09/03/2017

Powerwalker UPS under Debian 9 Stretch

With the conversion of my home server to Debian 9 Stretch, it was time to take care of the uninterruptible power supply connected to it. When I bought the UPS, I installed the manufacturer's software for monitoring and automatically shutting down the server on it. After a short time, however, I no longer liked the bloated and poorly maintained Java solution and I did without a UPS software component on the server. This had the disadvantage that the server would not shut down automatically in the event of a power failure, but the UPS provided a certain bridging time.

After the power went out for about an hour last week and the UPS was no longer functional due to an old battery, the topic of "UPS" came up again and I wanted to see what I could do with a new battery and one Coupling with my Linux server can cause. As an alternative to the manufacturer software "ViewPower" (which, by the way, could no longer be installed) I came across NUT - Network UPS Tools . My PowerWalker VI 600 LCD is only marked as "moderately compatible" on the list of compatible devices, but as long as I could read the measured values and the server would shut down automatically when the battery was weak, I was satisfied. So I gave NUT a chance. I want to briefly examine the establishment of NUT in this article.

UPS monitoring with NUT

All necessary software packages are through

```
apt install nut
```

Installed. This includes the 3 components of a NUT system: The appropriate driver for the UPS, upsd and upsmon. Upsd is the daemon that uses the driver to establish the connection to the UPS (server). Upsmon accesses upsd as a client, monitors the measured values and initiates actions such as shutting down the system. To avoid confusion, one should keep this three-way division in mind.

(Background: upsd and upsmon are separate software components so that several UPS monitors / servers can be connected to one upsd / UPS via the network. This is useful if several servers are connected to one UPS.)

Set up drivers and make UPS available

In order to realize a simple setup with NUT, is in the configuration file **/etc/nut/nut.conf**MODEto **standalone**set:

```
MODE=standalone
```

Then the appropriate driver for the UPS is activated. In my case the "blazer_usb" driver has to be activated. Which driver fits which UPS model can be looked up on the compatibility list of the NUT project. The **/etc/nut/ups.conf**following snippet is attached to the configuration file :

```
[powerwalker]
    driver = blazer_usb
    port = auto
    desc = "Meine Powerwalker UPS"
```

This makes the UPS known and activates the appropriate driver. On my Debian system, NUT had no access to the UPS USB device by default. NUT couldn't work like that. So I got **lsusb**a list of the connected USB devices via . The following entry belonged to the UPS:

```
Bus 004 Device 002: ID 0665:5161 Cypress Semiconductor USB to Serial
```

From the vendor and product ID (0665, 5161) I was able to **/etc/udev/rules.d/90-nut-ups.rules**create a new Udev rule which gives NUT the appropriate rights for the device when the USB device is connected:

```
ACTION=="add", \
SUBSYSTEM=="usb", \
ATTR{idVendor}=="0665", ATTR{idProduct}=="5161", \
MODE="0660", GROUP="nut"
```

Vendor ID and Product ID must be **lsusb**adapted to the output of in the configuration . I then received the Udev rule sets via

```
udevadm control --reload-rules
udevadm trigger
```

activated. The "trigger" command didn't work for me straight away, so I quickly disconnected the UPS and reconnected it via USB.

The **upsdrvctl start**command can now be used to check whether the UPS has been correctly recognized. If everything is okay, you can read the following line in the output:

```
Supported UPS detected with mustek protocol
```

Configure UPSd

In the next step the UPS daemon - the UPS server - is set up. The **/etc/nut/upsd.conf**following snippet is added below, which only allows local access to upsd:

```
ACL all 0.0.0.0/0
ACL localhost 127.0.0.1/32
ACCEPT localhost
REJECT all
```

In **/etc/nut/upsd.users**addition, a new user is created in who should have access to the daemon. (add the following below :)

```
[local_mon]
    password = mypass
    allowfrom = localhost
    upsmon master
    instcmds = ALL
```

the password “mypass” can of course be freely chosen.

Configure UPSMon

So that the UPS Monitor monitors the newly set up UPS, **/etc/nut/upsmon.conf**the following section is added:

```
MONITOR powerwalker@localhost 1 local_mon mypass master
POWERDOWNFLAG /etc/killpower
SHUTDOWNCMD "/sbin/shutdown -h now"
```

The password corresponds to the one selected above.

Start NUT

The NUT services for server and monitor are now started:

```
systemctl start nut-server
systemctl start nut-client
```

It is worth taking a quick look at the current status of the services via **systemctl status nut-server**or **systemctl status nut-client**to see whether the configuration is correct and the services are working properly.

Output UPS values

Current values for battery voltage, mains voltage and frequency and a lot of other status information can now be sent with

```
upsc powerwalker
```

issued or via

```
watch -n 1 upsc powerwalker
```

to be observed.

Settings such as the activation of the alarm signal in the event of a power failure can also be changed:

```
upscmd -u local_mon -p mypass powerwalker@localhost beeper.toggle
```

(Pay attention to the password “mypass”!). A short battery test can be performed via

```
upscmd -u local_mon -p mypass powerwalker@localhost test.battery.start.quick
```

be carried out. A list of all so-called “instant commands” can be found here: http://networkupstools.org/docs/developer-guide.chunked/apas02.html