Cygwin/X Contributor’s Guide

Harold L Hunt, II

Jon Turney

Cygwin/X Contributor's Guide

by Harold L Hunt, II

by Jon Turney

Table of Contents

Chapter 1. Overview

The Cygwin/X project can use your help! We will do everything we can to make experienced contributors productive as soon as possible. We also want to make it as easy as possible for new contributers to make Cygwin/X their first open source project.

Anyone who despaired of touching the monolithic tree will find things much easier now with modular X. If you want to see Cygwin/X stay current and add new features, then WE NEED YOU.

We need programmers, documentation writers, and website maintainers.

Chapter 2. Programming

Overview

This chapter provides a consolidated overview of all of the information needed to begin making source code contributions to Cygwin/X. Creating a source code contribution for Cygwin/X requires an amazingly small amount of information; however, prior to this document that tiny amount of information was difficult to obtain, as it was scattered across several documents and source code files. New programmers with no open source project experience, as well as programming gurus, will be able to make source code contributions to Cygwin/X after reading this chapter. Programming gurus are great; our intention is to create more of them.

This document is primarily focused on the Cygwin/X X server; most other X.Org components are extremely stable and work out-of-the-box on Cygwin

The primary source of information on developing X is the X.Org developer startpage (http://www.x.org/wiki/DeveloperStart)

Downloading the X Window System source code tree can take anywhere from 10 minutes to 10 hours, depending upon the speed of your network connection. If you have an active network connection at your disposal you may want to skip ahead to the Section called *Obtaining the Source Code* and start downloading the source code tree now. You will find it advantageous to have a source code tree as you read the other sections.

Source Code Tree Layout

A brief overview of the `xserver` source tree layout, highlighting the parts of important and interest for Cygwin/X development:

- `dix` contains [drawing] *device independent X* routines. `main.c` contains the `main` entry-point function for the Cygwin/X X Server. The X Server startup procedure can be followed by examining `main`.
- `fb` contains the modern framebuffer drawing procedures used by Cygwin/X.
- `hw` contains [drawing] hardware dependent functions.
  - `dmx` contains the Distributed Multihead X X Server. The DMX X server acts as a proxy for multiple back-end X servers.
  - `kdrive`
    - `ephyr` contains the Xephyr X Server which uses a window on a host X Server as its framebuffer. Unlike Xnest it supports modern X extensions such as Composite, Damage, randr, etc.
    - `fake` contains the Xfake X Server. xfake is similar to xvfb, but discards all data written to the framebuffer.
  - `vfb` contains the Virtual Framebuffer X Server. The vfb server draws to a system memory framebuffer. Xvfb is primarily used for testing, or for running X clients which require an X server but there is no interest in seeing the content of it’s windows.

  - `xfree86` contains source code for the X Window System servers that run on various operating systems that generally have low-level access to the graphics hardware. Cygwin/X does not have low-level access to the graphics hardware, thus Cygwin/X is not able to utilize the X Window System server.
  - `xnest` contains source code for the Nested X Server which runs inside of another X Server.
  - `xquartz` contains the source code for the XQuartz X Server, which runs on Mac OS X.
  - `xwin` contains the source code for the Cygwin/X X Server. This is the primary directory that Cygwin/X programmers will be interested in.

- `include` contains header files specific to the X Server program, such as graphics context structures. This directory is useful to Cygwin/X programmers when they need to lookup the name or data type of members of various X Server structures.
- `mi` contains *machine independent* drawing routines. These drawing routines are used by the Cygwin/X Native GDI X Server engine. In turn, the machine independent routines depend on `winGetSpans`, `winFillSpans`, and `winSetSpans`, which are implemented in the Native GDI engine.
- `miext` contains various machine independent X extensions.
  - `shadow` contains source code for the *shadow* framebuffer layer that the Cygwin/X X Server depends upon. This directory is of primary importance to Cygwin/X, but it is maintained by other programmers and is only of direct interest to Cygwin/X programmers when it fails to build. The shadow layer does three things:
    1. Allows the `fb` graphics routines to draw to an offscreen framebuffer.
    2. Keeps track of the regions of the offscreen framebuffer that have been drawn on.
    3. Calls one of Cygwin/X's engine dependent ShadowUpdate () functions to transfer the updated regions of the offscreen framebuffer to the screen.

- `os` contains *operating system dependent* X Server functions. However, the functions in the `os` have been written in such a way that they are actually compatible with most UNIX-style operating systems, include Cygwin.

Other packages of interest in the X.Org Release

- `xorg-docs` contains documentation for various components of the X Window System. Cygwin/X-specific documentation is not contained in this directory.
- The various X Window System protocol headers.
- The various X client and X Server libraries. Cygwin/X programmers occasionally need to fix Cygwin-related build errors that occur in these libraries.
- The various standard X utility and test applications (e.g. **xeyes**, **xhost**, **xinit**, **xlogo**, etc.). Cygwin/X programmers occasionally need to fix Cygwin-related build errors that occur in these applications.
- The various font packages that contain font definition files used to compile fonts.

Cygwin/X X Server Architecture

Cygwin/X's X Server architecture was heavily inspired by Angebranndt94, the Definition of the Porting Layer for the X v11 Sample Server.

Server Privates

X Servers use various structures to pass information around to functions. Some of those structures are colormaps, *graphics contexts* (GCs), *pixmaps*, and *screens*. The X protocol defines the contents of each of these structures, however, the X Server implementation and various X Server libraries (*MI*, *FB*, *Shadow*, etc.) may require additional information to be associated with these internal structures. For example, the Cygwin/X X Server must associate a Windows window handle (hwnd) with each X Server screen that is open.

Privates are the mechanism provided by the X protocol for associating additional information with internal X Server structures. Privates originally consisted of a single pointer member contained in each structure, usually named `devPrivate` or `devPriv`. This original specification only allowed one of the X Server layers (mi, fb, shadow, etc.) to have privates associated with an internal structure. Privates have since been revised.

The current privates implementation requires that each X Server layer call a function on startup to indicate that that layer will require privates and to obtain an index into the array of privates that that layer's privates will be stored at. Modern privates are generally stored in an array of type DevUnion pointed to by a structure member named `devPrivates`; DevUnion is defined in `xserver/include/miscstruct.h`. There are two different memory allocation schemes for `devPrivates`.

Memory for privates structures can either be preallocated or allocated upon use. Preallocation, the preferred method for GCs, pixmaps, and windows, requires that the size of the privates memory needed be specified during X Server initialization. Preallocation allows the *DIX* layer to allocate all memory needed for a given internal structure, including all privates memory, as a single contiguous block of memory; this greatly reduces memory fragmentation. Allocation upon use, used by screens, requires the *DDX* structure creation function to allocate memory for the privates; `winScreenInit` calling `winAllocatePrivates`, which allocates screen privates memory directly, is an example of this. Allocation upon use can optionally and non-optimally be used by GCs, pixmaps, and windows.

Macros

Three macros are provided for each class of privates that make setting up and using the privates easier. The macros for screen privates are examined as an example.

```
winPrivScreenPtr winGetScreenPriv(ScreenPtr pScreen);
```

`winGetScreenPriv` takes a non-NULL pointer to a screen, a ScreenPtr, and returns the pointer stored in the DDX privates for that screen. Passing a NULL or invalid ScreenPtr to `winGetScreenPriv` will cause an access violation, crashing the Cygwin/X X Server.

```
void winSetScreenPriv(ScreenPtr pScreen, void * pvPrivates);
```

`winSetScreenPriv` takes a non-NULL pointer to a screen, a ScreenPtr, and sets the DDX privates pointer to the value of the *`pvPrivates`* parameter. Passing a NULL or invalid ScreenPtr to `winSetScreenPriv` will cause an access violation, crashing the Cygwin/X X Server.

```
void winScreenPriv(ScreenPtr pScreen);
```

`winScreenPriv` takes a non-NULL pointer to a screen, a ScreenPtr, and declares a local variable in the calling function named `pScreenPriv`. `winScreenPriv` may only be called at the top of a C function within the variable declaration block; calling the function elsewhere will break the ANSI C rule that all variables must be declared at the top of a scope block. Passing a NULL or invalid ScreenPtr to `winScreenPriv` will cause an access violation, crashing the Cygwin/X X Server.

Engine System

The Cygwin/X X Server uses several methods of drawing graphics on the display device; each of these different drawing methods is referred to as an engine. Each of the engines can be classified as either a Shadow FB engine, a Native GDI engine, or as a Primary FB engine. It should be noted that the Primary FB engine is deprecated and is discussed here only for completeness. The engines are discussed in the following sections, in order of importance.

Shadow FB Engines

The Shadow FB engines use Keith Packard's *FB* drawing procedures wrapped with his *Shadow* layer that allows drawing to an *offscreen framebuffer* with periodic updates of the *primary framebuffer*.

Native GDI Engine

The Native GDI engine will eventually translate individual X graphics calls into their GDI equivalent. Some X graphics calls do not translate directly to a GDI call so they may be passed through the MI layer to change them into a serious of lower level calls that are supported. Currently, the Native GDI engine passes all X graphics calls through the MI layer to convert them into three functions: `FillSpans`, `GetSpans`, and `SetSpans`. The functionality of those three functions, as of 2001-10-28, is limited to the functionality needed to draw the familiar X background pattern upon X Server startup.

Primary FB Engine

The Primary FB engine is deprecated. Primary FB works in the same manner that the original Cygwin/X X Server worked, namely, it uses `IDirectDrawSurface_Lock` to obtain a pointer to the *primary framebuffer* memory at server startup. This memory pointer is held until the X Server shuts down. This technique does not work on all versions of Windows.

Locking the primary framebuffer on Windows 95/98/Me causes the Win16Mutex to be obtained by the program that locks the primary framebuffer; the Win16Mutex is not released until the primary framebuffer is unlocked. The Win16Mutex is a semaphore introduced in Windows 95 that prevents 16 bit Windows code from being reentered by different threads or processes. For compatibility reasons, all GDI operations in Windows 95/98/Me are written in 16 bit code, thus requiring that the Win16Mutex be obtained before performing those operations. All of this leads to the following situation on Windows 95/98/Me:

1. The primary framebuffer is locked, causing the Cygwin/X X Server to hold the Win16Mutex.
2. Windows switches the Cygwin/X X Server out of the current process slot; another process is switched in.
3. The newly selected process makes a GDI function call.
4. The GDI function call must wait for the Win16Mutex to be released, but the Win16Mutex cannot be released until the Cygwin/X X Server releases the Win16Mutex. However, the Cygwin/X X Server will not release the Win16Mutex until it exits. The end result is that the Win16Mutex has been deadlocked and the Windows machine is frozen with no way to recover.

Windows NT/2000/XP do not contain any 16 bit code, so the Win16Mutex is not an issue; thus, the Primary FB engine works fine on those operating systems. However, drawing directly to the primary framebuffer suffers performance problems. For example, on some systems writing to the primary framebuffer requires doing memory reads and writes across the PCI bus which is only 32 bits wide and features a clock speed of 33 MHz, as opposed to accessing system memory, which is attached to a 64 bit wide bus that runs at between 100 and 266 (effective) MHz. Furthermore, accessing the primary framebuffer memory requires several synchronization steps that take many clock cycles to complete. The end result is that the Primary FB engine is several times slower than the Shadow FB engines.

The Primary FB engine also has several unique issues that are difficult to program around. Development of the Primary FB engine has ceased, due to the difficulty of maintaining it, coupled with the fact that Primary FB does not run on Windows 95/98/Me and with the poor performance of Primary FB. The Primary FB source code has been left in place so that future programmers can enable it and see the poor performance of the engine for themselves.

User Input

At the end of `InitInput` in `hw/xwin/InitInput.c` we open `/dev/windows`, a special device which becomes ready when there is anything to read on the windows message queue, and add that to the select mask for `WaitForSomething` using `AddEnabledDevice`.

The X server's main loop calls the ***OS*** layer `os/WaitFor.c`'s `WaitForSomething` function, which waits for something to happen using `select`. When `select` returns, all the wakeup handlers are run. Any queued Win32 user input messages (as well as other Win32 messages) are handled when `hw/xwin/winwakeup.c`'s `winWakeupHandler` function is called. Each Win32 user input message typically queues an input event, or several input events, using the ***MI*** layer's `mi/mieq.c`'s `mieqEnqueue` function.

Enqueued MI input events are processed when the ***DIX*** layer `dix/dispatch.c`'s `Dispatch` function calls `hw/xwin/InitInput.c`'s `ProcessInputEvents` function, which calls `mi/mieq.c`'s `mieqProcessInputEvents`.

Keyboard

Win32 keyboard messages are processed in `winwndproc.c`'s `winWindowProc`. The messages processed are:

- WM_SYSKEYDOWN
- WM_KEYDOWN
- WM_SYSKEYUP
- WM_KEYUP

The WM_SYSKEY* messages are generated when the user presses a key while holding down the **Alt** key or when the user presses a key after pressing and releasing the **F10** key. Processing for WM_SYSKEYDOWN and WM_KEYDOWN (and respectively WM_SYSKEYUP and WM_KEYUP) messages are identical because the X Server does not distinguish between a normal key press and a key press when the **Alt** key is down.

Win32 uses virtual key codes to identify which key is being pressed or released. Virtual key codes follow the idea that the same virtual key code will be sent for keys with the same label printed on them. For example, the left and right **Ctrl** keys both generate the VK_CONTROL virtual key code. Virtual key codes are accompanied by other state information, such as the extended flag, that distinguishes between the multiple keys with the same label. For example, the left **Ctrl** key does not have the extended flag asserted, while the right **Ctrl** key does have the extended flag asserted. However, virtual key codes are not the way that key presses have traditionally been identified on personal computers and in the X Protocol.

Personal computers and the X Protocol use scan codes to identify which key is being pressed. Each key on the keyboard generates a specified number when that key is pressed or released; this number is called the scan code. Scan codes are always distinct for distinct keys. For example, the left and right **Ctrl** keys generate distinct scan codes, even though their functionality is the same. Scan codes do not have additional state information, as the multiple keys with the same label will each generate a unique scan code. There is some debate as to which of virtual key codes or scan codes is the better system.

The X Protocol expects that keyboard input will be based on a scan code system. There are two methods of sending a scan codes from a virtual key code message. The first method is to create a static table that links the normal and extended state of each virtual key code to a scan code. This method seems valid, but the method does not work reliably for users with non-U.S. keyboard layouts. The second method simply pulls the scan code out of the *`lParam`* of the keyboard messages; this method works reliably for non-U.S. keyboard layouts. However, there are further concerns for non-U.S. keyboard layouts.

Non-U.S. keyboard layouts typically use the right **Alt** key as an alternate shift key to access an additional row of symbols from the **‘**, **1**, **2**, ..., **0** keys, as well as accented forms of standard alphabetic characters, such as á, ä, å, ú and additional alphabetic characters, such as ß. Non-U.S. keyboards typically label the right **Alt** key as **AltGr** or **AltLang**; the Gr is short for “grave”, which is the name of one of the accent symbols. The X Protocol and Win32 methods of handling the **AltGr** key are not directly compatible with one another.

The X Protocol handles **AltGr** presses and releases in much the same way as any other key press and release. Win32, however, generates a fake **Ctrl** press and release for each **AltGr** press and release. The X Protocol does not expect this fake **Ctrl** press and release, so care must be taken to discard the fake **Ctrl** press and release. Fake **Ctrl** presses and releases are detected and discarded by passing each keyboard message to `winkeybd.c`'s `winIsFakeCtrl_L` function. `winIsFakeCtrl_L` detects the fake key presses and releases by comparing the timestamps of the **AltGr** message with the timestamp of any preceding or trailing **Ctrl** message. Two real key events will never have the same timestamp, but the fake key events have the same timestamp as the **AltGr** messages, so the fake messages can be easily identified.

Special keyboard considerations must be handled when the Cygwin/X X Server loses or gains the keyboard focus. For example, the user can switch out of Cygwin/X, toggle the **Num Lock** key, then switch back into Cygwin/X; in this case Cygwin/X would not have received the **Num Lock** toggle message, so it will continue to function as if **Num Lock** was in its previous state. Thus, the state of any mode keys such as **Num Lock**, **Caps Lock**, **Scroll Lock**, and **Kana Lock** must be stored upon loss of keyboard focus; on regaining focus, the stored state of each mode key must then be compared to that key's current state, toggling the key if its state has changed.

Mouse

Win32 mouse messages are processed in `winwndproc.c`'s `winWindowProc`. The messages processed are:

- WM_MOUSEMOVE
- WM_NCMOUSEMOVE
- WM_LBUTTON*
- WM_MBUTTON*
- WM_RBUTTON*
- WM_MOUSEWHEEL

Handling mouse motion is relatively straight forward, with the special consideration that the Windows mouse cursor must be hidden when the mouse is moving over the client area of a Cygwin/X window; the Windows mouse cursor must be redisplayed when the mouse is moving over the non-client area of a Cygwin/X window. Win32 sends the absolute coordinates of the mouse, so we call `miPointerAbsoluteCursor` to change the position of the mouse.

Three-button mouse emulation is supported for users that do not have a three button mouse. When three-button mouse emulation is disabled, mouse button presses and releases are handled trivially in `winmouse.c`'s `winMouseButtonsHandle` by simply passing the event to `mieqEnqueue`. Three-button mouse emulation is quite complicated.

Three-button mouse emulation is handled by starting a timer when the left or right mouse buttons are pressed; the button event is sent as a left or right mouse button event if the other button is not pressed before the timer expires. The button event is sent as an emulated middle button event if the other mouse button is pressed before the timer runs out.

The mouse wheel is handled in `winmouse.c`'s `winMouseWheel` by generating sequences of button 4 and button 5 presses and releases corresponding to how much the mouse wheel has moved. Win32 uses variable resolution for the mouse wheel and passes the mouse wheel motion as a delta from the wheel's previous position. The number of button clicks to send is determined by dividing the wheel delta by the distance that is considered by Win32 to be one unit of motion for the mouse wheel; any remainder of the wheel delta must be preserved and added to the next mouse wheel message.

Other Windows messages

Certain other WM_ messages are also processed. TBD.

Prerequisites for Building the Source Code

Required Packages for Building

Many developer libraries and developer tools are required to build Cygwin/X. Several packages are required in addition to the default packages installed by the Cygwin installer. Following is a list of additional packages that are required to compile Cygwin/X natively in Cygwin. Note that some of these packages are meta packages that will automatically cause several other packages to be selected for installation; do not unselect any of these automatically selected packages.

- Required tools:

  autoconf, automake, binutils, bison, bzip2, cygport, diffutils, fileutils, findutils, flex, gawk, gcc, git, libtool, make, patch, pkg-config, python3, sed, tar

- Required protocol headers:

  bigreqsproto, compositeproto, damageproto, dmxproto, fixesproto, fontsproto, glproto, inputproto, kbproto, presentproto, randrproto, recordproto, renderproto, resourceproto, scrnsaverproto, windowswmproto, xcmiscproto, xextproto, xf86bigfontproto xineramaproto, xproto,

- Required development libraries:

  libdmx-devel, libfontenc-devel, libfreetype-devel, libGL-devel, libpixman1-devel, libX11-devel, libXRes-devel, libXau-devel, libXaw-devel, libXdmcp-devel, libXext-devel, libXfont-devel, libXi-devel, libXinerama-devel, libXmu-devel, libXpm-devel, libXrender-devel, libXtst-devel, libxcb-icccm-devel, libxcb-image-devel, libxcb-keysyms-devel, libxcb-shape-devel, libxcb-util-devel, libxkbfile-devel, zlib

- Other miscellaneous required packages :

  dri-drivers, font-util, khronos-opengl-registry, python3-lxml, xorg-util-macros, xtrans

> **Note:** The /usr/share/doc/Cygwin/xorg-server-n.n.n.README file installed with the X server binary package lists up-to-date runtime and build requirements.

> **Tip:** Use **setup -q -P**`packagename,packagename,etc.` to quickly install the required packages.

Compilation environment setup

libXfont linkage issue

> **Note:** As of libXfont 1.5 and X server 1.16, the linkage issue discussed here should be resolved. This section is retained for information only.

libXfont must be statically linked for the server to start correctly, otherwise it fails with errors loading all fonts, including the mandatory fixed font.

This is due to limitations of Cygwin's current weak symbol handling, which requires static linking to work correctly. For example, in libXfont the RegisterFPEFunctions function is defined weak and should be overloaded with RegisterFPEFunctions defined in dix/dixfont.c in the xserver. However, such overloading will only currently work for Cygwin when libXfont is statically linked with the X server, and not as a shared libary.

If Cygwin's libXfont-devel package contains a shared library stub, /usr/lib/libXFont.dll.a, when building the X server, you must prepare your compilation environment so that the libXfont shared libary stub is not linked with.

A quick and dirty way of achieving this is to move libXFont.la and libXfont.dll.a aside whilst building the X server.

```
mv /usr/lib/libXFont.la    /usr/lib/libXFont.la.old
mv /usr/lib/libXFont.dll.a /usr/lib/libXFont.dll.a.old
```

A cleaner way is to generate a customized .pc file for libXfont and arrange for that to be in your PKG_CONFIG_PATH when ./configuring the X server. For example:

```
sed -e 's| -lXfont| /usr/lib/libXfont.a|' /usr/lib/pkgconfig/xfont.pc > ~/xfont.pc
export PKG_CONFIG_PATH=~:$PKG_CONFIG_PATH
```

Obtaining the Source Code

Obtaining via Cygwin setup

The source code for the packages distributed via Cygwin setup is also available via Cygwin setup. To install the source for the X server, run Cygwin setup and tick the 'Src?' check-box for the 'xorg-server' package.

This may have multiple patches applied on top of the upstream X.Org X Window System source code, and is known to build and function on Cygwin, so this should be the starting point for new developers.

On installing the source code package, setup will unpack it under /usr/src. You should find the source archive, any needed .patch files, and a .cygport file which automates the distribution configuration, build and packaging tasks.

> **Note:** Due to the large number of patches applied to the upstream source, the current source package contains a source archive prepared directly from a git repository containing those patches on top of the upstream source, rather than containing the upstream source archive and many separate patches.

The sources can unpacked and prepared using cygport as follows:

```
Username@CygwinHost ~
$ cd /usr/src

Username@CygwinHost /usr/src
$ cygport xorg-server-n.n.n-n.cygport prep
[lots of output as archive is unpacked and patches applied]

$ cygport xorg-server-n.n.n-n.cygport compile
[lots of output as source is configured and built]

Username@CygwinHost /usr/src
$ cd xorg-server-n.n.n-n/src/xorg-server-n.n.n/
[navigate to the source directory]

Username@CygwinHost /usr/src/xorg-server-n.n.n-n/src/xorg-server-n.n.n/
$
```

> **Note:** Alternatively you may manually untar the archive and apply any patches (in the correct order).

> **Note:** It is necessary to run the package's autogen.sh script to regenerate the configure script and Makefiles if the patches modify the autoconf or automake source files

> **Note:** For details of using cygport to generate packages for distribution, see the Section called *Packaging a Cygwin/X Distribution*

Obtaining from version control

The packaging script for the packages distributed via Cygwin setup is currently held in a git repository. Intermediate versions between released packages can be obtained from there.

```
Username@CygwinHost ~
$ git clone https://github.com/jon-turney/xorg-server-cygport.git
```

This will obtain a .cygport file. and any .patch files. You can then add the source archive by downloading it with cygport.

```
Username@CygwinHost ~
$ cygport xorg-server-n.n.n-n.cygport download
```

Then proceed as in the Section called *Obtaining via Cygwin setup*

Obtaining from X.Org

Cygwin/X source code is contained in, and distributed with, the X Window System source code releases (http://xorg.freedesktop.org/wiki/Releases/Download).

Anonymous read-only access to the X Window System git source (http://cgit.freedesktop.org/xorg/xserver/) tree hosted on freedesktop.org (http://freedesktop.org/Software/xorg) is available.

```
$ git clone git://anongit.freedesktop.org/git/xorg/xserver
```

You will probably want to look at the .cygport file in the source package obtained in the Section called *Obtaining via Cygwin setup* and check you understand if you need to use the configuration options used there. For example:

```
$ ./configure --prefix=/usr --with-log-dir=/var/log
```

Consult the *git* documentation for details on using git.

The CYGWIN branch exists in git for historical reasons. Current development follows the mainline (called the *master* branch in git terminology).

If you just want to look at the Cygwin/X source, use the cgit interface to the X.Org tree (http://cgit.freedesktop.org/xorg/xserver/). Most of the Cygwin/X-specific code is in the xserver/hw/xwin (http://cgit.freedesktop.org/xorg/xserver/tree/hw/xwin) directory.

Native Compiling

Compiling the Source Code

Standard Build

Follow these steps to create a standard, non-debug, build:

Change the current directory to your X Window System development directory:

```
Username@CygwinHost ~
$ cd ~/xserver

Username@CygwinHost ~/xserver
$ ./configure --prefix=/usr --with-log-dir=/var/log
[lots of output]

Username@CygwinHost ~/xserver
$ make
[lots more output]
```

Standard build is now complete.

> **Note:** You may wish to consult the .cygport file for the current ./configure flags used in distributed packages

> **Note:** The unpacked source occupies approximately 80MB of disk space. Building the source requires approximately an additional 160MB. On my ageing 2.2MHz Athlon64 3500+, a full build takes about 20 minutes.

> **Note:** If you wish to keep build products separate from the source, you may run configure from a separate build directory.

Debug Build

Follow these steps to create a build with debugging information:

Change the current directory to your X Window System development directory:

```
Username@CygwinHost ~
$ cd ~/xserver

Username@CygwinHost ~/xserver
$ ./configure --prefix=/usr --with-log-dir=/var/log --enable-debug CFLAGS="-g -O0"
```

```
[lots of output]

Username@CygwinHost ~/xserver
$ make
[lots more output]
```

Debug build is now complete.

Running a local build

Follow these steps to run the built X server:

1. Change the current directory to your X Window System build directory:

   ```
   Username@CygwinHost ~
   $ cd xserver
   Username@CygwinHost ~/xserver
   $
   ```

2. Invoke the hw/xwin/Xwin executable:

   ```
   Username@CygwinHost ~/xserver
   $ hw/xwin/XWin
   ```

   The X server you have built will now attempt to run.

Installing a local build

Installing a local build installs the built X server(s) and associated man pages.

1. Change the current directory to your desired X Window System build directory:

   ```
   Username@CygwinHost ~
   $ cd ~/xserver/build/build-prefix

   Username@CygwinHost ~/xserver/build/build-prefix
   $
   ```

2. Make the **install** target, which installs everything:

   ```
   Username@CygwinHost ~/xserver/build/build-prefix
   $ make install
   ```

Cross Compiling

Cross compiling is the act of the building source code on one system, the build host, into executables or libraries to be run on a different host, the target host. The build host and the target host may differ in operating system and/or processor type.

Cross compiling is much trickier than building on the native host. There are a whole new class of problems that can happen when cross compiling that are simply not an issue when building on Cygwin. You should be familiar with building Cygwin/X on Cygwin, as described in the Section called *Native Compiling*, before attempting to cross compile Cygwin/X.

See Appendix A for notes on building a cross compiler.

Building Cygwin/X

Building the source code when cross compiling X Window System is nearly identical to the process described below in the Section called *Native Compiling* of the Native Compiling section. One divergence from the aforementioned instructions is that you will be using a **bash** shell on your cross compiling host, rather than on your native Cygwin host.

1. When configuring, you must pass *`--target=i686-pc-cygwin`* to **./configure** to cause the build system to build for the target, Cygwin, platform:
2. When installing a build, you must pass *`DESTDIR=/stagingdir`* to **make install** to install the target platform build into `/stagingdir`.

   > **Tip:** Never run **make install** on your host platform without the *`DESTDIR`* parameter, as that will cause the Cygwin build of X Window System to be installed over top of your local X Window System installation, which would completely destroy your host system's X Window System installation.

   OR, when configuring, you can pass *`--prefix=/stagingdir`* to **./configure** to cause the build system to be configured to install the target platform build into `/stagingdir`. This avoids the possibilty of installing over the native installation, at the cost of baking the prefix `/stagingdir` into the target binaries.

Contributing Patches

Submit patches for Cygwin/X source code and documentation to the cygwin-xfree@cygwin.com mailing list. All patches are thoughtfully considered.

Please ensure your editor of choice both understands and preserves UNIX-style end of line characters.

Please ensure patches are in unified diff format (e.g. using **diff -u**)

Packaging a Cygwin/X Distribution

Cygwin/X uses a cygport build and packaging script that automates all of the tasks required to build, create binary packages, and source code packages.

Note: These instructions assume that you want to build a distribution from the source packages available from Cygwin's setup program.

You can a similar technique to build a distribution from locally modified sources, cygwin-ports git or an X.Org release tarball instead.

1. Use Cygwin setup to install the xorg-server source package, it will be automatically unpacked under `/usr/src`
2. Invoke cygport on the .cygport file contained in the source package installed above. This will create the source and binary packages `xorg-server-n.n.n-X-src.tar.bz2` and `xorg-server-n.n.n-X.tar.bz2`

   ```
   Username@CygwinHost /usr/src
   $ cygport xorg-server-x.x.x-x.cygport all
   ```

Reference Documentation

X developer reference documentation.

X.Org documents

Offical X.Org documentation.

Print versions of various X Window System Manuals also exist.

- Definition of the Porting Layer for the X V11 Sample Server (http://www.x.org/releases/X11R7.5/doc/core/Xserver-spec.pdf)

  Essential reading.

  The current version of this document is available in the xorg-docs package.
- Xlib - C Language X Interface (http://www.x.org/docs/X11/xlib.pdf)

  The current version of this document is available in the libX11 runtime package.
- X Protocol (http://www.x.org/docs/XProtocol/proto.pdf)
- X Inter-Client Communication Conventions (http://www.x.org/docs/ICCCM/icccm.pdf)
- X Logical Font Description Conventions (http://www.x.org/docs/XLFD/xlfd.pdf)

Further reading

Other documents of interest.

- The X Selection Mechanism or, How to Cut and Paste in 1000 lines or more (http://x.cygwin.com/docs/cg/porting-docs/selection.pdf) (40 KiB, 13 pages)

1990 - Good information/tutorial about the X selection mechanism. From Papers and Talks by Keith Packard (http://keithp.com/~keithp/talks/) converted to PDF

X server porting documents

Other documents about X server porting.

- Strategies for Porting the X V11 Sample Server (http://x.cygwin.com/docs/cg/porting-docs/strat.pdf) (77 KiB, 22 pages)

  1998 - Mainly of historical interest now, but might give you some insight
- Godzilla's Guide to Porting the X V11 Sample Server (http://x.cygwin.com/docs/cg/porting-docs/gdz.pdf) (38 KiB, 11 pages)

  1990 - Old and thin, but relevant
- Design of eXcursion Version 2 for Windows, Windows NT, and Windows 95 (http://x.cygwin.com/docs/cg/porting-docs/eXcursion2.pdf) (301 KiB, 14 pages)

  1996 - Discusses some of the difficulties in creating an X Server for Microsoft Windows. More geared towards implementing a server that translates X raster ops into Windows GDI raster ops. Cygwin/X does not currently translate raster ops, though the framework to do so is in place, and development on raster op translation could be resumed in the future.
- Writing a Graphics Device Driver and DDX for the Digital UNIX X Server (http://x.cygwin.com/docs/cg/porting-docs/xikdoc.pdf) (602 KiB, 272 pages)

  1997 - Contains some good hints in the DDX section

Chapter 3. Documentation

Overview

Cygwin/X documentation is written in XML according to the DocBook (http://docbook.org/) document type definition (DTD). These XML input files are then compiled using an autoconf and automake build system. We currently build the following output formats: HTML, PDF, PS, RTF, and TXT.

Obtaining the Source Code

Source of latest cygwin-x-doc release

To obtain the source of the latest release of the cygwin-x-doc package start the cygwin setup, select directories and mirror and select the package cygwin-x-doc from the category X11. Mark the checkbox labelled src and install. This will install the documentation source in /usr/src/cygwin-x-doc.

Source from CVS

The documentation source code is available from sourceware.org CVS. To obtain them please use the follow commands:

```
$ cvs -d :pserver:anoncvs@sourceware.org:/cvs/cygwin-xfree login
CVS password: <hit return>
$ cvs -d :pserver:anoncvs@sourceware.org:/cvs/cygwin-xfree co doc
[output as files are checked out]
$ (cd doc && autoreconf)
[output as autoconfiguration scripts are regenerated]
```

You should now have the sources in an directory called doc.

If you just want to look at the Cygwin/X documentation source, use the CVSweb interface to the Cygwin/X documentation tree (http://sourceware.org/cgi-bin/cvsweb.cgi/doc/?cvsroot=cygwin-xfree).

Setting Up a DocBook Build Environment

Setup a DocBook build environment on Cygwin

Required Packages for building documentation

- openjade
- openSP
- texlive-collection-htmlxml, texlive-collection-genericrecommended, texlive-collection-latexrecommended (for jadetex)
- docbook-dsssl
- lynx (for converting html to text)
- ImageMagick (for converting .png figures to .ps)

Building the Documentation

Follow these instructions to build the Cygwin/X documentation source code:

1. Open a shell on your documentation build host; you should see a window like the following:

   ```
   Username@CygwinHost ~
   $
   ```

2. Change the current directory to the documentation source code directory:

   ```
   Username@CygwinHost ~
   $ cd cygwin-x-doc-1.0.0

   Username@CygwinHost ~/cygwin-x-doc-1.0.0
   $
   ```

3. Create a `build` directory and change the current directory to that directory:

   ```
   Username@CygwinHost ~/cygwin-x-doc-1.0.0
   $ mkdir build

   Username@CygwinHost ~/cygwin-x-doc-1.0.0
   $ cd build

   Username@CygwinHost ~/cygwin-x-doc-1.0.0/build
   $
   ```

4. Configure the documentation source code:

   ```
   Username@CygwinHost ~/cygwin-x-doc-1.0.0/build
   $ ../configure
   ```

   > **Note:** Use **./configure --enable-hardcopy** to enable building of all documentation formats, otherwise just HTML will be built

5. Build the documentation:

```
Username@CygwinHost ~/cygwin-x-doc-1.0.0/build
$ make all
```

6. Building the documentation is now complete.

Packaging a Documentation Distribution

Follow these instructions to build a Cygwin/X documentation source code distribution:

1. Edit the version tag in the third line of the file `configure.ac` to indicated a new version, or to add a branch name to the distribution. The line containing the version tag should look like:

```
AC_INIT(cygwin-x-doc, 1.0.0)
```

2. Commit this change to CVS and apply an appropriate tag:

```
Username@CygwinHost ~/cygwin-x-doc-1.0.0
$ cvs ci configure.ac
[...]
Username@CygwinHost ~/cygwin-x-doc-1.0.0
$ cvs tag -c VERSION_1_0_0
[...]
```

3. Change the current directory to the documentation source code build directory:

```
Username@CygwinHost ~/cygwin-x-doc-1.0.0
$ cd cygwin-x-doc-1.0.0/build

Username@CygwinHost ~/cygwin-x-doc-1.0.0/build
$
```

4. Build the documentation source code distribution:

```
Username@CygwinHost ~/cygwin-x-doc-1.0.0/build
$ make distcheck
```

5. The documentation source code distribution should now be contained in the current directory in a file called `cygwin-x-doc-1.0.0.tar.bz2`.
6. Building the documentation is now complete.

Chapter 4. Web Site Maintenance

The Cygwin/X web site is stored in sourceware.org CVS. The CVSROOT is :ext:sourceware.org:/cvs/cygwin/, and the path is htdocs/xfree/

Updating the documentation on the web site

A simple way of updating the documentation shown on the web site from the cywin-x-doc package is to build the documentation, then install it into a CVS checkout of web-site and then check it in.

```
$ cd path-to-website-checkout
$ export CVS_RSH=ssh
$ cvs -z9 -d :ext:user@sourceware.org:/cvs/cygwin/ co htdocs/xfree
[...]
$ cd path-to-cygwin-x-doc-checkout
$ ./configure --enable-hardcopy --with-docdir=path-to-website-checkout/htdocs/xfree/docs
$ make
$ make install
$ cd path-to-website-checkout/htdocs/xfree/docs
$ cvs ci
```

Bibliography

Articles

[Angebranndt94] *Definition of the Porting Layer for the X v11 Sample Server*, Angebranndt94, Susan Angebranndt, Raymond Drewry, Philip Karlton, Todd Newman, Robert W. Scheifler, Keith Packard, and David P. Wiggins, 1994.

Books

[ScheiflerGettys92] Robert W. Scheifler, James Gettys, Jim Flowers, and David Rosenthal, 1992, 1-55558-088-2, Butterworth-Heinemann, *X Window System: The Complete Reference to Xlib, X Protocol, ICCCM, and XLFD*.

[Richter99] Jeffrey Richter, 1999, 1-57231-996-8, Microsoft Press, *Programming Applications for Microsoft Windows: Mastering the critical building blocks of 32-bit and 64-bit Windows-based applications*.

[Petzold99] Charles Petzold, 1999, 1-57231-995-X, Microsoft Press, *Programming Windows: The definitive guide to the Win32 API*.

[McKay99] Everett N. McKay, 1999, 0-7356-0586-6, Microsoft Press, *Developing User Interfaces for Microsoft Windows: Practical and effective methods for improving the user experience*.

[JonesOhlund99] Anthony Jones and Jim Ohlund, 1999, 0-7356-0560-2, Microsoft Press, *Network Programming for Microsoft Windows: Clear, practical guide to Microsoft's networking APIs*.

[Yuan01] Feng Yuan, 2001, 0-13-086985-6, Prentice Hall PTR, *Windows Graphics Programming: Win32 GDI and DirectDraw*.

[CohenWoodring98] Aaron Cohen and Mike Woodring, 1998, 1-56592-296-4, O'Reilly & Associates, Inc., *Win32 Multithreaded Programming: Building Thread-Safe Applications*.

[CameronRosenblattRaymond96] Debra Cameron, Bill Rosenblatt, and Eric Raymond, 1996, 1991, 1-56592-152-6, O'Reilly & Associates, Inc., *Learning GNU Emacs: UNIX Text Processing*.

[Lewine91] Edited by Dale Dougherty, Donald A. Lewine, 1991, 0-937175-73-0, O'Reilly & Associates, Inc., *POSIX Programmer's Guide: Writing Portable UNIX Programs*.

[KernighanRitchie88] Brian W. Kernighan and Dennis M. Ritchie, 1988, 1978, 0-13-110370-9, Prentice Hall PTR, *The C Programming Language: ANSI C*.

Glossary

B

Bitmap (Win32)

Windows pixel map.

Bitmap (X)

X pixel map with bit depth equal to one. X pixel maps of bit depth not equal to one are called *pixmaps*.

C

Color Framebuffer Layer

Deprecated X Server layer providing implementations of the X graphics functions to draw on an antiquated framebuffer device. CFB is optimized to minimize CPU instructions at the expense of additional memory accesses; this does not work well on modern machines because memory access is the system performance bottle neck. CFB can only be initialized to draw on one depth of framebuffer per instantiation; this was done to eliminate CPU instructions that checked the current framebuffer depth, thus saving processing time on early machines.

Colormap

X Server colormap. Contains a table translating index values to red, green, blue 3-tuples that will be displayed on the screen when a given index value is contained in a bitmap.

D

Device Dependent X Layer

X Server layer that depends on the hardware; but not the operating system.

Device Independent X Layer

X Server layer that does not depend on the hardware layer, nor the operating system.

F

Framebuffer Layer

X Server layer providing implementations of the X graphics functions to draw on a modern framebuffer device. FB is optimized to minimize memory accesses at the expense of additional CPU instructions; this works well on modern machines because memory access is the system performance bottle neck.

G

Graphics Context

X Server graphics context. Stores information describing a graphics operation to perform, such as the foreground and background colors, fill style, stipple, and tile.

git

git is an open source distributed version control system. More information can be found at the git project homepage (http://git.or.cz/).

M

Machine Independent Layer

X Server layer providing user input and graphics display functions that are independent of the machine used by the DDX layer. The MI drawing functions depend on only three DDX functions: `FillSpans`, `GetSpans`, and `SetSpans`.

O

Offscreen Framebuffer

Essentially a *bitmap*, in the Windows sense, of size and color format that can be displayed on the screen. An offscreen framebuffer may be identical in size and color format to the *primary framebuffer*, but this is not always required.

OS Layer

X Server layer that depends on the operating system; but not the hardware.

P

Pixmap

X pixel map with bit depth not equal to one. X pixel maps of bit depth one are called *bitmaps*.

Primary Framebuffer

The block of memory, essentially a *bitmap*, that describes what is currently being displayed on the screen. Any updates to the primary framebuffer will be displayed on the screen after the next screen refresh.

Privates

Additional information associated with internal X Server structures, such as *colormaps*, *GCs*, *pixmaps*, or *screens*.

S

Screen

X Server screen. A screen usually corresponds to a display device; however, Cygwin/X's X Server corresponds each screen to one Windows window. A single instance of the Cygwin/X X Server may have several screens.

Shadow

X Server shadow layer that allows *FB* to draw to an offscreen framebuffer and occasionally call a *DDX* function that transfers the updated regions to the screen.

X

X Display Manager

An X Display Manager presents a graphical login screen to X users. Often an XDM will allow the user to select a desktop environment or window manager to be for their login session. Some X Display Managers are xdm, gdm (Gnome Display Manager), and kdm (KDE Display Manager).

X Display Manager Control Protocol

XDMCP allows XDM to process logins for users remote to the machine that XDM is running on; login sessions will be run on the machine running XDM. For example, at a university you may use XDMCP to login to an X session running on an engineering department computer from your dorm room.

See Also: X Display Manager.

Appendix A. Building a cross-compiler

Here be dragons: Building a cross-compiler is not often tested. You will likely encounter problems following these instructions, and require an in-depth understanding of what you are doing in order to fix those problems. These instructions are meant to be a template to be completed with your own understanding, rather than a recipe to be followed blindly. Reports to the mailing list that you followed these instructions, got an error message and are now stuck, will be ignored with harsh, uncaring indifference.

Note: Even once you have built your cross-compiler, there are a whole new class of problems that can happen when cross-compiling that are simply not an issue when building on the target. For example: you will need to somehow make the dependencies of your program available to the cross-compiler, hardly anything uses HOST_EXEEXT correctly, etc.

This mailing list thread (http://cygwin.com/ml/cygwin/2010-09/msg00194.html) discusses some of the issues with building a cross-compiler, and provides an example script. It also discusses cross-compiling the Cygwin DLL itself.

You will want to read carefully the GCC installation guide (http://gcc.gnu.org/install/).

These instructions will assume you have chosen a suitable working directory, e.g. `~/cygwin/`

To build a minimal cross-toolchain, we need to build binutils (for the cross-assembler and cross-linker) and GCC (for the cross-compiler).

Obtaining binutils and GCC source

binutils and GCC releases that are known to work for Cygwin are distributed with source code by the Cygwin project. These may contain patches against the stock upstream release required to build or function correctly on Cygwin, therefore, it is highly recommended that you obtain the binutils and GCC sources from the Cygwin mirror network (http://cygwin.com/mirrors.html).

Follow these steps to download Cygwin sources:

1. Create a directory to store the binutils and GCC sources in, such as `~/cygwin/src/`
2. The URL listed for your nearest mirror site should take you to the `cygwin/` directory on the mirror
3. Download the following files from `cygwin/release/`, saving them to `~/cygwin/src/`.
   - `binutils/binutils-2.20.51-2-src.tar.bz2`
   - `gcc4/gcc4-4.5.0-1-src.tar.bz2`

```
user@crosshost ~ $ mkdir -p ~/cygwin/src
user@crosshost ~ $ cd ~/cygwin/src
user@crosshost ~/cygwin/src $ wget $YOUR_MIRROR/release/binutils/binutils-2.20.51-2-src.tar.
```

```
user@crosshost ~/cygwin/src $ wget $YOUR_MIRROR/release/gcc4/gcc4-4.5.0-1-src.tar.bz2
```

Currently GCC is provided as cygport package. You will want to examine closely the .cygport file contained in the source package which shows how GCC is configured and built natively.

You will want to read carefully the cygwin-specific READMEs installed by the corresponding binary packages into `/usr/share/doc/Cygwin/`, which contain important information and build instructions

Obtaining Cygwin headers and libraries

The usual technique for building GCC cross-compilers is to:

1. build binutils
2. build a bootstrap compiler (--without-headers --enable-languages=c) that will only be used to build the C runtime library.
3. use the bootstrap compiler to build the C runtime library.
4. rebuild the final compiler, including internal libraries that need the target-specific C runtime library in order to be compiled properly.

Unfortunately, this technique cannot be applied building a cross-compiler for Cygwin, not least due to the use of C++ code in `winsup/`.

The simplest method of escaping from this chicken-and-egg situation is to make the Cygwin headers and libraries available at the time of building the cross-compiler, by installing them from the Cygwin binary packages containing those headers and libraries.

Headers and libraries from the following packages are required:

- cygwin (needed for building libgcc)
- win32api (needed for building libgcc)
- iconv (needed for building libstdc++)

```
user@crosshost ~/cygwin/src $ wget $YOUR_MIRROR/release/cygwin/cygwin-1.7.7-1.tar.bz2
user@crosshost ~/cygwin/src $ wget $YOUR_MIRROR/release/w32api/w32api-3.15-1.tar.bz2
user@crosshost ~/cygwin/src $ wget $YOUR_MIRROR/release/libiconv/libiconv-1.13-10.tar.bz2
user@crosshost ~/cygwin/src $ cd ~/cygwin
user@crosshost ~/cygwin $ tar xjf src/cygwin-1.7.7-1.tar.bz2 usr/include usr/lib
user@crosshost ~/cygwin $ tar xjf src/w32api-3.15-1.tar.bz2 usr/include usr/lib
user@crosshost ~/cygwin $ tar xjf src/libiconv-1.13-10.tar.bz2 usr/include usr/lib
user@crosshost ~/cygwin $ ln -s ../usr/include/ i686-pc-cygwin/include
user@crosshost ~/cygwin $ ln -s ../usr/lib/ i686-pc-cygwin/lib
```

Alternatively, these headers and libraries can be obtained by copying the contents of the `/usr/lib` directory and `/usr/include` directory of a Cygwin host, to the `~/cygwin/i686-pc-cygwin` directory on your build host. *Ensure the method you use to copy these files preserves symlinks.*

Building binutils

1. Change the current directory to the `~/cygwin/src` directory:

```
user@crosshost ~/cygwin $ cd ~/cygwin/src/
```

2. Extract the binutils archive:

```
user@crosshost ~/cygwin/src $ tar jxf binutils-2.20.51-2-src.tar.bz2
```

3. Create a `~/cygwin/build/binutils-2.20.51-2/` directory and change the current directory to that directory:

```
user@crosshost ~/cygwin/src $ mkdir -p ~/cygwin/build/binutils-2.20.51-2
user@crosshost ~/cygwin/src $ cd -p ~/cygwin/build/binutils-2.20.51-2
```

4. Configure binutils:

```
user@crosshost ~/cygwin/build/binutils-2.20.51-2/build $ ../../src/binutils-2.20.51-2/co
```

   - The Cygwin binutils source tarball doesn't seem to record the configuration used to build the Cygwin binary package.

5. Build binutils:

```
user@crosshost ~/cygwin/build/binutils-2.20.51-2/build $ make all 2>&1 | tee all.log
```

6. Install binutils:

```
user@crosshost ~/cygwin/build/binutils-2.20.51-2/build $ make install 2>&1 | tee install
```

7. Modify the PATH environment variable to include the directories that the binutils executables were installed in, so they are available when we build GCC:

```
user@crosshost ~/cygwin/build/binutils-2.20.51-2/build $ export PATH=~/cygwin/bin:$PATH
```

Building GCC

1. Change the current directory to the `~/cygwin/src/` directory:

```
user@crosshost ~ $ cd ~/cygwin/src
```

2. Extract the GCC archive, then extract the upstream GCC source archive and apply the patches it contains :

```
user@crosshost ~/cygwin/src $ tar jxf gcc4-4.5.0-1-src.tar.bz2
user@crosshost ~/cygwin/src $ tar jxf gcc-4.5.0.tar.bz2
user@crosshost ~/cygwin/src $ (patching and autoreconf commands omitted)
```

   - Where the patches touch the configuration mechanism, you need to regenerate the files generated by autotools. autoreconf doesn't work, I don't know why, so you need to invoke the correct autotools in the correct directories. The cygport file provides an example of how to do this.
   - GCC is picky about the exact versions of the autotools in use, so you need to make the versions it requires available.

- For bonus points, use **cygport prep** to extract the source and apply the patches.

3. Create a `~/cygwin/build/gcc-4.5.0-1/` directory and change the current directory to that directory:

   ```
   user@crosshost ~/cygwin/src $ mkdir ~/cygwin/build/gcc-4.5.0
   user@crosshost ~/cygwin/src $ cd ~/cygwin/build/gcc-4.5.0
   ```

   - It's highly recommended that GCC be built into a separate directory from the sources which does not reside within the source tree. Building GCC in the source directory is generally untested, and building into a subdirectory of the source directory is unsupported.

4. Configure GCC:

   ```
   user@crosshost ~/cygwin/build/gcc-4.5.0 $ ../../src/gcc-4.5.0/configure --prefix=/home/us
   --disable-bootstrap --enable-version-specific-runtime-libs --enable-static --enable-shar
   --disable-__cxa_atexit --disable-sjlj-exceptions --enable-languages=c,c++ --disable-symve
   2>&1 | tee configure.log
   ```

   - Use the same configure options as used in the cygport file or reported by gcc -v

5. Build GCC:

   ```
   user@crosshost ~/cygwin/build/gcc-4.5.0-1 $ make all 2>&1 | tee all.log
   ```

6. Install GCC:

   ```
   user@crosshost ~/cygwin/build/gcc-4.5.0-1 $ make install 2>&1 | install.log
   ```

Building binutils and GCC is now complete. Test your cross-compiler by checking that a 'hello world' program can be successfully compiled on your build host and run on your Cygwin target host.

Appendix B. GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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

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

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

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties — for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the

Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See Copyleft (http://www.gnu.org/copyleft/).

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright © YEAR YOUR NAME

Permission is granted to copy, distribute and/or modify this document under the
terms of the GNU Free Documentation License, Version 1.3 or any later version
published by the Free Software Foundation; with no Invariant Sections, no
Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in
the section entitled "GNU Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with… Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts
being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Colophon

This document was produced from DocBook (http://docbook.org/) source XML using OpenJade (http://openjade.sourceforge.net/) and the DocBook DSSSL Stylesheets.

The production process for this instance of this document was started at 2014-10-10 19:45 .