

GPSD Installation Instructions

Eric S. Raymond – <esr@thyrsus.com>

Here are the steps for installing GPSD and verifying its performance. They assume you have GPSD available as an installable binary package.

Most of these installation instructions are generic to POSIX.

Instructions for building GPSD from source (including cross-building), and some special notes on installation on *BSD, WSL, OS X, macOS, and the Raspberry Pi are in the file "build.adoc" in the source distribution.

Check that your GPS is live and you can get data from it

Start by making sure you can get data from your GPS, otherwise the later steps will be very frustrating. In this command:

```
stty -F /dev/ttyXXX speed 4800 && cat </dev/ttyXXX
```

replace ttyXXX with the filename of the port, and 4800 with the correct speed. This will probably be either /dev/ttyUSB0 or /dev/ttyS0. If you are on a *BSD Unix or MacOS X, replace -F with -f.

When you run this command, you should see text lines beginning with \$ come to stdout (possibly after a short initial burst of binary garbage). If you don't see this, you may have OS-level problems with your serial support, but more likely have the wrong device. Look again.

If you have trouble with the preceding step, check your cabling first. Verify that the device is connected and that its power LED (if it has one) is lit.

If you seem to have some sort of serial-device problem, check that your kernel properly supports the device you are using. For GPSes using an RS-232 port (which is no longer common) you will need serial-port support compiled into your kernel. Various USB-to-serial adapter chips found in GPSes require specific drivers.

Under a stock Linux kernel these will all be loaded on demand when the USB system sees the appropriate vendor/product ID combinations. See build.adoc for instructions relating to custom kernels.

Check that your system configuration will allow GPSD to work

Ensure that device permissions will enable gpsd to read from and write to GPS devices even after it drops root privileges. If you are running Fedora Core, Ubuntu, or stock Debian you can skip this step, as the stock configuration has the right properties.

gpsd requires two things: (1) that GPS devices have group read and write enabled, and (2) that group name is compiled into the gpsd binary. On Debian and derivatives including Ubuntu this is "dialout"; on Gentoo/Fedora/openSuse it is "uucp".

Before dropping privileges, gpsd will ensure that it has access to devices given to it on the command line by forcing their group read and write permissions on.

On a Linux with udev, check the files in /etc/udev/permissions.d to ensure that /dev/tty* devices are all created with the same group and with 0660 permissions.

When gpsd drops privileges, its default is to set uid to 'nobody' and group to the owning group of the prototype device (the configure option gpsd_user=foo will cause gpsd to change to 'foo' instead).

If your system has the Linux hotplug facility installed you can skip the permission-setting part; the hotplug scripts will force the permissions for you. You still have to make sure all the tty devices are in the same group.

Check your installation prerequisites

A minimum build of GPSD can run pretty close to the metal; all it absolutely needs is the C runtime support. The test clients and various additional features have additional prerequisites:

asciidoctor	to build the documentation and www
dbus	gpsd will issue DBUS notifications
gnuplot	to plot gpsprof output.
GTK	for python-GI
libtinfo5	low-level terminfo library (see below)
libusb-1.0.x or later	for older Garmin USB devices
ncurses	for cgps and gpsmon clients
pps-tools	for PPS time keeping
PyGObject	for xps and xgpsspeed clients (see below)
pyserial	for ubxtool and zerk in direct-serial mode
python2.x(x>=6) or 3.y(y>=2)	required for various clients and utilities
python-cairo	for python-GI
Qt	libQgpsmm depends on this

Some ncurses packages contain the terminfo library; some break it out separately as libtinfo5 or libtinfo.

The PyGObject package goes by several names, and is split up into sub packages different ways, depending on the distribution. Sometimes python-gi, python-gobject, python-cairo, etc. The packages also need the underlying system libraries (GTK, GLib, etc.)

The asynchronous python module (gps/aiogps.py) and its example client (example_aiogps.py) require Python 3.6+.

See below for more specific module requirements in the individual distribution instructions.

Installing gpsd

Before installing gpsd on your system, make sure that all parts of any previous installation have been removed. Do not mix gpsd parts from different sources. The gpsd clients and the server must be of the same version.

Install your distributions package(s)

Up-to-date gpsd packages are generally available for Linux distributions including Debian and derivatives (including Ubuntu and Mint), Fedora and derivatives (including CentOS), openSUSE, PCLinuxOS, Mageia, Gentoo, and Slackware. In the embedded space, CeroWRT and Yocto carry GPSD. The GPSD package in the FreeBSD ports tree is also reliably up to date. Even if your distribution is not on this list, it is quite likely GPSD has already been packaged for it.

Whatever distribution you are running, the name of the core GPSD package containing the service daemon is almost certainly "gpsd". However, many distributions break up GPSD into separate installable packages for the core daemon and clients; you should search your repository index for anything with gpsd as a prefix.

Installing using pypi

There is a **gpsd** client on **pypi**. It is not from us, we have no control over it, we can not contact the maintainer, it is very old, and it does not work with any version of **gpsd** released in the last decade.

Don't use it. Install **gpsd** from your distribution repository, or install from source.

How to test the software

1. You should start **gpsd** while running as root. Starting as a normal user will cause some loss of functionality. Starting with `sudo` will cause a different loss of functionality.
2. Start **gpsd**. You'll need to give it as an argument a path to a serial or USB port with a GPS attached to it. Your test command should look something like this:

```
gpsd -D 5 -N -n /dev/ttyUSB0
```
3. Once **gpsd** is running, telnet to port 2947. You should see a greeting line that's a JSON object describing GPSD's version. Now plug in your GPS (or AIS receiver, or RTCM2 receiver).
4. Type `?WATCH={"enable":true,"json":true};` to start raw and watcher modes. You should see lines beginning with `{` that are JSON objects representing reports from your GPS; these are reports in GPSD protocol.
5. Start the **xgps** or **cgps** client. Calling it with no arguments should do the right thing. You should see a display panel with position/velocity-time information, and a satellite display. The displays won't look very interesting until the GPS acquires satellite lock.
6. Have patience. If you are cold-starting a new GPS, it may take 15-20 minutes after it gets a good skyview for it to download an ephemeris for each satellites in view, and the current almanac. Only then can it deliver the best quality fixes.
7. A FAQ and troubleshooting instructions can be found at the GPSD project site.

Once you have verified correct operation

1. If you installed from a `.deb` package under Debian or a Debian-derived system, you may need to `dpkg-reconfigure -plow gpsd` to enable the hotplug magic ("Start **gpsd** automatically").
2. Check out the list of supported hardware at the Hardware page on the GPSD project's website. If your GPS isn't on the list, please send us information to add a new line to the table. Directions are included on that page. We can also use updates of the latest version number known to work with hardware already supported.
3. GPSD includes **gpsd.php**, a PHP script, that you can use to generate a PHP status page for your GPS if you wish. (It may not be in the core package.) It should be manually copied to your HTTP document directory. The first time it's invoked, it will generate a file called `gpsd_config.inc` in that directory containing configuration information; edit to taste.
4. There are other non-essential scripts that may be useful; these are in the `contrib/` directory of the source. They may not be available in the packages available from distributions.

For special instructions related to using GPSD for time service, see the GPSD Time Service HOWTO in the distribution or on the web.

Raspberry Pi tips

Any USB connected GPS that is known to work with **gpsd** will work fine on the RasPi. No special instructions apply.

A very popular option is to install the AdaFruit Ultimate GPS HAT. With this GPS you also get a good 1PPS signal. This works as any other GPS with **gpsd**, but there are two things to note. The GPS takes over the serial console: `/dev/ttyAMA0`. The PPS signal will be on GPIO Pin #4.

Only three specific changes need to be made to make the HAT work. First in the file `/boot/cmdline.txt`, remove this part `"console=ttyAMA0,115200 kgdboc=ttyAMA0,115200"`. That frees the serial port from console use so the GPS can use it.

Second you need to tell the boot process to load the `pps_gpio` module and attach `/dev/pps0` to GPIO pin 4. Do that by adding this line to the bottom of `/boot/config.txt`: `dtoverlay=pps-gpio,gpiopin=4`

Reboot so those changes take effect.

Run `gpsd` like this:

```
~ # gpsd -D 5 -N -n /dev/ttyAMA0 /dev/pps0
```

If you are on the RasPi with `gpsd` version 3.17, or above, `/dev/pps0` can be autodetected, and used for PPS if available.

`gpsd` 3.17 and up only:

```
~ # gpsd -D 5 -N -n /dev/ttyAMA0
```

You can verify `gpsd` is using the PPS by running `ntpshmmon`:

```
~ # ntpshmmon
#      Name  Seen@           Clock           Real           L Prec
sample NTP0  1461619703.641899335 1461619703.445224418 1461619703.000000000 0  -1
sample NTP2  1461619703.642203397 1461619702.999262204 1461619703.000000000 0 -20
sample NTP0  1461619704.142097363 1461619703.445224418 1461619703.000000000 0  -1
sample NTP2  1461619704.142204134 1461619703.999258157 1461619704.000000000 0 -20
```

If you do not see NTP2 then you misconfigured the `pps_gpio` driver.

The serial time is provided to `ntpd` on NTP0, the PPS time is on NTP2, not on NTP1 like described earlier. So your `ntp.conf` will need to be adjusted from:

```
# GPS PPS reference (NTP1)
server 127.127.28.1 prefer
fudge 127.127.28.1 refid PPS
```

To:

```
# GPS PPS reference (NTP2)
server 127.127.28.2 prefer
fudge 127.127.28.2 refid PPS
```

Now proceed as for any other operating system to use `gpsd`.

Be sure to validate that your PPS signal is not offset by the pulse width. That would mean `gpsd` is using the wrong edge.

Detailed instructions are available from their website: <https://learn.adafruit.com/adafruit-ultimate-gps-hat-for-raspberry-pi/>

You will need to dig deeper to make the PPS work, here is a good reference: <http://www.satsignal.eu/ntp/Raspberry-Pi-NTP.html>

Special Notes for Windows

Only Windows Subsystem for Linux 1 provides a reasonable means of running `gpsd` at this time. WSL2 lacks a GUI, USB and serial support making it unsuitable at this time.

About WSL 1

WSL 1 is a component of Microsoft Windows that implements an alternate kernel. Linux distributions, notably Alpine, Debian, Kali, OpenSUSE, and Ubuntu may run on top of it.

There are some issues known which affect gpsd.

- `/dev/ttyS*` nodes have a 1 indexed number, like in MS Windows.
- Windows 10 may attempt to use your GPS itself.
- Older pl2303 (knockoff) serial chipsets are no longer supported \ in Windows 10

Installing a Linux distribution on WSL 1 or WSL 2

1. Install a Linux distribution by clicking on the Microsoft Store \ Icon in the taskbar.
2. Click on the search icon (it is a magnifying glass).
3. Type in 'Linux' or the name of a supported distribution. (see list)
4. Click on the icon of your chosen Linux Distribution
5. Click 'Get' then click 'Install' and busy-wait.
6. Click on the start menu and scroll to your Linux distribution and \ click it.
7. Follow the distribution-specific on-screen instructions to finish \ installing your Linux distribution.

Recommended packages

Due to current WSL limitations, it is recommended at this time that you only install the equivalent of the following packages on your distribution.

```
Python
SCons (preferably 3.0+)
ncurses-dev (to build/run cgps and gpsmon)
asciidoc (to build the documentation)
```

Optionally, the following packages might also be installed

```
pyserial (for direct control of UBlox GPS and GREIS devices)
gnuplot (to generate graphs of gpsprof data)
libusb-dev (to possibly use crusty old Garmin GPS receivers)
git (if building from the development sources)
```

Building on WSL 1 or WSL 2

Follow instructions in the distro-specific section in the file **build.adoc**.

Last updated 2024-10-27 07:02:19 UTC