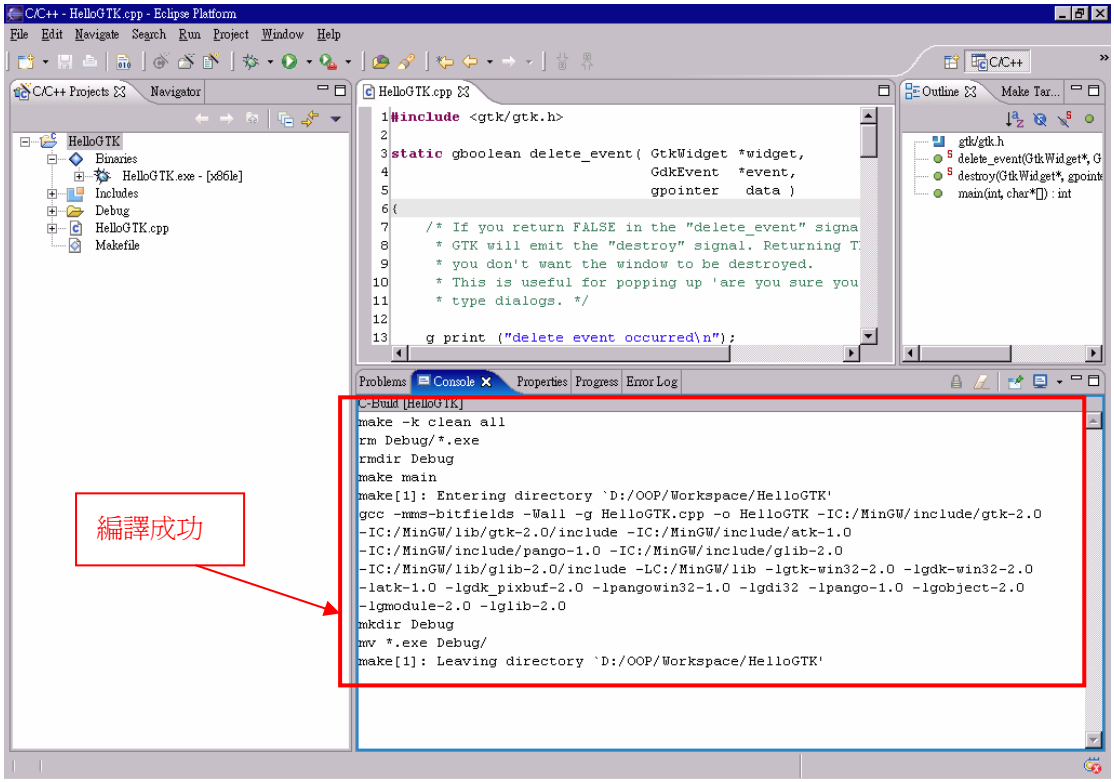


圖 4：編譯成功畫面

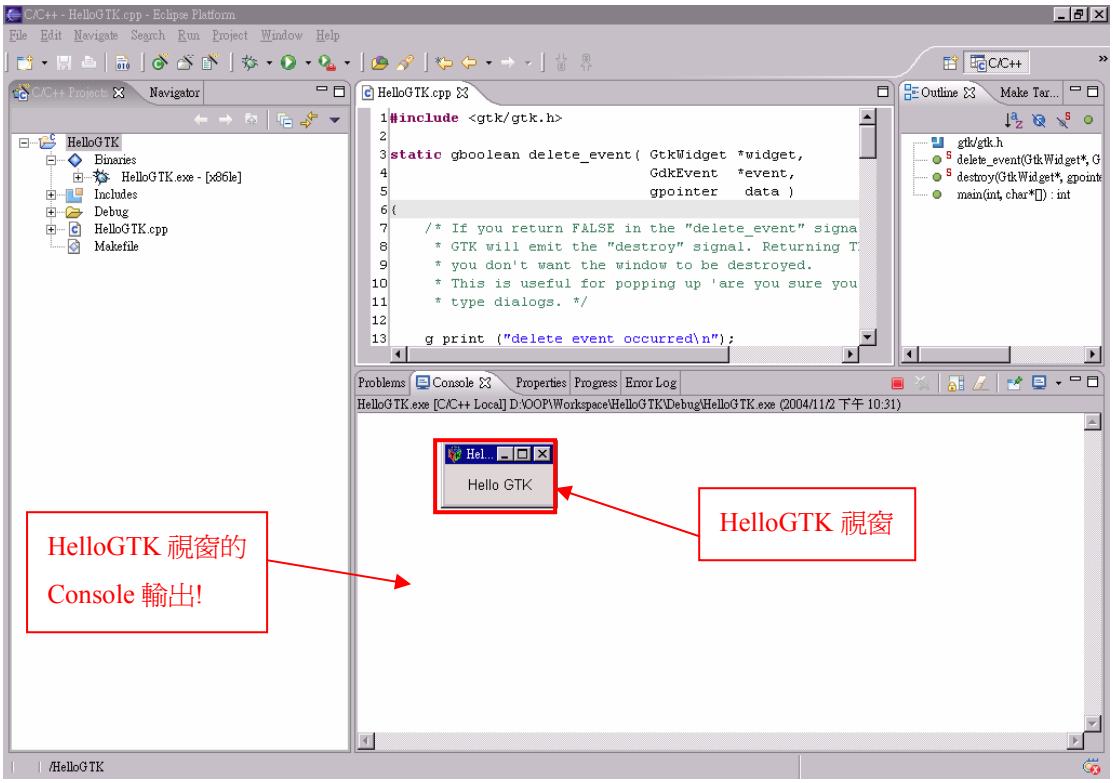


圖 5：執行畫面

備註

1. 以 GTK+撰寫程式的方式就如同傳統寫 Win32 SDK 的方式，稍有難度。而它的方式也是僅是以撰寫 C 的方式開發視窗程式，進一步的資訊得參考 GTK 所提供的 Library。

參考資料

1. http://www.gtk.org/