Bookworm vs Bullseye

Table of Contents

1 Colophon

2 Introduction

This is not an in depth and fully tested guide to migrating from Bullseye to Bookworm. Rather it is a repository of the issues I have encountered and potential solutions to them.

It is assumed that the reader is familiar with the Linux command line and at least one text editor. Desktop users will need to open a terminal to execute many of the commands in this guide.

2.1 What's Not Covered

- OS installation.
- In place upgrade from Bullseye to Bookworm
- Quite a lot else, probably.

2.2 Requirements:

- An internet connection for software update and installation.
- Raspberry Pi (any model) with a configured and working network connection.
- For a headless[1] Pi, VNC or ssh enabled.

1 I.E. no monitor, mouse, or keyboard connected.

2.3 Conventions

```
Text like this indicates input to or output from the command line.
```

`Text like this` also refers to full or partial commands but is not generally intended to be entered into the command line as is.

“SD card” refers equally to full size and micro SD cards. It should also be taken to refer to any boot storage medium in use.

Where “pi” occurs as a username, replace as appropriate.

Any mention of “python” refers equally to python 2 and python 3 unless explictly specified otherwise. On RPiOS Bookworm `python` and `python3` are the same file.

All non system paths used in this guide follow the Filesystem Hierarchy Standard[2] though this is not mandatory.

2 https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

3 Fallback to zeroconf

With the switch from dhcpcd to Network Manager network connections/interfaces no longer fall back to a self assigned IP address in the link local/zeroconf subnet.[3][4] There are a number of ways to reinstate the old behaviour:

3.1 Make Network Manager Use DHCPCD

This is my preferred method as it's a one time change and works for all interfaces.

1. Update apt:

   ```
   sudo apt update
   ```

2. Install dhcpcd:

   ```
   sudo apt install -y dhcpcd
   ```

3. Open `/etc/NetworkManager/NetworkManager.conf` in your preferred text editor. You will need to be root or use sudo.

4. Add the folloowing line to the `[main]` section

   ```
   dhcp=dhcpcd
   ```

5. Save and close

6. Reboot

3 169.254.0.0/16

4 A time of writing anyway.

3.2 Use Multiple Connections For the Same Interface

Network Manager supports multiple connection profiles (hence forth connections) for a network interface. These are attempted in descending order of priority.

Making use of this feature means no additional software is required and unlike 3.1 can be configured on a per interface basis. Unlike 3.1 it must be configured for each interface where fallback is required.

Do not apply this approach to loopback interfaces (e.g. lo). This has not been tested with WiFi interfaces and will likely break things if used on those.

1. Add a connection for DHCP:

```
sudo nmcli con add con-name eth0-dhcp ifname eth0 type ethernet
   connection.autoconnect-priority 100 connection.autoconnect-retries 2
```

2. Add a lower priority connection for zeroconf:

```
sudo nmcli con add con-name eth0-zeroconf type ethernet ifname eth0
   connection.autoconnect-priority 50 ipv4.method link-local ipv4.link-
   local enabled
```

3. Make Network Manager reload its connections:

```
sudo nmcli con reload
```

4. Reboot.

Replace eth0 in the above with the name of the interface your are configuring.

Depending on configured timeout values falback to zeroconf may take longer than on Bullseye.

The shell script at https://github.com/thagrol/nm/blob/main/nm-setup.sh can assist with the above.

3.3 Other Methods

Other possible methods include creating a fallback connection in Network Manager with a static IP address in the link local subnet, configuring the interface with a static IP address in `/etc/network/interfaces`[5], using dhcpcd with a dhcp configured interface in `/etc/network/interfaces`.

5 Network Manager is set to ignore any interfaces configured here.

4 Python, Pip, and Venvs

With Bookworm the default python configuration is that modules installed through pip must be installed into a virtual environment (A.K.A a venv). There are good reasons for this change and it was imposed upstream of Raspberry Pi Ltd.

Attempt to use `pip install` or `sudo pip install` without a venv will result in the following message:

```
error: externally-managed-environment

× This environment is externally managed
╰─> To install Python packages system-wide, try apt install
    python3-xyz, where xyz is the package you are trying to
    install.

    If you wish to install a non-Debian-packaged Python package,
    create a virtual environment using python3 -m venv path/to/venv.
    Then use path/to/venv/bin/python and path/to/venv/bin/pip. Make
    sure you have python3-full installed.

    For more information visit http://rptl.io/venv

note: If you believe this is a mistake, please contact your Python installation
or OS distribution provider. You can override this, at the risk of breaking
your Python installation or OS, by passing --break-system-packages.
```

Third party installation scripts and out of date guides may also fail due to this change.

While not a tutorial on venvs the rest of this section should give you some ways to handle this change.

4.1 Does This Apply To Me?

In short, yes. But if you never install any additional python packages or only ever install packages using `apt`[6] you should never be impacted by this change.

6 e.g. `sudo apt update && sudo apt install python3-pygame -y`

4.2 Getting The Old Behaviour Back

It is possible to get the old behaviour back but all this really achieves is pushing the problem further down the line. It also makes your system different from the RPiOS default and thus makes it harder for others to help you.

This is a system wide change and applies to all users.

To disable:

```
sudo mv /usr/lib/python3.11/EXTERNALLY-MANAGED /usr/lib/python3.11/EXTERNALLY-
MANAGED.bak
```

To re-enable:

```
sudo mv /usr/lib/python3.11/EXTERNALLY-MANAGED.bak
/usr/lib/python3.11/EXTERNALLY-MANAGED
```

4.3 Temporarily Overriding The New Behaviour

Pip has the command line option `--break-system-packages` that can be used to force installation of modules in the old manner.

To install a module for the current user:

```
pip install --break-system-packages SomeModule
```

To install a module for the all users:

```
sudo pip install --break-system-packages SomeModule
```

However, this is not recommended as it makes your system different from the RPiOS default and thus makes it harder for others to help you.

4.4 Using Your venv From Another User

Python venvs can be used by users other than their owner (e.g. to run your python program from /etc/rc.local). This is what is required when running with sudo, from rc.local, or from a systemd service as root.

In the examples below the user pi has a venv at /home/pi/.venv. The user ip has no venv but wants to run pi's code which relies on pi's venv.

4.4.1 Activate The venv As The Alternate User

1. If you don't know it, find the location of the venv.
2. Activate the venv. e.g.:

```
source /home/pi/.venv/bin/activate
```

3. In the same shell or in a sub-shell of it run the python program, e.g.:

```
python /home/pi/SomeProgram.py
```

4.4.2 Use The venv Directly

1. If you don't know it find the full path to the python binary inside the venv. With the venv active `which python` or `which python3` should tell you e.g. `/home/pi/,venv/bin/python`.
2. Instead of calling the python program with the system binary call it with the one from the venv e.g.

```
/home/pi/.venv/bin/python /home/pi/SomeProgram.py
```

4.4.3 Use a Shebang Line and call the file directly

A shebang line as the first line in a python program tells the shell which python interpreter to use when the file is called directly. It is ignored by the python interpreter. In order to be useful the file must have execute permission[7].

1. If you program does not have a shebang, add one as the first line of the file e.g.:

```
#!/home/pi/.venv/bin/python
```

   If if has one change it as above.

2. Launch your program using

```
/home/pi/SomeProgram.py
```

7 `chmod +x SomeProgram.py`

4.4.4 Other Options

Other options include:

- Recreating and activating the venv under the target user.
- Creating a system wide venv usable by all users specific to the program in question.
- Creating a system wide venv usable and modifiable by all users.
- For python programs that require sudo: fix the issue that makes it require sudo and run as a normal user. This is the correct fix regardless of whether a venv is in use or not.

4.5 Cron, rc.local, and Systemd Services

Processes started under cron, rc.local, or as systemd services run under a different environment to processes started from a logged in shell. User logion scripts are not run and the actual user may not be the same as the development user.

4.5.1 Cron

In general you will need to use one of the methods in 4.4 however as each cron job runs in its own shell the following, while tempting and seemingly correct will not work:

```
0 * * * * source /home/pi/.venv/bin/activate
1 * * * * python /home/pi/SomeProgram.py
```

The environment set up in the first job is entirely separate to the environment of the second and will have been discarded prior to the second job starting.

Instead use the method from 4.4.2:

```
1 * * * * /home/pi/.venv/bin/python /home/pi/SomeProgram.py
```

or from 4.4.3:

```
1 * * * * /home/pi/SomeProgram.py
```

`@reboot` cron jobs work in the same manner.

4.5.2 rc.local

The method in 4.4.1 needs a little care in that activating the venv must not be done as a background job.

```
source /home/pi/.venv/bin/activate &
python /home/pi/SomeProgram.py &
```

will fail as it will use the wrong python interpreter.[8]

```
source /home/pi/.venv/bin/activate
python /home/pi/SomeProgram.py &
```

will work as expected.

If you wish to start code that requires a different venv, deactivate the current one and activate the new one before calling that code.

4.4.2 and 4.4.3 should work as expected

8 Background jobs are started in their own sub shell which, in this case is almost immediately discarded.

4.5.3 Systemd Services

I have yet to test venvs with systemd however I'd expect trying to activate one using

```
ExecStartPre=source /home/pi/.venv/bin/activate
```

to fail. I recommend using 4.4.2 or 4.4.3 in your ExecStart.

4.5.4 An Alternativet

Instead of trying to activate the venv and call the python program directly a simple shell script can be used instead. This also has the benefit in rc.local of restricting the activated venv to that script only.

```
#!/bin/bash
source /home/pi/.venv/bin/activate
python /home/pi/SomeProgram.py
```

Call the bash script instead of calling the python program.[9]

9 Don't forget to grant your script execute permission.

4.6 Shebangs and venvs

As mentioned above, shebangs tell the shell which python[10] interpreter to use to run the file's content.

The most common shebangs for python seem to be `#!/usr/bin/python` and `#!/usr/bin/env python` neither of which play well with venvs.

4.6.1 #!/usr/bin/python

This will always use the python binary found at /usr/bin/python bypassing any venv whether active or not. Avoid this method where ever possible.

4.6.2 #!/usr/bin/env python

This uses your current environment[11] to finf which python interpreter to run. If a vrnv is active it will be used even if it is the wrong venv. If no venv is active the system interpreter will be used.

4.6.3 An Alternative

See 4.4.3

10 Not just python. Any text file can have one. It doesn't have to match the file extension either – a .py file could have a shebang pointing to bash.

11 Via the `env` command.

5 No /var/log/syslog

Upstream Debian has removed syslog. All logging in now done through the systemd journal.

5.1 Option 1

Learn to use `journalctl`. Reading the man page would be a good place to start.[12]

5.2 Option 2

Install rsyslogd:

```
sudo apt update && sudo apt install -y rsyslog
```

Caution: this may leading to double logging and a consequent increase in disc space usage.

12 man journalctl

6 X11 vs Wayland/Wayfire

This is a significant change in Bookworm with major impact however it does not impact user of the lite images.

At time of writing Wayland is only the default on Pi 4 and Pi 5.[13] Earlier models still default to X11.

6.1 Getting The Old Way Back

The easiest way to get the old, pre-bookworm behaviour back is to switch back to X11 from Wayland:

1. Open a terminal

   ```
   sudo raspi-config
   ```

2. Select *6 Advanced Options*
3. Select *A6 Wayland*
4. Select *W1 X11*
5. Exit raspi-config and reboot

13 See https://www.raspberrypi.com/news/bookworm-the-new-version-of-raspberry-pi-os/

6.2 VNC Problems

6.2.1 Switching Between X11 and Wayland

Switching between X11 and Wayland[14] will disable the VNC server. You must enable it via `sudo raspi-config` in order to be able to connect even if it was previously enabled on the system you are switching to.

6.2.2 Client Problems

The VNC sever used under Wayland is know to cause problems with a significant number of VNC viewers. The current recommendation is to use the TigerVNC client.

14 Or between Wayland and X11

6.3 The Autostart File

`/etc/xdg/lxsession/LXDE-pi/autostart` and `/home/pi/.config/lxsession/LXDE-pi/autostart` are not used by Wayland/Wayfire. Any changes made to these will be ignored.

A similar functionality is provided in the `[autostart]` section `$HOME/.config/wayfire.ini`. Unlike `/etc/xdg/lxsession/LXDE-pi/autostart` this is a per user configuration.

The format of each line is

```
[autostart]
unique-ID = some command
```

The ID doesn't matter except that it must be unique. Commands are run with sh so if you have a shell script that relies on features of a more advanced shell either use an appropriate shebang in your script or call the required shell and pass it your script name on the command line.

`[auotstart]` is required only once at the start of the section.

An example:

```
[autostart]
terminal = lxterminal
foo = $HOME/myscript.sh
bar = bash myotherscript.sh
```

Wayfire does not appear to support the same `@` prefix feature[15] that LXDE does so you'll need to handle exit detection and restart your self, for example by wrapping the actual command in a small bash script such as this:

```
#!/bin bash
while true; do
    #your command goes here
    lxterminal
done
```

Changes made to wayfire.ini will be ignored if you later switch back to X11 from Wayland/Wayfire.

15 Automatic restart if the process exits.

6.4 The Autostart Folders

Under X11 putting a .desktop file[16] into `/etc/xdg/lxsession/LXDE-pi/autostart`, `/home/pi/.config/lxsession/LXDE-pi/autostart`, `/etc/xdg/autostart`, or `/home/pi/.config/autostart` caused the shortcut to be run at login.

`/etc/xdg/lxsession/LXDE-pi/autostart` and`/home/pi/.config/lxsession/LXDE-pi/autostart` are ignored by Wayland/Wayfire.

`/etc/xdg/autostart`, and `/home/pi/.config/autostart` are used by both systems. .desktop files placed here will be run regardless of your selection of X11 or Wayland.

16 e.g. myprogram.desktop

7 Change Log

2023-11-05 Initial release