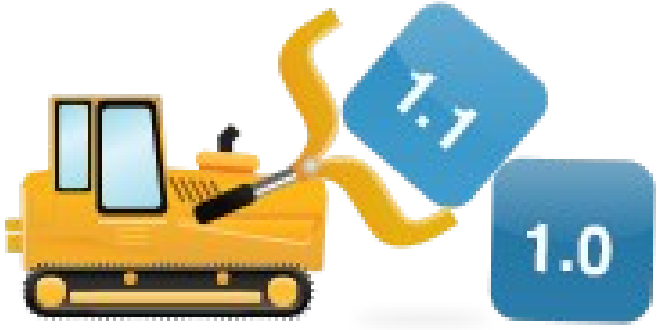
Kivy - Buildozer



Hamilton Python Users Group
Ian Stewart
9 March 2020

Kivy - Buildozer

Objectives:

- Create GUI applications using python3 on linux platform.
- Deploy the applications to an Android mobile phone.

Solution:

- Kivy - Python library for developing mobile apps.
- Buildozer - Python packager for Android and iOS.

Kivy

Kivy is a free and open source Python library for developing mobile apps and other multitouch application software with a natural user interface (NUI).

Initial release: February 1, 2011.
Stable release: 1.11.1 - 21 June 2019.
Repository: https://github.com/kivy/kivy
Website: https://kivy.org/
Docs: https://kivy.readthedocs.io/en/latest/index.html

Buildozer

Buildozer is a tool that aims to package mobile phone applications easily. It automates the entire build process, downloading the prerequisites like python-for-android, Android SDK, NDK, etc.
Buildozer manages a file named *buildozer.spec* in your application directory. It will use the specification file to create a package for Android, iOS, and more.

Initial release: December, 2012.
Stable release: 1.0 - 30 December 2019.
Website: https://buildozer.io/
Docs: https://buildozer.readthedocs.io/en/latest/

Kivy - Buildozer

Initially many things failed ...and this went on for days. So...

- Inserted a blank SSD into my laptop.
- Installed Ubuntu-Mate 19.10 (latest – for updated kivy)
- Includes Python V3.7
- Applied all updates
- No virtual environment
- Apt installed openjdk-8-jdk (Not 11, 13 or 14)
- Apt installed python3-kivy v1.10.1 (Not 1.11.1)
- Pip installed Buildozer v1.0
- Apt installed many dependencies
- Setup mobile phone to *Install unknown applications*

Kivy - Buildozer

Tutorials – Many youtube clips available. Recommend:

14 x 10 minute tutorials by Alexander Taylor
https://www.youtube.com/watch?v=F7UKmK9eQLY&list=PLdNh1e1kmiPP4YApJm8ENK2yMlwF1_edq

Videos are based on blogs in 2014. Blog still available.
For example Tutorial #1:
http://inclem.net/2014/01/09/kivy-crash-course/1_making-a-simple-app/

Code used in videos:
https://github.com/inclement/kivycrashcourse

Tutorial – Blank Screen

```
from kivy.app import App

class TutorialApp(App):
    pass

if __name__ == "__main__":
    TutorialApp().run()
```

Logging - Console

```
$ python3 kivy_1.py
[INFO   ] [Logger      ] Record log in /home/ian/.kivy/logs/kivy_20-03-0
[INFO   ] [Kivy        ] v1.10.1
[INFO   ] [Python      ] v3.7.5 (default, Nov 20 2019, 09:21:52)
[GCC 9.2.1 20191008]
[INFO   ] [Factory     ] 194 symbols loaded
[INFO   ] [Image       ] Providers: img_tex, img_dds, img_sdl2, img_pil,
[INFO   ] [Window      ] Provider: sdl2(['window_egl_rpi'] ignored)
[INFO   ] [GL          ] Using the "OpenGL" graphics system
[INFO   ] [GL          ] Backend used <gl>
[INFO   ] [GL          ] OpenGL version <b'2.1 Mesa 19.2.8'>
[INFO   ] [GL          ] OpenGL vendor <b'Intel Open Source Technology C
[INFO   ] [GL          ] OpenGL renderer <b'Mesa DRI Intel(R) 965GM '>
[INFO   ] [GL          ] OpenGL parsed version: 2, 1
[INFO   ] [GL          ] Shading version <b'1.20'>
[INFO   ] [GL          ] Texture max size <8192>
[INFO   ] [GL          ] Texture max units <16>
[INFO   ] [Window      ] auto add sdl2 input provider
[INFO   ] [Window      ] virtual keyboard not allowed, single mode, not
[INFO   ] [Base        ] Start application main loop
```

Tutorial – Coloured Button with Text

```
# A coloured button filling the screen
# Color is a tuple of rgba components

from kivy.app import App
from kivy.uix.button import Button

class TutorialApp(App):
    def build(self):
        return Button(text='Hello!',
                      background_color=(0, 0, 1, 1),
                      font_size=150)

if __name__ == "__main__":
    TutorialApp().run()
```

Tutorial – Label. Move Rotate Scaling

```
from kivy.app import App
from kivy.uix.label import Label
from kivy.uix.scatter import Scatter
from kivy.uix.floatlayout import FloatLayout

class TutorialApp(App):
    def build(self):
        f = FloatLayout()
        s = Scatter()
        l = Label(text='Hello!',
                  font_size=150)
        f.add_widget(s)
        s.add_widget(l)
        return f

if __name__ == "__main__":
    TutorialApp().run()
```

Kivy – Multitouch simulation

On Linux Desktop with traditional monitor there is no Multi-touch.

Right-click to simulate where your fingers touch the screen, then use mouse to perform moving, rotating and scaling.

If developing on a laptop then it may have a multi-touch pad that allows simulation of using you fingers on an Android screen.

Buildozer Features

Change directory to the development folder with *main.py* file

- Initialize and provide the *buildozer.spec*
- Build an Andriod application package in *bin/* folder
- Deploy the package to an Andriod phone
- Provide *logcat* application for debugging

Buildozer

- **$ buildozer init** Creates *buildozer.spec* file.
- Edit *buildozer.spec* at three places:

```
# (str) Title of your application
title = Hello 3

# (str) Package name
package.name = tutorial_3

# (str) Supported orientation (one of
# landscape, sensorLandscape, portrait or all)
orientation = all
```

Mobile Phone Settings

Samsung Galaxy J5: example

- *Settings* --> *About phone* --> *Software Information* --> *Build Number* x 7 taps.
- *Settings* now has *Developer options*
- ~~*Settings* --> *Developer options* --> *Unlock OEM* to **on**~~
- *Settings* --> *Biometrics and security* -> *Install unknown apps* --> *My Files* --> *Allow from this source* to **on**
- *Settings* --> *Developer options* --> *Select USB Configuration* --> *MTP (Media Transfer Protocol)*
- Know the ARM architecture 7 or 8, and which architecture is Android OS compiled for?

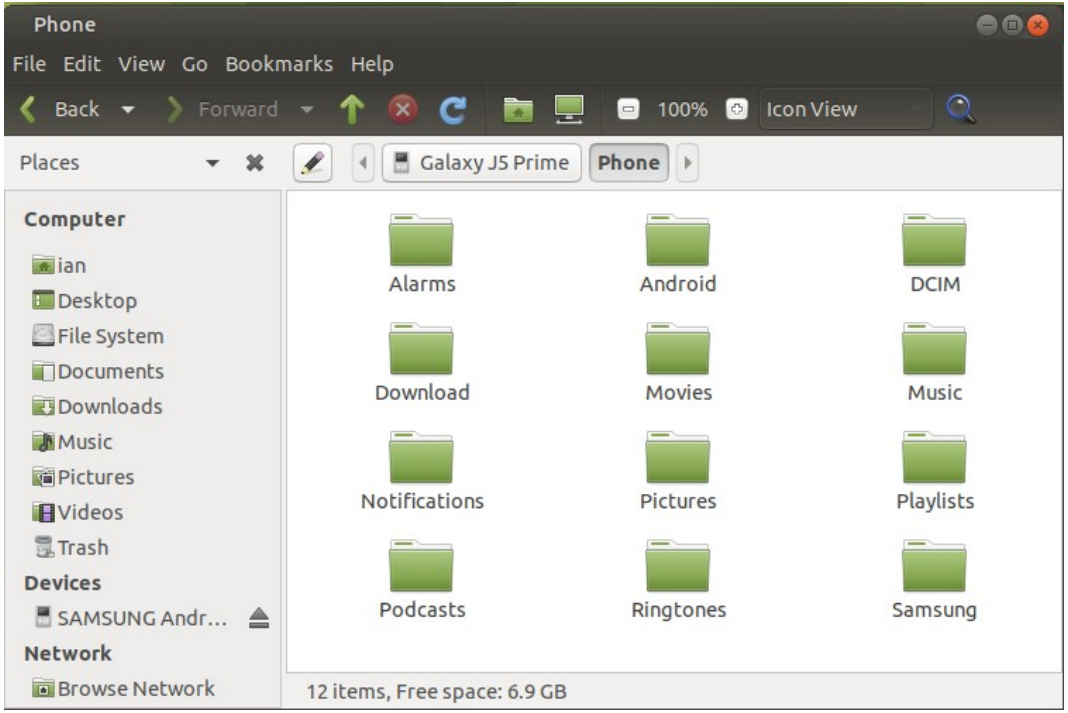
Mobile Phone

Plug Android phone into USB port on development Linux PC. Phone has pop up message:

Allow access to phone data

...click **allow**

On the PC the File Manager should now see the mobile phones folder:



Kivy – Buildozer – Development Steps

1. Plug phone into a USB port on the Linux PC.
2. Create an application development folder. **$ cd** to it
3. Write a python GUI application using kivy and name it *main.py*
4. Test the app on Linux desktop. **$ python3 main.py**
5. **$ buildozer init** – creates *buildozer.spec* file
6. Edit *buildozer.spec* file
7. **$ buildozer android debug deploy**. This builds the Andriod application package and deploys it to the phone. Takes 30 mins? A copy of the app is in the *bin/* folder
8. Click on app on the phone to launch it.

Kivy – Buildozer – Files Review

ian kivy

Name	Size	Type
bin	1 item	folder
.buildozer	3 items	folder
buildozer.spec	10.4 kB	RPM spec file
main.py	419 bytes	Python script

ian kivy bin

Name	Size	Type
myapp__armeabi-v7a-0.1-armeabi-v7a-debug.apk	12.1 MB	Android package

Name: .buildozer
Type: folder (inode/directory)
Contents: 39,461 items, totalling 1.2 GB (1.3 GB on disk)
Location: /home/ian/kivy

Name: .buildozer
Type: folder (inode/directory)
Contents: 35,721 items, totalling 4.4 GB (4.5 GB on disk)
Location: /home/ian

Kivy – Buildozer - Dependencies

```
$ sudo apt install openjdk-8-jdk
$ sudo apt install python3-kivy
$ sudo pip3 install buildozer
$ sudo apt install python3-dev
$ sudo apt install cython3
$ sudo apt install cython
$ sudo apt install python3-pip
$ sudo apt install git
$ sudo apt install automake
$ sudo apt install autoconf
$ sudo apt install libltdl-dev
$ sudo apt install libffi-dev
$ sudo apt install lld
$ sudo apt install kivy-examples
```

Buildozer – Error or Feature?

Initial actions on a folder with your main.py code are:

- $ buildozer init
- Edit *buildozer.spec* file
- $ buildozer android debug deploy <-- *30+ minutes.*

Make changes to main.py

- $ buildozer android debug deploy <-- *1 minute. OK*

In *bulldozer.spec* file change **package.name**
package.name = hello_3 to **package.name = hello_4**

- $ buildozer android debug deploy <-- *5 minutes. Huh?*

App fails on loading with *libpythonXXX.so* missing.
Can change **title** OK. E.g. **title = Hello 4**

Demos

- From the tutorials demo the code running on Linux and on Android.
- Demo a buildozer init. Look at the buildozer.spec file.
- Show the libpythonXXX.so error on Android.
- Using a .kv file working on Linux but not on Android.
- Show bouncing boxes with inline build instead of .kv file
- With the scrolling label code, change main.py to alter font size then *buildozer android debug deploy*. Takes about 1 minute.
- Using python os module to collect information on Android.
- Show Linux File Managers perspective of folders on Phone

Any Questions?

Appendix - Notes

Jargon – Todo add more...

P4a - Python for Android
NDK – Native development Kit. A set of tools tha allows you to use C and c++ code with Android.
SDL - Simple DirectMedia Layer is a crossplatform development library designed to provide low level access to audio, keyboard, mouse, joystick and graphics via OpenGL and Direct3D.

Kivy - Sister projects:

- Buildozer: generic Python packager for Android and iOS.
- Plyer: platform-independent Python wrapper for platform-dependent APIs.
- Pyjnius: dynamic access to the Java/Android API from Python.
- Pyobjus: dynamic access to the Objective-C/iOS API from Python.
- Python for Android: toolchain for building and packaging Python applications for Android.
- Kivy iOS: toolchain for building and packaging Kivy applications for iOS.
- Audiostream: library for direct access to the microphone and speaker.
- Kivy Designer: UI designer for Kivy.
- KivEnt: entity-based game engine for Kivy.
- Garden: widgets and libraries created and maintained by users.
- kivy-sdk-packager: Scripts for Kivy SDK generation on Windows, macOS and Linux.
- kivy-remote-shell: Remote SSH+Python interactive shell application.
- KivyPie: Raspbian-based distribution running latest Kivy framework on the Raspberry Pi.
- OSCPy: a fast and reliable OSC implementation.