What’s the best way to distribute a binary application for Linux?

Asked 13 years, 3 months ago Modified 2 years, 7 months ago Viewed 18k times

38

I just finished porting an application from Windows into Linux.
I have to create an installer of the application.
The application is **not** open source => I should distribute the application's binaries (executable file, couple .so files, help files and images).

I found several methods to do it:
- RPM and DEB packages;
- installer in .sh files;
- Autopackage.

I don't like first method (RPM and DEB packages) because I don't want to mantain different packages for different Linux distros.

What is the **best way** to distribute a binary application for Linux?

linux binary package software-distribution

Share Edit Follow

edited May 23, 2017 at 11:46
Community Bot
1 1

asked Oct 5, 2009 at 23:56

Dmitriy
3,265 7 44 55

10 Answers

Sorted by: Highest score (default)

28

Having been through this a couple of times with commercial products, I think the very best answer is to use the native installer for each supported platform. Anything else produces an unpleasant experience for the end-user, and in practice you have to test on every platform you want to support anyway, so it's not really a significant burden to maintain packages for each. The idea that you can create a binary that can "just work" on every platform out there, including some you've never even heard of, just really doesn't work all that well.

My recommendation is that you pick a platform or two to support initially (Red Hat and Ubuntu would be my suggestions) and then let user demand drive the creation of additional installation packages. Perhaps make it known that you're willing to support additional platforms, for a modest fee that covers your time and effort in packaging and testing on that platform. If a platform proves to be very different, you may need to charge more for ongoing support.

Oh, and I cannot overemphasize the value of virtual machines for scenarios like this. You need to build VMs for each platform you support, and perhaps multiple VMs per platform to make it easy to test different configurations.

Share Edit Follow

answered Oct 6, 2009 at 17:52

divegeek
4,615 2 22 28

5

There were a lot of good answers (mine included :)) here. Although that is more about binary compatibility (which you do need to worry about).

For installer I would recommend autopackage (we successfully released several versions of our software with it), they did the "installer.sh" part already and more (desktop integration for example).

You have to be careful and test your upgrade scenarios and stuff, depending on how complex you package structure is, but it is pretty neat overall. I fixed few bugs with dependency handling in 1.2.6, so it should be fine.

UPDATE: The original question was deleted, so reposting full answer here, ignore all references to autopackage, that was merged into Listaller, not sure if relevant parts survived.

For standard libraries (like crypto++, pthreads, etc) that are likely to be available in a distribution -- link dynamically and tell users to get them from their distro repository. Or link statically if it is feasible.

For weird libraries that you must control version of (if you want to deploy Qt4 app on territory of enemy gnomes for example), compile them yourself and install into a private spot only your app knows about.

Never install private libs into standard places unless you can be sure to not interfere with package systems of all distros you support. (and that they can't interfere with you either).

Use rpath instead of LD_LIBRARY_PATH, and set it properly for all you binaries and all dlls that reference each other. You can set rpath on you binary to "$ORIGIN;$ORIGIN/../lib;/opt/my/private/libs" and have linker search those places before any standard paths. (have to setsome linker flag for origin to work I think). Make sure to set rpath on your libs too: for example QtGui needs QtCore, and if user happens to install standard package with different version, you absolutely don't want it picked up (exe -> ../lib/QtGui.so (4.4.3) -> /usr/local/lib/QtCore.so (4.4.2) -- a sure way to die early).

If you compile with any rpath, you can change it later with chrpath, thus making it possible to tweak install location as part of post processing or install script.

Maintain binary compatibility. GLIB_C is pretty much static for your users, so you should link against some sufficiently old version. 2.3 is a safe bet. You can use APBuild -- a gcc wrapper that enforces GLIB_C version and does few other binary compatibility tricks, so you don't have to compile all you apps on a really old distro.

If you link to anything statically, it generally will have to be rebuilt with APBuild too, otherwise it is bound to drag newer GLIB_C symbols. All .so's you install privately will naturally have to be built with it too. Sometimes you have to patch third party libs to use older symbols. (I had to patch ruby to return real permissions instead of effective ones, since there is no such functions in older GLIB_C. Still not sure if I broke anything :)).

For integration with desktop environments (file associations, mime-types, icons, start menu entries, etc) use xdg-utils. Beware though, like everything on linux they don't really like spaces in filenames :). Make sure to test those things on each target distro -- xdg implementations are riddled with bugs and quirks.

For actual install you can either provide variety of native packages (rpm, deb and a few more), or roll out your own installer, or find installer that works on all distros bypassing native package managers. We successfully used Autopackage (same people who made APbuild) for that.

Share Edit Follow

edited Jan 21, 2019 at 22:10

answered Oct 6, 2009 at 0:42

Eugene

7,121 1 30 36

Question was deleted as too broad, I pasted the answer @OrestesKappa – Eugene Jan 21, 2019 at 22:11

3

Create a .tar.bz2 archive with the binary, then publish a feed for it, like this:

```
<?xml version="1.0" ?>
<interface uri="http://mysite/myprog.xml"
           xmlns="http://zero-install.sourceforge.net/2004/injector/interface">
  <name>MyProgram</name>
  <summary>what it does</summary>
  <description>A longer description goes here.</description>

  <implementation main='bin/myprog'
                  id="sha1new=THEDIGEST"
                  version='1.0'>
    <archive href='http://mysite/myprogram-1.0.tar.bz2'
             size='10000'/>
  </implementation>
</interface>
```

Sign it with your GPG key. You can use the tools on 0install.net to calculate the digest and add the GPG signature for you in the correct format.

Then, put it on your web-site at the address in the *uri* attribute. Any user on most Linux distributions (e.g. Ubuntu, Fedora, Debian, Gentoo, ArchLinux, etc) can then install and run your program with:

```
0launch http://mysite/myprog.xml
```

Their system will also check for updates periodically. There are various GUIs for the different desktop environments, but the command-line will work everywhere.

Also look at some of the existing feeds for inspiration.

Share Edit Follow

answered May 3, 2010 at 9:53

Thomas Leonard

6,998 2 36 40

3

It's possible to install an RPM on Debian and an APT on RHEL.

If you are going to statically link this program, or dynamically link only with libraries that you will be distributing in the package, then it doesn't much matter how you distribute it. The simplest way is tar.gz and that would work.

OTOH if it is dynamically linked with system libraries, and particularly if it has dependencies on dynamic libraries that will be shared with the client's other applications, then you kind of need to do either RPM, APT, or both.

Share Edit Follow

answered Oct 6, 2009 at 0:57

DigitalRoss

142k 24 242 328

A .deb, not an "APT" (which stands for A Package Tool) – Pascal Thivent Oct 6, 2009 at 1:30

Ok. The Wikipedia entry sure is wrong, then. Is "Advanced Packaging Tool" the official, but never-used name? en.wikipedia.org/wiki/Advanced_Packaging_Tool – DigitalRoss Oct 6, 2009 at 3:47

My bad, it's indeed "Advanced Packaging Tool" (should have checked that). Nevertheless, APT is **not** a packaging format, it's a package management system to work with Debian's `.deb` packages. – Pascal Thivent Oct 6, 2009 at 18:00

3

You may want to try out InstallBuilder. It is crossplatform (runs on Windows, Linux, Mac OS X, Solaris and nearly any other Unix platform out there). It is used by Intel, Motorola, GitHub, MySQL, Nokia/Trolltech and many other companies so you will be in good company :) In addition to binary installers, it can also create cross-distro RPMs and DEB packages.

InstallBuilder is commercial, but we offer free licenses for open source programs and very significant discounts for mISVs or solo-developers, just drop us a line.

Share Edit Follow

edited Oct 7, 2009 at 10:36

Peter Mortensen

30.8k 21 104 125

answered Oct 6, 2009 at 13:38

Daniel Lopez

3,302 2 29 29

2

There is no **best way** (universally speaking). tar.gz the binaries, that should work.

Share Edit Follow

edited Oct 7, 2009 at 10:52

Peter Mortensen

30.8k 21 104 125

answered Oct 6, 2009 at 0:01

Federico klez Culloca

25.5k 17 59 95

What about creating shortcuts to the application executable? – Dmitriy Oct 6, 2009 at 0:25

That depends as well. for what desktop environment ? (although the guys at freedesktop.org should have a standard for that) – Stefano Borini Oct 6, 2009 at 1:03

"What about creating shortcuts to the application executable?" You'd better ask the user first. I don't like programs mucking around with my toys uninvited. – dmckee --- ex-moderator kitten Oct 6, 2009 at 1:15

Today, I would also look at Snapcraft and Flatpak which are embraced by some popular distributions. I explored other options and it is what ended up working best for me. Flatpak in particular also helped me

2

learn about standard Linux desktop conventions to follow.

Share Edit Follow

answered May 18, 2020 at 0:28

Zorg

968 6 9

2

I tell you an additional possibility, although I am not aware of its status: the Loki installer. Loki was a company doing videogames porting for Linux. It went down in 2002, but the installer is available.

InstallShield is also available for linux. No idea on the status though.

Although many people are proposing you to go with tar.gz, please don't. I assume you want to provide a pleasant experience for the installation procedure to your users. A tar.gz is one of the most low level, low quality, low usability choices you can do. It works everywhere because it does basically nothing, as you know.

The guys at freedesktop.org and the LSB are quite clear on where to put stuff. What you need is a friendly program to do that. Autopackage imho has the numbers (I love it), but despite its age, I haven't seen a single program out there distributed as an autopackage.

Evaluate it carefully, but don't skip the chance of being part of the momentum in favour of it, just because it's not popular. If it works for you, and it works for your users, everything else does not matter.

Share Edit Follow

edited Oct 6, 2009 at 1:10

answered Oct 6, 2009 at 1:02

Stefano Borini

135k 95 293 421

InstallBuilder is an up-to-date alternative to InstallAnywhere installbuilder.bitrock.com – Daniel Lopez Oct 6, 2009 at 13:32

1

You may also want to look at AppImage (https://appimage.org/). The concept is that it produces a single binary file that the user downloads, sets executable, and runs directly; no installation necessary, no dependencies to install (since the app image typically includes all the dependencies except basic stuff like glibc). This makes for a really great user experience!

Some downsides:

- The image may be large, since it probably includes all files/libraries/... the app depends on.
- As the image creator, you're responsible for security updates to any of the libraries you add into your image.
- An AppImage is great for a user-run application that's pretty isolated from anything else on the system (i.e. daemons, system configuration, etc.), but if your app relies on things like udev integration, desktop file installation, dbus registration, etc. this isn't easy, since the apps files aren't available when the app isn't running (making udev rules hard), and there is by definition no installer that gets run (making desktop file installation hard).

Share Edit Follow

answered Feb 20, 2020 at 18:02

Stephen Warren

230 1 8

0

I've also looked into this at work and I'd have to agree there really isn't a "best way". If your application is being distributed as source then I'd go with the make/configure methods packaged up in a tar.gz. That seems fairly universal in the Linux world.

A good way to get an idea of what to do is to look at larger organziation and see how they distribute their binaries.

Share Edit Follow

edited Oct 7, 2009 at 10:51

Peter Mortensen

30.8k 21 104 125

answered Oct 6, 2009 at 0:26

snowballhg

368 1 10
