   各种安装包打包发布工具（安装制作工具）评测

   作为程序员，经常遇到需要打包自己劳动成果的时候，这也算是广大码工辛苦奔波之后的杀青之作，收尾之笔。因此，安装包制作的好坏也直接关系到整个工程项目的质量。

0.WinRAR-这个软件HAWARK就不多解释了，对付小型的非专业性的程序封装，想必那是极好，简单、快捷、高效，十分具体方法，请在百度里搜索 “ WinRar制作安装程序 ” 这几个关键字，定有度娘亲自指点！

我倒是用过360压缩制作安装包

1.Install Shield XXX-安装制作工具中的元老级软件，也是标准的恐龙级软件，动不动就上百兆的身躯，不愧是安装制作工具中的大哥大（不过太大的东西，有些人总觉得不舒服）！什么功能都有，连Linux和WindowsMobile下的安装都可以做，MSI、EXE、CAB等都可以通通搞定，属于全能型大将。网上破解版的也挺多的，down下来就能用，操作上相对专业一些，不过也不算太复杂，汉化版的也早出来了，上手不难。一句话：大型软件，专业操作，又大又强！

2.Advanced Installer-弱弱的问一句，我可以复制么？不管了，直接复制了：安装制作工具中的元老级软件，也是标准的恐龙级软件，动不动就上百兆的身躯，不愧是安装制作工具中的巨无霸！什么功能都有，连Linux和WindowsMobile下的安装都可以做，MSI、EXE、CAB等都可以通通搞定，属于全能型大将。网上破解版的也挺多的，down下来就能用，最新版的是V10.0，这个软件操作上相对专业一些，不过也不算太复杂，上手比Install Shield还简单。一句话：大型软件，专业操作，又大又强！

elf是linux下可执行文件的格式，对应windows中的PE格式，并不是一种打包方式。

linux下的打包方式有很多：

有依赖方式：sh脚本方式、deb安装包、rpm（Red-Hat Package Manager）安装包

无依赖打包形式：Flatpak、Snap、AppImage(下面有例子)、sfpkg

或者让用户自己去编译安装

终端下 使用ldd命令查看需要的连接库。之后自己编写shell程序，在其中增加环境变量路径，加上自己的so文件目录。

Linux打包：（不推荐，一般都会比较大）

如果是linux下，可以使用Qt Installer Framework(可以制作在线安装)，或者使用其余打包软件。相比较Windows，linux可以直接使用ldd命令查看相关的依赖。但是因为linux存在过多有差异的发行版，linux真打包起来需要很多很多so文件，于是最终打包文件一般都会很大。

无依赖打包形式：Flatpak、Snap、AppImage(下面有例子)、sfpkg

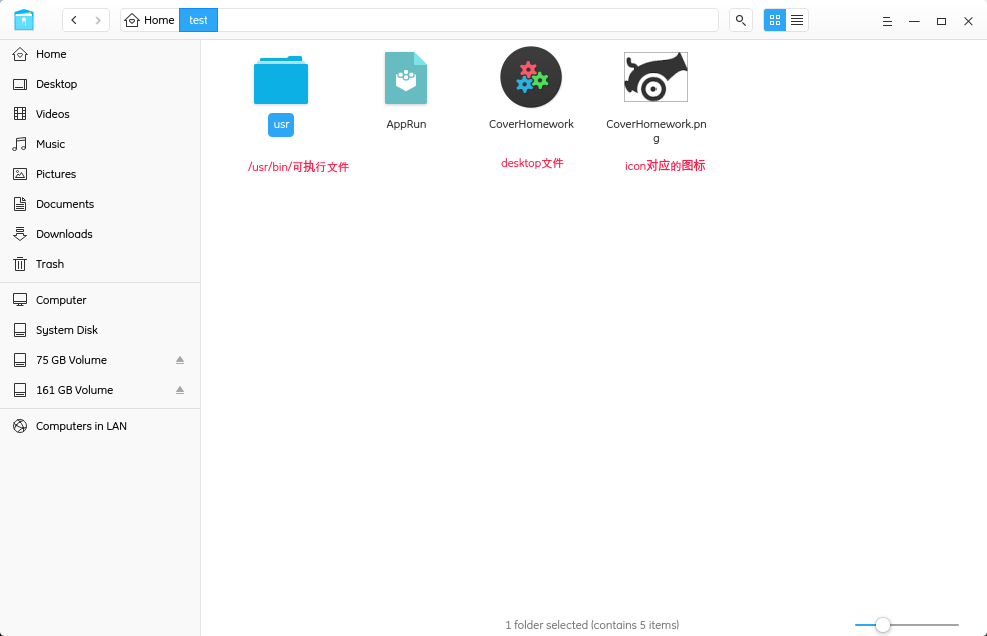
AppImage将所有依赖打包，解决的是依赖问题；Flatpak提供独立环境，各种各样的runtime（比如deepin自己就弄了一个deepin runtime，没能解决发行版过多带来的花式依赖问题，不过runtime目前才800+MB，一种缩小版系统的感觉），因为有独立环境，可以将runtime中缺少的依赖直接打包到最后的文件中，解决的是依赖冲突问题。

比较： https://github.com/AppImage/AppImageKit/wiki/Similar-projects#comparison

AppImage打包：

appimage其实就是一个压缩包,打包了所有的依赖,运行的时候就解压缩然后运行在里面。

appimagetool打包应用，appimaged注册和取消注册应用于系统daemon服务...

不过qt使用的是https://github.com/probonopd/linuxdeployqt，创建一个目录，二进制文件放到./bin下，qml放到/qml下，linuxdeployqt name -appimage就开始打包，缺少的目录与文件会自动默认创建。下面的文件是必须的：

so文件缺少要安装对应版本库：虽然qt中没有用到这个库，缺少不影响运行，但是linuxdeployqt借助ldd递归检查依赖的时候，要求一个也不能少（一个可以提升的空间）：

ERROR: ldd outputLine: "libpq.so.5 => not found"

ERROR: for binary: "/opt/Qt5.10.1/5.10.1/gcc\_64/plugins/sqldrivers/libqsqlpsql.so"

ERROR: Please ensure that all libraries can be found by ldd. Aborting.

上面的错误意为打包目标依赖的libqsqlpsql.so缺少依赖的libpq.so.5

libmysqlclient.so.18=>libmysqlclient18

libpq.so.5=>libpq5

安装了对应库以后，so文件就会出现在/usr/lib/x86\_64-linux-gnu下

一开始我没有打包正确，得到的AppImage有150MB，文件也是不可执行的。后来重新打包，Appimage才25MB。（可能是去除了不必要的依赖）

flatpak(xdg-app)打包： <http://docs.flatpak.org/en/latest/getting-started.html>

（中文介绍）<https://blog.csdn.net/beyond_zhangna/article/details/79774771>

<https://blog.csdn.net/qq_32768743/article/details/80024622> （About Qt按钮还提供了查看当前环境中qt版本信息）

sudo apt install flatpak flatpak-builder

sudo apt install deepin-fprt-com.deepin.platform deepin-fprt-com.deepin.sdk

关键概念：运行时、基准运行时（规定运行flatpak的发行版必须有哪些软件和实现了哪些标准）、SDK、绑定库（Bundle Libraries）、沙盒（SandBox）、Portals、存储库(remote)、缓存（结果文件夹删除以后，同样的命令会迅速创建，在当前目录下的.flatpak-builder/cache/refs/heads）(~/.local/share/flatpak/repo是一个同样文件夹结构的文件夹)（～/.cache里面虽然有flatpak的位置但是为空）。

flatpak应用要求每一个应用程序指定runtime，使用它的依赖。有freedesktop、gnome、kde、deepin等几种选择。flatpak remote-add --from gnome <https://sdk.gnome.org/gnome.flatpakrepo>像GTK的话，可以选择gnome，QT的话可以选择KDE（基于freedesktop，增加了qt、kde结构），因为他们里面自带了gui库，当然直接选择freedesktop，然后手动加入gui库也是可以的。

deepin全家桶是基于bundle的deb文件。

安装环境：环境有runtime（以Platform结尾）、打包环境（SDK）

deepin：

sudo apt install flatpak flatpak-builder deepin-fprt-com.deepin.platform-15.5 deepin-fprt-com.deepin.sdk-15.5

kde： <https://community.kde.org/Guidelines_and_HOWTOs/Flatpak>

应用命名方式与Android一样，作为容器中的ID，这个ID可能还会被用于D-BUS，所以名字7-zip应该命名org.\_7\_zip.xxxx。

应用要放入仓库（remote），而对仓库的任何提交（修改、删除、创建仓库）都需要-gpg-sign=<key>或者--no-gpg-verify。

finish-args 描述应用沙箱权限

--share=network - access the network

--socket=x11 - show windows using X11

--share=ipc - share IPC namespace with the host (necessary for X11)

--socket=wayland - show windows with Wayland

--device=dri - OpenGL rendering

--socket=pulseaudio - play sound with PulseAudio

Access to the entire bus with --socket=system-bus or --socket=session-bus should be avoided, unless the application is a development tool.

--filesystem=host - access normal files on the host, not including host os or system internals described below



--filesystem=home - access the user’s home directory

--filesystem=/path/path - access specific paths

--filesystem=xdg-download - access a specific XDG folder

--persist=path - map paths from the user’s home directory into the sandbox filesystem

--device=all can be used to access devices like controllers or webcams.

cleanup 相关属性，用于指定清除在build过程中产生的中间产物

modules 应用程序模块，相关库、依赖模块，比如qt.json

{

"name": "qt",

"disabled": true,

"buildsystem": "simple",

"build-commands": [

"./configure -prefix /app -confirm-license -opensource -nomake examples -no-angle -no-opengl -no-sql-sqlite -no-sql-sqlite -ssl -openssl-linked",

"make -j2 install"

],

"sources": [

{

"type": "archive",

"url": "https://download.qt.io/archive/qt/5.8/5.8.0/submodules/qtbase-opensource-src-5.8.0.tar.xz",

"sha256": "c17111ae02a44dc7be1ec2cf979a47ee9e58edf4904041a525c21f4fa53fc005"

}

]

}

flatpak-builder --repo=<repo> <build-dir> <manifest> 希望导入仓库

flatpak-builder --install <build-dir> <manifest> 不考虑仓库

modules类型：

archive - .tar or .zip archive files

git - Git repositories

bzr - Bazaar repositories

file - local file (these are copied into the source directory)

script - an array of shell commands (these are put in a shellscript file)

shell -an array of shell commands that are run during source extraction

patch - a patch (are applied to the source directory)

extra-data - data that can be downloaded at install time; this can include archive or package files

大多数flatpak命令默认在系统范围内执行。要仅为当前用户执行命令,请使用--user选项。

flatpak-builder选项介绍：

--force-clean build目标文件夹如果已经存在，则会被删除；

--disable-cache关闭缓存;

--keep-build-dirs 保留过程中文件，在当前文件夹下的.flatpak-builder，具体目录在构建的一开始有讲;

flatpak list -d --app --runtime 显示已经安装的App、Runtime

删除org.gnome.Platform/x86\_64/3.20：flatpak --user uninstall org.gnome.Platform 3.20（不加上版本号应该会移除所有版本）

flatpak remotes

flatpak remote-add --if-not-exists flathub https://dl.flathub.org/repo/flathub.flatpakrepo

flatpak remote-delete flathub

QT flatpak：

flatpak --user remote-add --no-gpg-verify --if-not-exists explosivebattery explosivebattery 建立仓库

pro文件中需要：

isEmpty(PREFIX){

PREFIX = /usr

}

target.path = $${PREFIX}/bin/

INSTALLS += target

或者直接：

target.path = /app/bin/

INSTALLS += target

Mainfest文件：

{

"app-id":"cn.hellovega.qt.CoverHomework",

"runtime":"com.deepin.Platform",

"runtime-version":"15.5.2",

"sdk":"com.deepin.Sdk",

"command":"CoverHomework",

"finish-args":[

"--socket=x11",

"--share=network"

],

"modules":[

{

"name":"CoverHomework",

"no-autogen":true,

"sources":[

{

"type":"git",

"url":"file:///home/vega/test/CoverHomework",

"branch":"master"

},

{

"type":"script",

"commands":[

"sed -i 's|/usr|/app|g' CoverHomework.pro",

"qmake PREFIX=/app"

],

"dest-filename":"configure"

}

]

}

]

}

flatpak-builder --repo=explosivebattery --force-clean 结果文件夹 Mainfest.jso 效果是在当前目录下产生一个explosivebattery文件夹，算是ref，install的时候用到

flatpak --user install explosivebattery cn.hellovega.qt.CoverHomework 安装

flatpak run cn.hellovega.qt.CoverHomework 运行

flatpak build-bundle explosivebattery cn.hellovega.qt.CoverHomework 打包

最后打包出来的因为经过压缩，仅仅400+KB（QT直接产生Release可执行文件有900+KB）。

给用户安装App的时候可以编写一个flatpakref文件，从某一个的remote添加指定名字的应用程序：

[Flatpak Ref]

Title=qBittorrent

Name=org.qBittorrent.qbittorrent

Branch=stable

Url=https://chocobo1.github.io/qBittorrent-flatpak/ Url参数似乎没有什么用

IsRuntime=False

RuntimeRepo=https://flathub.org/repo/flathub.flatpakrepo

GPGKey=xxx

Install:

flatpak install --from https://raw.githubusercontent.com/Chocobo1/qBittorrent-flatpak/master/qBittorrent.flatpakref --assumeyes

Run:

flatpak run org.qBittorrent.qbittorrent

Update:

flatpak update --app org.qBittorrent.qbittorrent

Remove:

flatpak uninstall org.qBittorrent.qbittorrent

减小打包体积：

1. 可执行文件与动态共享库文件SO中都有符号表，可以移除来瘦身，同时我们还希望万一出事了，比如coredump了，我们能获取更多的信息，这时候我们又希望有符号表。https://blog.csdn.net/bingqingsuimeng/article/details/50522242
2. ldd -u xxx 列出不使用的so文件