

How to distribute a GTK+ application on Windows?

Asked 6 years, 1 month ago Modified 3 years, 4 months ago



23



I have installed GTK+ (specifically GTK3) and copied the GTK+ dlls to my application directory. Which dlls are required for GTK+ installation. Which dlls are required for GTK+ installation.

EDIT: The GTK+ documentation now covers themes on Windows. (Although it doesn't mention GTK3 — for that see the answers below.)

[windows](#) [gtk](#) [gtk3](#) [msys2](#)

Share Improve this question Follow

edited May 9, 2020 at 4:36

asked Mar 4, 2018 at 6:51



[bradrn](#)

8,427

2

23

52

- I usually used a combination of `ntldd` and trial and error. You can test it by setting your PATH to be empty and then running your app. But these days I use static compilation in my own build system so no DLLs are needed. – [David Grayson](#) Mar 6, 2018 at 22:50

4 Answers

Sorted by: Highest score (default)



15



It turns out that running `ldd mygtkapp.exe` (with the `ldd` provided with MinGW) gave me a listing of all the dlls required to let it run. To get only the dlls which were gtk dependencies (and not e.g. Win32 dlls) I used the following command: `ldd mygtkapp.exe | sed -n 's/\([^\]*\)\ => \\/mingw.*\/\1/p' | sort`. My program used the Haskell bindings, so the dependencies might be a bit different, but this is what I got:



```
libatk-1.0-0.dll
libbz2-1.dll
libcairo-2.dll
libcairo-gobject-2.dll
libepoxy-0.dll
libexpat-1.dll
libffi-6.dll
libfontconfig-1.dll
libfreetype-6.dll
libgcc_s_seh-1.dll
libgdk_pixbuf-2.0-0.dll
libgdk-3-0.dll
libgio-2.0-0.dll
libglib-2.0-0.dll
```

```

libgmodule-2.0-0.dll
libgobject-2.0-0.dll
libgraphite2.dll
libgthread-2.0-0.dll
libgtk-3-0.dll
libharfbuzz-0.dll
libiconv-2.dll
libintl-8.dll
libpango-1.0-0.dll
libpangocairo-1.0-0.dll
libpangoft2-1.0-0.dll
libpangowin32-1.0-0.dll
libpcre-1.dll
libpixman-1-0.dll
libpixman-1-0.dll
libpng16-16.dll
libstdc++-6.dll
libwinpthread-1.dll
zlib1.dll

```

Note also that there are a couple of other things you need to do to make a completely standalone application, particularly if you're using stock icons; for more details on this, see <https://stackoverflow.com/a/34673860/7345298>. Note however that I needed to copy the 16x16 directory instead of the scalable directory.

EDIT: I've actually found the following command to be very useful as well: `ldd mygtkapp.exe | grep '\mingw.*\.dll' -o | xargs -I{} cp "{}" ..` This command actually copies the dlls to the current directory obviating the need to laboriously do it yourself.

Share Improve this answer Follow

edited Apr 20, 2019 at 5:12

answered May 2, 2018 at 8:53



bradrn

8,427 2 23 52

How do you package the themes and icons ? – Lothar Jul 7, 2019 at 22:15

@Lothar Generally, I just follow stackoverflow.com/questions/26738025/... and gtk.org/download/windows.php#Vcpkg (look at section "Building and distributing your application"). But on the whole, it's hard to know what is the 'correct' approach — you'll probably have to experiment a bit before you find something that works. – bradrn Jul 7, 2019 at 23:43

@Lothar This comment thread is getting very old (nearly a year since my last comment), but I see that the GTK documentation now contains [instructions](#) for packaging the themes and icons. – bradrn May 9, 2020 at 4:37

I see this approach used quite often, like [here](#) but for some reason ldd mygtkapp.exe doesn't list any gtk dll, only windows ones, like `ntdll.dll => /c/WINDOWS/SYSTEM32/ntdll.dll (0x7ffb1c5f0000)`
`ntdll.dll => /c/Windows/SysWOW64/ntdll.dll (0x77b50000)` `wow64.dll =>`
`/c/WINDOWS/System32/wow64.dll (0x7ffb1a820000)` `wow64win.dll =>`
`/c/WINDOWS/System32/wow64win.dll (0x7ffb1c270000)` – Jack Nov 28, 2020 at 3:30 ✎

@Jack did you ever find out why this was happening? – muszeo Jul 21, 2021 at 22:24



13

You have some hints on the [Windows page of the GTK website](#). This is the section named [Building and distributing your application](#). It features a blog post about [distributing a GTK application on Windows](#).



The solution proposed there is to create a MSYS2 package for your application, and then install it and all its dependencies (GTK among them) in a specific directory, so that you can redistribute the whole package.



2020-12-09 EDIT:

Reading the other answers, I want to add that this method not only gets the dependencies for shared objects and binaries right, but that should also work with other kinds of resources (images, help files, etc.) that are in the required packages, as well as shared objects loaded at runtime with `dlopen`-based functions. This is something you can't get with just calling `ldd` to find the dependencies.

Share Improve this answer Follow

edited Dec 9, 2020 at 15:13

answered Mar 5, 2018 at 9:35



liberforce

11.4k 39 48

-
- 4 No, you're mixing things up. Tools like msi tools will package files together to produce an msi package. They assume you know which files you want to ship. In the OP question, he doesn't know what to ship. You may use msitools, or wix, or nsis or whatever, as a second step, but first you have to identify what you will distribute, and that's what my answer is about. – [liberforce](#) Mar 6, 2018 at 10:33

I hope people don't start with this idiocracy of packaging msys2 for end user apps. The only way is to learn which files are required. – [Lothar](#) Oct 9, 2018 at 1:58

-
- 4 @Lothar: You missed the point. The developer creates the msys2 package for her application. Then she can use pacman, the package manager provided by msys2, to install her application *and its dependencies* in a separate directory. So you let the dependency solver do the work of finding the files that are needed. This is what you package for the end user. And as an additional bonus, people using msys2 can easily install your package too. – [liberforce](#) Oct 9, 2018 at 7:50

So how can i hide msys2 inside an msi installer? That's not the way it's supposed to be. No pacman and no other package manager. You are a nerd and not a windows administrator. People simply will not accept this - and for good reasons. – [Lothar](#) Oct 11, 2018 at 10:33

-
- 5 Could you please calm down and talk technical instead of calling me a nerd? – [liberforce](#) Oct 11, 2018 at 11:00



2

The following procedure can be used to obtain the necessary DLLs:

1. Download [Listdlls](#)
2. Leave your application running
3. Open the PowerShell window where Listdlls.exe is located





4. Use the command `./Listdlls.exe application_name.exe`



5. Copy all listed paths to a text file

6. Delete all lines that contain "C:\WINDOWS*" in your text file

7. Leave only the lines that contain "C:\msys64*...*dll" in your text file

8. Open Msys2 Shell

9. Use `$ cp "paste_all_paths_from_dlls" "destination_path"`

Cautions before using the `$ cp` command. In your text file, change "\" to "/" in the paths and remove line breaks.

[Share](#) [Improve this answer](#) [Follow](#)

answered Jan 30, 2020 at 12:37



[Edson Pacholok](#)

21 3



0



```
ldd kangaroo.exe | grep '/mingw64/bin/*.dll' -o | xargs -I{} cp "{}" .
```

[Share](#) [Improve this answer](#) [Follow](#)

answered Jul 6, 2019 at 5:23



[Andy Tao](#)

313 2 7

2 This is pretty much what I wrote in my own answer — the only difference is that you use `grep -o` where I use `sed`. — [bradrm](#) Jul 6, 2019 at 10:16