



Published on Free Software Magazine (<http://www.freesoftwaremagazine.com>)

UPS (Uninterruptible Power Supply) installation and configuration

Preventing unscheduled power related downtime

By [Ken Leyba](#)

An inexpensive way to prevent unscheduled downtime or data loss due to power problems is with a UPS or Uninterruptible Power Supply. However, a UPS by itself is not enough for proper operation. Hardware, software, and configuration together make up a UPS system that will recover from unexpected power loss or power fluctuations that can damage systems and peripherals.

Introduction

When considering data loss, system downtime and disaster recovery, backup methods are primarily discussed. There are many methods of preventing data loss, including clustering, backup, security and power conditioning. Proper power can prevent an initial disaster from ever occurring. Providing proper power can be in the form of an Uninterruptible Power Supply or UPS. A UPS has rechargeable batteries to supply emergency power in the event of immediate power loss. If the power loss is longer than the batteries can supply, then the UPS can signal the server to initiate a power down sequence to properly shutdown, preventing data loss. When power is returned the server can return to operation after having made a clean shutdown.

Other power related problems that occur can be minimized with the circuitry of a UPS. Voltage sags and spikes, brown outs and line noise (from other machinery like elevators, air conditioners or office equipment), can all be isolated by a UPS. These power related fluctuations can wreak havoc on systems and devices. For a relatively low cost a UPS can prevent downtime due to power anomalies.

Network UPS Tools

The Network UPS Tools (NUT) [1] are a group of tools that are used to monitor and administer UPS hardware. NUT uses a layered scheme of equipment, drivers, server and clients. The equipment consists of the monitored UPS hardware. Drivers specific to the UPS hardware communicate or poll the UPS for status information in the form of variables. The driver programs talk directly to the UPS equipment and run on the same host as the server. The server `upsd` serves data from the drivers to the network. Clients talk to the `upsd` server and initiate tasks with the status data.

As indicated by the name, Network UPS Tools, NUT is a network based UPS system that works with multiple UPSs and systems. One of the many features of NUT allows multiple systems to monitor a single UPS, not requiring special UPS sharing hardware connections. The master/slave relationship synchronizes shut-downs so the slaves can initiate power-down sequences before the master switches off UPS power.

UPS (Uninterruptible Power Supply) installation and configuration

This article details the installation and configuration of a single system with a UPS connected to the serial port of the system. This is the natural first step of getting NUT installed and configured. If the UPS will supply more than one system, the second and subsequent systems can be configured as slaves.

The NUT developers also have a different take on when the systems should be powered down. NUT will wait until the UPS is “on battery” and “low battery” before it considers the UPS “critical”. This philosophy gets the most out of the UPS batteries and will wait until the critical moment to initiate a power down sequence, just in case the power comes back on line. There is an option to override this behavior if desired with `upssched`, which can be found in the documentation. With the `upssched` utility, commands can be invoked based on UPS events.

In typical GNU/Linux fashion, NUT is not the only tool available for monitoring a UPS. `Apcupsd` [2] is used for power management and control of APC model UPSs. There are also several graphical frontends for workstation class machines.

Preparing for installation

Prior to installing and using a UPS and its associated software, a few things must be in place. Since the system is going to be shutdown there must be a way to bring the system back up when power returns. The system BIOS needs to be configured correctly. Most modern BIOSes have an option to power-on when main power, now supplied by the UPS, is returned. Server system BIOSes will most likely support the “power on when power returns” option. If the BIOS does not support this option (more common with workstation class systems), a BIOS update may correct it. With servers configured headless, without a monitor or keyboard, there are also settings to ignore keyboard errors. Commonly these types of systems are administered via SSH or administration utilities like Webmin [3].

The UPS must also have the correct signal cable from the UPS to the system. With a USB type of UPS this is not a concern. A UPS that communicates via the serial port needs the correct signal cable that supports intelligent signaling between the UPS and system. See the UPS vendor for the correct cable or a custom cable can be built with information from the Network UPS Tools web site.

Installing NUT

The example system is running a basic install of Debian GNU/Linux V4.0 [4] and utilizing an APC SmartUPS 700. Debian is an excellent, long term supported GNU/Linux, which is ideal for small enterprise deployments, as well as much larger environments. Different GNU/Linux distributions may install the software and configuration files in different directories. Since the server is configured without a GUI interface, all commands and configuration are done through the command line as the `root` user. Using the APT package tool, `apt-get`, the package `nut` can easily be installed:

```
# apt-get install nut
```

The package tool installs the NUT software, documentation, man pages and example configuration files. Debian specific documentation is found in the `/usr/share/doc/nut/` directory. Extensive NUT documentation can be found in `/usr/share/doc/nut/docs/` and the example configuration files in `/usr/share/doc/nut/examples/`. Some of the documentation is compressed with `gzip` which can be uncompressed or viewed with the `zcat` utility.

```
# zcat /usr/share/doc/nut/README.Debian.gz | less
```

UPS (Uninterruptible Power Supply) installation and configuration

The configuration files exist in the `/etc/nut/` directory. The `ups.conf` configuration file contains the UPS definitions. The UPS is defined with the `[labsvr]` entry. The `driver` and `port` fields must be defined, the `desc` field is optional and describes the UPS. Additional UPS definitions can be configured in this file; however, this example is a single UPS and server configuration.

```
[labsvr]
  driver = apcsmart
  port = /dev/ttyS0
  desc = "Lab Server"
```

The definition, between the square brackets, is user definable, with the exception of word `default`, since it is used by the Network UPS Tools. The correct driver name for your UPS can be found in the file `/usr/share/nut/driver.list`. For proper permissions to use the serial port the `nut` user must be added to the “dialout” group, and can be accomplished with the `addgroup` command. To manually test the configuration and to verify the configuration is correct the `upsdrvctl` (UPS driver controller) command is used. After verification the driver can be stopped.

```
# addgroup nut dialout
Adding user `nut' to group `dialout' ...
Done.
# /sbin/upsdrvctl start labsvr
Network UPS Tools - UPS driver controller 2.0.4
Network UPS Tools (version 2.0.4) - APC Smart protocol driver
  Driver version 1.99.8, command table version 2.0
Detected SMART-UPS 700 [QS0331213446] on /dev/ttyS0
# /sbin/upsdrvctl stop labsvr
Network UPS Tools - UPS driver controller 2.0.4
Stopping UPS: labsvr
#
```

Since this example is a single server, the access control list for server communication is minimal. Configuring the access control lists is accomplished in the configuration file `upsd.conf`. The ACL (access control list) `all` is defined with the `netblock` in CIDR format, the old style `address/network` format can also be used. Additionally the ACL `localhost` is defined. The `ACCEPT` field allows communication to the server for `localhost` and the `REJECT` field blocks all other access. Similar to other access control lists, flow goes from the top down. The `ACCEPT` is evaluated before the `REJECT`, if the `REJECT` line were before `ACCEPT`, `localhost` would meet the rule and not be allowed access.

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

The `upsd.users` configuration file is used to define users that have access to administrative commands. Here are defined the users and what access each user is allowed; each section begins with user names in brackets and continues to the next bracketed user or end of the file. The `password` field defines the user’s password. The `allowfrom` field grants access based on the user’s source IP address, the values are defined in the ACL lists configuration file `upsd.conf`. The `upsmon` field is set to either `master` or `slave` to allow the `upsmon` process to work.

```
[monmaster]
  password = p455w0rd
  allowfrom = localhost
  upsmon = master
```

UPS (Uninterruptible Power Supply) installation and configuration

The final configuration file `upsmon.conf` defines which systems the `upsmon` process will monitor, as well as how to shutdown systems when necessary. The `MONITOR` line defines the UPS to monitor. The first field is the UPS to monitor, in this case `labsvr@localhost`. The second field is the power value that defines the number of power supplies the UPS supplies. In simple configurations this is normally set to 1. The next two fields are the user name and password that were previously defined in `upsd.users`. The last field will be either `master` or `slave` process. A `master` process is one in which the process is running on the system that is plugged in directly and communicates with the UPS. A `slave` is a process that gets power from the UPS but doesn't communicate directly to it.

```
MONITOR labsvr@localhost 1 monmaster p455w0rd master
POWERDOWNFLAG /etc/killpower
SHUTDOWNCMD "/sbin/shutdown -h +0"
```

The `POWERDOWNFLAG` defines a file name to be created in *master* mode when the UPS needs to be powered off. This file is cleared when the system comes back up. Finally, `SHUTDOWNCMD` is the actual shutdown command performed enclosed in quotes.

After the configuration of the UPS and Network UPS Tools is complete, a couple of housekeeping tasks need to be accomplished. Since several of the configuration files contain user names and passwords, they will have permissions set so only the `root` user and `nut` group can read them with the following commands:

```
# chown root:nut /etc/nut/*
# chmod 640 /etc/nut/*
```

With Debian GNU/Linux, two items must also be changed in the file `/etc/default/nut`, `START_UPSD` and `START_UPSMON` are changed from “no” to “yes”.

```
START_UPSD=yes
START_UPSMON=yes
```

The `nut` `init.d` script can be run to start the UPS monitor tools and `/var/log/syslog` is checked to verify everything is running correctly.

```
# /etc/init.d/nut start
Starting Network UPS Tools: upsdrvctl upsd upsmon.

# tail /var/log/syslog
Sep 01 13:36:48 labserver apcsmart[2519]: Startup successful
Sep 01 13:36:48 labserver upsd[2520]: Connected to UPS [labsvr]: apcsmart-ttyS0
Sep 01 13:36:50 labserver upsd[2521]: Startup successful
Sep 01 13:36:50 labserver upsmon[2523]: Startup successful
Sep 01 13:36:50 labserver upsd[2521]: Connection from 127.0.0.1
Sep 01 13:36:50 labserver upsd[2521]: Client monmaster@127.0.0.1 logged into UPS [labsvr]
```

To quickly poll the status of a UPS server, the `upsc` UPS client utility is used. The first example displays the value of the `ups.status` variable, which is `OL` (or “on line”), meaning the UPS `labsvr@localhost` is on line power. If the value of `ups.status` were `OB` (“on battery”), then the UPS would be supplying battery power to the server. The second invocation displays all available variables and the values for `labsvr@localhost`.

```
# upsc labsvr@localhost ups.status
OL
# upsc labsvr@localhost
battery.alarm.threshold: 0
battery.charge: 100.0
```

UPS (Uninterruptible Power Supply) installation and configuration

```
battery.charge.restart: 00
battery.date: 08/02/03
battery.packs: 000
battery.runtime: 7860
battery.runtime.low: 120
battery.voltage: 27.60
battery.voltage.nominal: 024
driver.name: apcsmart
driver.parameter.port: /dev/ttyS0
driver.version: 2.0.4
driver.version.internal: 1.99.8
input.frequency: 60.00
input.quality: FF
input.sensitivity: H
input.transfer.high: 132
input.transfer.low: 103
input.transfer.reason: S
input.voltage: 120.2
input.voltage.maximum: 121.5
input.voltage.minimum: 119.6
output.voltage: 120.2
output.voltage.target.battery: 115
ups.delay.shutdown: 180
ups.delay.start: 000
ups.firmware: 50.14.D
ups.id: UPS_IDEN
ups.load: 008.3
ups.mfr: APC
ups.mfr.date: 08/02/03
ups.model: SMART-UPS 700
ups.serial: QS0331213446
ups.status: OL
ups.temperature: 037.8
ups.test.interval: 1209600
ups.test.result: NO
#
```

Testing power loss is accomplished with the `upsdrvctl` utility. The value of `ups.delay.shutdown` is the amount of time in seconds the UPS will wait before shutting down. In the above listing the value is 180 seconds. This value can be changed with the `upswr` utility, though a valid user in `upsd.users` with proper permissions must be defined to change variable values. Refer to the `upsd.users` and `upswr` man pages for more information. The 180 second delay is enough time to allow for a proper shut down of this system.

```
# upsdrvctl shutdown labsvr; shutdown -h +0
```

After this command is run, `upsdrvctl` tells the UPS to issue its shutdown sequence. The second command tells the system to shutdown immediately. The server shuts down and after the 180 second delay the UPS shuts down and powers back up. If the BIOS is set correctly in the server, when the UPS supplies line power again the server system comes back on line.

In addition to the extensive documentation installed in `/usr/share/doc/nut/` and on the NUT web site, the man pages contain excellent detailed information on the utilities, configuration files and drivers.

Conclusion

Power interruptions are a common problem in many areas and can cause eventual failure of components and systems. Having an uninterruptible power supply to prevent damage and initiate proper power down

UPS (Uninterruptible Power Supply) installation and configuration

sequences can save many headaches as well as avoid disaster. Just implementing a UPS system is not enough and the proper UPS, server cabling and motherboard BIOS are all part of a reliable system.

Bibliography

[1] [Network UPS Tools](#)

[2] [Apcupsd, APC UPS Daemon](#)

[3] [Webmin, Web based system administration](#)

[3] [Debian GNU/Linux](#)

Biography

[Ken Leyba](#) (/user/5507" title="View user profile.): Ken has been working in the IT field since the early 80's, first as a hardware tech whose oscilloscope was always by his side, and currently as a system administrator. Supporting both Windows and Linux, Windows keeps him consistently busy while Linux keeps his job fun.

Copyright information

This article is made available under the "Attribution-NonCommercial" Creative Commons License 2.5 available from <http://creativecommons.org/licenses/by-nc/2.5/>.

Source URL:

http://www.freesoftwaremagazine.com/articles/ups_installation_and_configuration
