Packaging Applications

•••

Let’s look at a simple application called gscreenshot

gscreenshot (what and where)

- Simple screenshooter written in Python
- https://github.com/thenaterhood/gscreenshot

```
setup.py
src
└── gscreenshot
    ├── __init__.py
    └── resources
    ├── gui
    │   ├── glade
    │   │   ├── __init__.py
    │   │   └── main.glade
    │   └── __init__.py
    ├── __init__.py
    └── LICENSE

4 directories, 6 files
```

How would we install this?

- It follows the conventions for a Python application so it’s super simple!

```
sudo python setup.py install
```

What magic happened?

- Python applications are typically installed into
  - `/usr/lib/python<x>.<x>/site-packages/<app name>`
- The previous command run on my system (Archlinux, Python 3.4) installed gscreenshot to
  - `/usr/lib/python3.4/site-packages/gscreenshot`
- Since Python is not a compiled language, the installed files look just like the source code
- Python's install tools also let you define data files, so we also installed a file into /usr/bin so we can run the app from the command line

That’s not so hard, but...

- Doing it this way, our package manager won’t handle things for us
  - If we try to install it again from our package manager, it will yell at us for file conflicts
  - If we want to update or uninstall, we need to go through Python or do it manually
- This requires us to pre-emptively install Python and some other things that gscreenshot uses
  - What if we want to share the app?
- If we have a more complex app or an app that isn’t Python, it’s not quite so easy

Well that sucks, make (*apt*|*pacman*|*emerge*|*yum*|...) handle it!

Let’s talk about packaging!

What are packages?

- They’re managed by your package manager (go figure!)
- Collections of files and directories to install, and how to install them
- Usually software or libraries (also icon packs, art, etc)
- Often available through your package manager but sometimes are downloaded as .deb, .rpm, etc files

What’s in a package?

- Binaries and executables for the software, including icons, artwork, etc

- Data about the software
  - Where to put all the files (generally, the files themselves)
  - What version it is
  - In some cases, repositories the package manager can pull updates from

Basic steps for building a package

- Write (or find) an application that you want to install
- Download the application source code
- Learn about the application (what software it depends on, what version it is, license, author, etc)
- Write the metadata files for the package (these differ depending on your distribution)
- Run your distribution’s tool to build the actual package

gscreenshot cheatsheet

- Current version is 2.0.2
- Depends on
  - Python3
  - Scrot
  - pygtk
  - python-gobject
  - python-pillow
  - python-distutils

The Arch Way

- Write a PKGBUILD!
  - This is a shell script that decompresses, builds, and installs the application into a non-system directory
  - This also contains the package metadata - version, source, etc as variables
- When you build the package, Arch will download the source code, compile it, and install it to a local location, usually ./pkg, then compress it into a package

Writing the PKGBUILD (metadata)

```
pkgname=gscreenshot
pkgver=2.0.2
pkgrel=1
pkgdesc="A GUI front-end for scrot"
arch=('any')
url="https://github.com/thenaterhood/gscreenshot"
license=('GPL')
groups=()
depends=("python3"
      "python-pillow"
      "scrot"
      "gtk3"
      "python-setuptools"
      "python-gobject")
makedepends=("unzip"
      "fakeroot")
source=("https://github.com/thenaterhood/gscreenshot/archive/v$pkgver.zip")
noextract=("v$pkgver.zip")
md5sums=('fb2e4a7683b41318ff649d98e0506c6f')
```

Writing the PKGBUILD (metadata)

```
pkgname=gscreenshot
pkgver=2.0.2
pkgrel=1
pkgdesc="A GUI front-end for scrot"
arch=('any')
url="https://github.com/thenaterhood/gscreenshot"
license=('GPL')
groups=()
depends=("python3"
      "python-pillow"
      "scrot"
      "gtk3"
      "python-setuptools"
      "python-gobject")
makedepends=("unzip"
      "fakeroot")
source=("https://github.com/thenaterhood/gscreenshot/archive/v$pkgver.zip")
noextract=("v$pkgver.zip")
md5sums=('fb2e4a7683b41318ff649d98e0506c6f')
```

Application name

Application version

Package version

Description

Architecture (x86_64, x86, any, ...

Project page

Other Requirements

Requirements to build the app

Download URL

Files to not untar

Checksum(s)

There are more (optional) variables that can be set. We won’t talk about them now.

Writing the PKGBUILD (build and packaging)

```
prepare() {
     unzip $srcdir/v$pkgver.zip
}
build() {
     echo "Nothing to build"
}
package() {
     cd $srcdir/gscreenshot-$pkgver
     python setup.py install --root="$pkgdir/" --optimize=1
     chmod +x "$pkgdir/usr/bin/gscreenshot"
     install -Dm644 dist/$pkgname.desktop \
     "$pkgdir/usr/share/applications/$pkgname.desktop"
     install -Dm644 dist/black-white_2-Style-applets-screenshooter.png \
     "$pkgdir/usr/share/pixmaps/$pkgname.png"
}
```

Writing the PKGBUILD (build and packaging)

```
prepare() {
     unzip $srcdir/v$pkgver.zip
}
build() {
     echo "Nothing to build"
}
package() {
     cd $srcdir/gscreenshot-$pkgver
     python setup.py install --root="$pkgdir/" --optimize=1
     chmod +x "$pkgdir/usr/bin/gscreenshot"
     install -Dm644 dist/$pkgname.desktop \
     "$pkgdir/usr/share/applications/$pkgname.desktop"
     install -Dm644 dist/black-white_2-Style-applets-screenshooter.png \
     "$pkgdir/usr/share/pixmaps/$pkgname.png"
}
```

Prepare to build by unzipping and other pre-build tasks

Compile! gscreenshot is Python, so nothing to do here.

“Install.” This installs the application to a packaging location (not the system). Make sure to use the $pkgdir variable as a “root” or “prefix” for where to put things.

Building the package (finally!)

- Save the PKGBUILD file
- Go to a directory where the creation of some folders and files will not annoy you
- Copy the PKGBUILD
- Run ‘mkpkg’
- You should now find a file called `gscreenshot-2.0.2-1-any.pkg.tar.xz` in that directory
- `sudo pacman -U gscreenshot-2.0.2-1-any.pkg.tar.xz` will install the package

Some pointers

- When a new package version is available, you can update the version number and download URLs in the pkgbuild and follow the same steps. Running the same install command will update your install.
- You can add files to the pkgbuild that pacman will preserve (not overwrite and not delete) if there are updates to them or the software is uninstalled
- You can run unit tests in the process of building the package by adding a check() function to the pkgbuild
- There are preinstalled pkgbuild templates to give you a starting point

Wait hold on, I want to share my package!

Some basic information about Arch packages

- ArchLinux has two places you can find packages. The official repositories (packages accepted into the main distribution) and the AUR (ArchLinux User Repository). We'll submit a package to the AUR.
- The AUR is git-based, so it's as easy as git clone and git push
- AUR packages can be commented on and voted on
- If you submit a package, you are the maintainer of the package. If you no longer want to maintain the package, you can request to orphan it (orphaned packages can be adopted).
- You can submit a pkgbuild for applications that are not yours, providing the license allows you to and they are not in the main repositories
- There are packaging standards you need to follow!

Create the package on the AUR

- Create an AUR account
- Pull the package using git
  - `git clone aur@aur.archlinux.org:gscreenshot.git`
  - If the package exists, we'll get a read-only copy
  - If the package does not exist, was orphaned, or was deleted, we'll get a read-write copy
- And we've created it!
- We'll put our PKGBUILD in here. If the package existed before, we'll update the one there

Add authors to the pkgbuild

- The Arch packaging standards require us to add a comment with ourselves in it as the author.

```
# Maintainer: Nate Levesque <public at thenaterhood dot com>
```

- gscreenshot had previous maintainers, so we need to recognize them as well

```
# Contributor: TDY <tdy@archlinux.info>
# Contributor: Matej Horváth <matej.horvath@gmail.com>
```

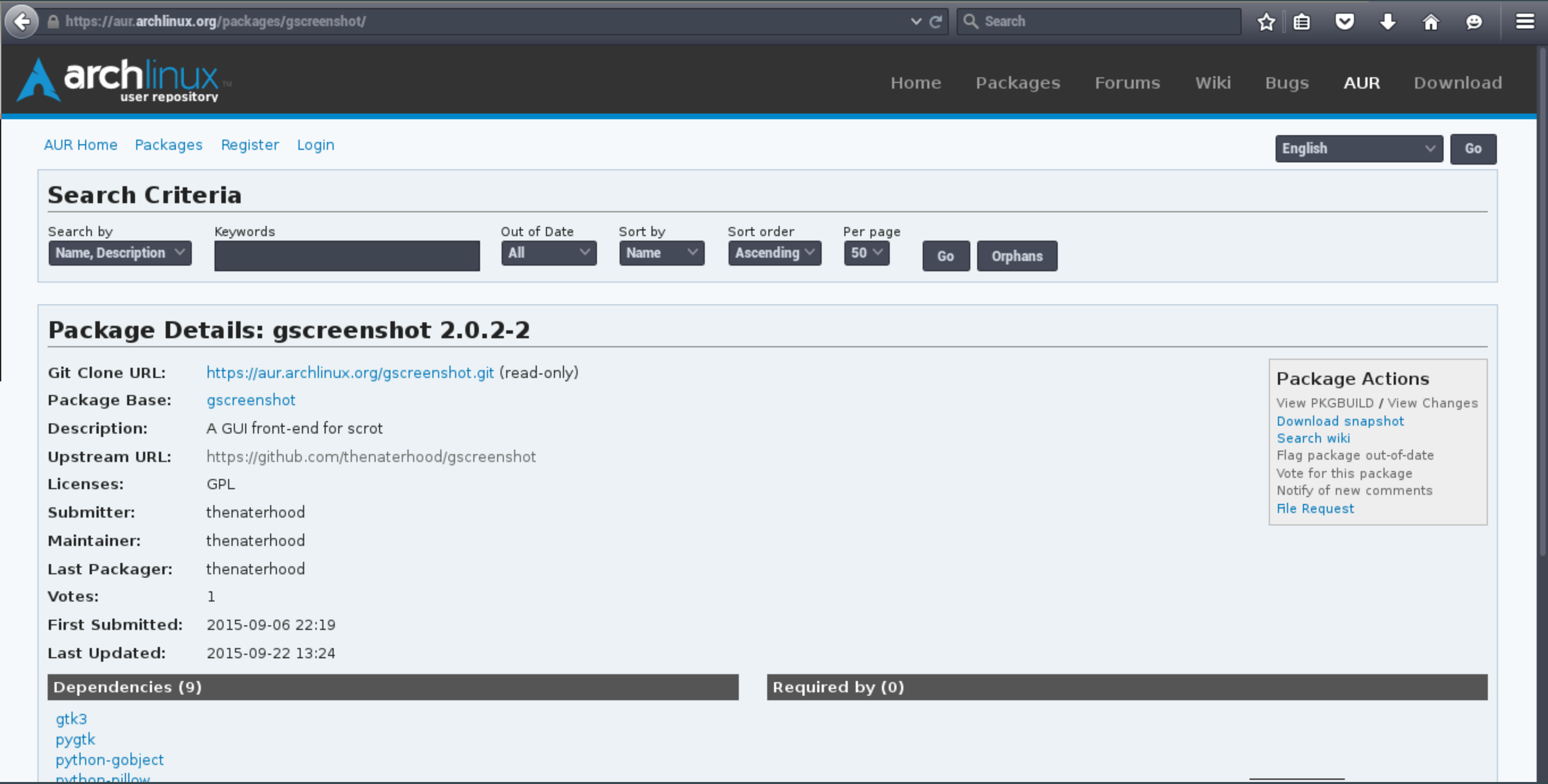
Create a .SRCINFO file

- `mksrcinfo`
- Yep. That’s super easy

Commit and push

- `git add .SRCINFO`
- `git commit -am “Add package to AUR”`
- `git push`

- And we’re done!


https://aur.archlinux.org/packages/gscreenshot/

AUR Home Packages Register Login

English Go

Search Criteria

Search by	Keywords	Out of Date	Sort by	Sort order	Per page
Name, Description		All	Name	Ascending	50

Go Orphans

Package Details: gscreenshot 2.0.2-2

Git Clone URL:	https://aur.archlinux.org/gscreenshot.git (read-only)
Package Base:	gscreenshot
Description:	A GUI front-end for scrot
Upstream URL:	https://github.com/thenaterhood/gscreenshot
Licenses:	GPL
Submitter:	thenaterhood
Maintainer:	thenaterhood
Last Packager:	thenaterhood
Votes:	1
First Submitted:	2015-09-06 22:19
Last Updated:	2015-09-22 13:24

Package Actions

View PKGBUILD / View Changes
Download snapshot
Search wiki
Flag package out-of-date
Vote for this package
Notify of new comments
File Request

Dependencies (9)

gtk3
pygtk
python-gobject

Required by (0)

Questions, comments, conundrums, concerns...