

How to turn an SDL_bgi program to an AppImage

AppImages are GNU/Linux programs that run on any GNU/Linux distro (if they're packaged correctly!) An AppImage is single file containing the program binary, libraries, and other resources; it can be made executable and run without installation. Please see <https://appimage.org/>

This document explains how to turn an `SDL_bgi` program to an AppImage. We will use the sample program `fern.c`, included in the `demo/` directory in the `SDL_bgi` sources. It assumes that `SDL_bgi` has been installed.

The procedure described below has been tested on fresh Ubuntu 18.04 and 20.04 systems. AppImages created on either platform also worked on the other; however, AppImages built on Ubuntu 18.04 are much smaller.

A tutorial is available [here](#) but, IMHO, it needs some improvement.

Software

Assuming we are building on the `x86_64` architecture, we need two programs:

- `appimagetool-x86_64.AppImage`

available here: <https://github.com/AppImage/AppImageKit/releases>

- `appimage-builder`

available here: <https://github.com/AppImageCrafters/appimage-builder>

First of all, install some required software:

```
$ sudo apt install -y python3-pip python3-setuptools patchelf \
desktop-file-utils libgdk-pixbuf2.0-dev fakeroot strace fuse \
python3-imagesize gtk-update-icon-cache
```

Install `appimagetool` somewhere in your `$PATH`; I suggest that you install in `$HOME/.local/bin`. Make sure that this directory is included in your `$PATH`.

```
$ wget https://github.com/AppImage/AppImageKit/\
releases/download/continuous/appimagetool-x86_64.AppImage \
-O $HOME/.local/bin/appimagetool
```

Install `appimage-builder` using `pip3`. Required Python packages will be installed in `$HOME/.local/lib/python3.8`; `appimage-builder` will be installed in `$HOME/.local/bin`.

```
$ pip3 install --user appimage-builder
```

All necessary software is now installed, and we are ready to build the AppImage.

Making the AppImage

We need three files: the program executable, an icon, and the `libSDL_bgi.so` library.

Create the `AppDir` directory tree, which will contain the files. If this directory tree already exists, remove it. This directory can be created anywhere.

```
~$ rm -rf AppDir ; \  
  mkdir -p AppDir/usr/bin \  
          AppDir/lib/x86_64 \  
          AppDir/usr/share/icons/fern/3.0.056/
```

Compile the `fern` application:

```
demo$ gcc -o fern fern.c -lSDL_bgi -lSDL2  
demo$ cp fern $HOME
```

The icon `icon.png` is available in the `SDL_bgi` sources in `doc/`:

```
doc$ cp icon.png $HOME
```

Find out the location of `libSDL_bgi.so`:

```
~$ locate libSDL_bgi.so  
/usr/lib/x86_64-linux-gnu/libSDL_bgi.so  
~$ cp /usr/lib/x86_64-linux-gnu/libSDL_bgi.so .
```

Copy the `fern` executable, `icon.png`, and `libSDL_bgi.so` to the right directories:

```
$ cp fern AppDir/usr/bin/ ; \  
  cp icon.png AppDir/usr/share/icons/fern/3.0.056/ ; \  
  cp libSDL_bgi.so AppDir/lib/x86_64/
```

Create the so-called “recipe” for the AppImage:

```
$ appimage-builder --generate  
INFO:Generator:Searching AppDir  
? ID [Eg: com.example.app]: fern  
? Application Name: Fern  
? Icon: icon  
? Executable path relative to AppDir [usr/bin/app]: usr/bin/fern  
? Arguments [Default: $@]: $@  
? Version [Eg: 1.0.0]: 1.0.0  
? Update Information [Default: guess]: guess  
? Architecture: x86_64  
...  
INFO:Generator:Recipe generation completed.
```

The application should run; if it doesn’t, the final AppImage might be defective. The `AppImageBuilder.yml` recipe file will be created.

The last stage builds the AppImage, skipping the `test` phase that only works if you have `docker` installed.

```
$ appimage-builder --skip-test --recipe AppImageBuilder.yml  
INFO:main:Running main script  
INFO:main:Running apt deploy
```

```
INFO:apt:apt-get update
```

```
...
```

```
INFO:root:AppImage created successfully
```

If everything worked as expected, you will find the newly created AppImage called **Fern-1.0.0-x86_64.AppImage**. If you plan to distribute the app, please check if it works on other Linux distros beforehand.