



Apcupsd Linux UPS daemon version 3.10.5

Apcupsd 3.10.5 User's Manual

Last update to this manual 24 February 2003

Release Notes

- [Known Bugs](#)
- [New features in Apcupsd 3.10.5](#)
- [New features in Apcupsd 3.10.4](#)
- [Features in Previous Versions](#)
- [Apcupsd License](#)

Apcupsd Reference Manual

- [General -- Man Page](#)
- [Quick Start for Beginners](#)
- [UPS Models and Cables Supported](#)
- [Operating Systems Supported](#)
- [Main Configuration Types](#)
- [Compiling and Installing](#)
- [Starting](#)
- [Stopping or Restarting](#)
- [Configuration Directives](#)
- [Configuration Examples](#)
- [Master/Slave Configuration](#)
- [Cables](#)
- [Testing](#)
- [Shutdown Sequence](#)
- [Trouble Shooting](#)
- [Apcupsd Network Information Server](#)
- [Apcupsd EVENTS](#)
- [Apcupsd STATUS](#)
- [Apcupsd DATA](#)
- [Apcupsd System Logging](#)
- [The CGI Network Interface](#)
- [apcaccess](#)
- [apctest](#)
- [Configuring your EEPROM](#)
- [UPS Programming Bible](#)
- [Apcupsd under Windows](#)
- [Using Apcupsd with a USB UPS](#)
- [Using Apcupsd with a SNMP UPS](#)
- [Configuration for Controlling Multiple UPSes](#)

- [Batteries](#)

Other Notes

- [Frequently Asked Questions](#)
 - [Security](#)
 - [Thanks](#)
-

Copyright (C), 1999–2003, Kern E. Sibbald

New Features in Apcupsd 3.10.5

This release is primarily version 3.10.4 but including a fix that closes a root exploit of slave machines. In addition, it makes the **--enable-master-slave** ./configure option work and has a few updates to the Mandrake release. See the ChangeLog below for more details.

apcupsd is mainly developed under Linux and will compile cleanly and work under most flavors of Unix as well as many other operating systems including Windows.

What to do if you find bugs :

send an email to apcupsd-devel@apcupsd.org (Developers mailing list) or go to one of the following sites:

<http://www2.apcupsd.com>

<http://sourceforge.net/projects/apcupsd>

<http://www.apcupsd.com>

Please be sure to include the version of **apcupsd** you are running, your operating system, and a detailed description of your problem.

Change Log

```
----> Release apcupsd-3.10.5 (03 Feb 2003)
03Feb03
- Added an avsnprintf() routine.
- Replaced all vsprintf() calls with avsnprintf() to close a
  master/slave exploit that was published.
- Remove awk processing of halt script for Mandrake as suggested by
  David Walser.
02Feb03
- Corrected --enable-oldnet to be --enable-master-slave as I
  had intended. Thanks to David Walser for pointing this out.
- Added David Walser's apcupsd.spec.in for Mandrake and his
  changes to configure.in.
```



New Features in Apcupsd 3.10.4

See the list below for the detailed change log. Note, a number of the details of the changes are only documented in the **apcupsd.conf** file, and unfortunately not in much detail.

The main new features are:

- Support for USB UPSes on Linux. The killpower feature does not work though (most likely a kernel "bug").
- Support for additional models and cables.
- New cable for running BackUPS CS in Smart mode (possibly also the BackUPS ES).
- The old master/slave code must be explicitly enabled with **--enable-master-slave** option on the **./configure** line.
- Different models are now handled by drivers.
- NIS code, master/slave code, and the drivers can be individually configured giving a much smaller memory foot print.
- A host of new configuration options to support enabling/disabling drivers and features.
- Support for limiting what subnets the NIS code will listen to using the new **NISIP** configuration directive.
- SNMP support.
- Support for system provided gd libraries.
- Source tree reorganization.

apcupsd is mainly developed under Linux and will compile cleanly and work under most flavors of Unix as well as many other operating systems including Windows.

What to do if you find bugs :

send an email to apcupsd-devel@apcupsd.org (Developers mailing list) or go to one of the following sites:

<http://www.apcupsd.org>

<http://www.sibbald.com/apcupsd>

Please be sure to include the version of **apcupsd** you are running, your operating system, and a detailed description of your problem.

Change Log

```
----> Release apcupsd-3.10.4 (21 Jan 2003) (not yet released)
- Added error messages if old master/slave code called.
- Reworked the messages for ./configure --help to be aligned and
  clearer for networking and master/slave
- A number of important patches all supplied by Mirko Doelle.
- Moved technotes for 2001 and 2002 into respective directories.
- Created a new kes-3.10.4 file to which I will append if there
  are additional changes to 3.10.4. This will reduce the number
  of release note files.
```

Changes submitted this submission:

19Jan03

- Fixed hangs in usb driver startup when could not open port by releasing the ups structure lock before the Error_abort calls. Thanks to Mirko Doelle for reporting this.

APC UPS management under Linux

- Fixed the default path for mkinstalldirs from \$(topdir) to \$(topdir)/autoconf. Reported by Mirko.
- Fixed configure.in to always create platform/apccontrol. Previously it was not being created for Debian. Reported by Mirko.
- Removed Debian specific installation of apccontrol. Reported by Mirko.
- Implemented code to display the apcupsd events with the most recent first. Code sent by Mirko, but I modified it slightly.

----> Release apcupsd-3.10.3 (12 Dec 2002)

- Tried to correct problems with Makefiles
- Thanks to David Walser who pointed out where on the Sun, the make install was doing terrible things -- I found that there was a missing semicolon in the new Makefile. Before my previous cleanup, there were actually 4 missing semicolons. Hopefully this will correct the problem.
- For a second time, David Walser came to the rescue finding the CGI install problem reported by lots of people. The new code used "make" instead of \$(MAKE) to call the CGI make. Fixed.

Changes submitted this submission:

- Removed Makefile code that creates and sets permission bits on /tmp \$(prefix) and \$(exec-prefix)
- Removed all occurrences of -z in the Makefiles (at least that I found) and replaced them with a more conservative formulation.
- Removed the install-symlinks script that caused some problems on distributions with blanks in the DISTVER name.
- Added install-symlinks to the suse Makefile.in. This is the only platform that currently uses it.

----> Release apcupsd-3.10.2 (08 Nov 2002)

- New cable design for BackUPS CS models to run it in Smart serial mode.
- Corrected a major bug in the smart and net code where the status word was getting clobbered.

Changes submitted this submission:

- A few days ago, slither_man sent me an email with some information on how to run a BackUPS CS in smart mode with a serial cable. He found the information by assuming that the UPS supported Smart mode and through trial and error. Well, it works!!!!!! Amazing!!!! Thanks slither_man.
- Documentation for new CUSTOM-RJ54 cable that can be constructed from the end of an ethernet cable and a DB9F connector.
- Changed all the MAIL instances in shell scripts into APCUPSD_MAIL. This helps keep separated the apcupsd specific shell variables from the generic \$MAIL shell variable that points to the user's mailbox:

```
riccardo@ao:~> echo $MAIL
/var/spool/mail/riccardo
riccardo@ao:~>
```

An user reported that ./configure script transformed internal \$MAIL executable program into her mailbox path. This may happen if the configure suite is broken (thing that I don't want to check further). That said, APCUPSD_MAIL now should always correctly point to the system default mail client program.

- Made so that error_cleanup is a generic function called by the generic error handlers. Now if you want specific error_cleanup

APC UPS management under Linux

you don't need to write also specific error handlers, provided that `error_cleanup` don't accept parameters (i.e. `specific_error_cleanup(void)`) but if you want to have a specific `error_cleanup` with parameters you must also write specific `error_exit` and `error_out` into which you will call the specific `error_cleanup` with parameters.

- Made so that `error_exit` and `error_out` are generic handlers that can be assigned, if needed, to specific handlers by the `main()` of each program. If not, the `Error*` routines will use the generic versions in `apclib.a`.
- Fixed wrong "true" usage in `powerflute`.
- Cleaned up the `terminate()` functions.
- Made `DeviceVendor` part of `snmp DEVICE` case insensitive.

----> Release `apcupsd-3.10.1` (16 Sep 2002)

- Fixed a filling error with USB status dword.
- Fixed `autoconf` check and dependances of `-lpanel`, etc with `-lncurses`.
- Made more portable the `apccontrol` external scripts when calling the mailer (subject is now in the echo lines instead of relying on the presence of a `-s` switch on the mailer).
- Added `gentoo` platform.
- Added `DESTDIR` variable for platform packaging.
- Fixed a off-by-one problem in events table.
- Conditional compilation of old and new network code. Old network code disabled by default while new network code enabled by default.
- Removed old `src/apcnet.c.old`, old implementation of old networking.

----> Release `apcupsd-3.10.0` (28 Jul 2002)

- Added documentation for SNMP UPSes. Documented the use of `--kill-on-powerfail` switch during shutdown.
- SuSE 8.0 is now supported.
- Added forward declaration of `inet_pton` and `localtime_r` when they are extraobj.
- Added `inet_pton` function. Implementation from Internet Software Consortium.
- Made `sp_flags` private to the dumb driver.
- Can't SET/CLEAR multiple flags: do them one by one. Fixed this bug in SNMP driver.
- Added 127A and 128A cables support for dumb UPSes.
- Implemented `killpower` for PowerNet MIB.
- Implemented the SNMP driver for APC's PowerNet MIB.
- Restructured `UPSINFO` so that now all the flags are contained into the Status bitmap.
- *BSD should compile cleanly again.
- Source tree is now under CVS revision control.
- Added support for listening on specific IP addresses/subnets in NIS server, from Troy.
- Doc updates, from Kern.

----> Release `apcupsd-3.9.9` (18 May 2002)

- Applied final Kern's patch.
- Added a little program 'devicedbg' to help in debugging device drivers with `gdb`. To compile, 'make devicedbg' in `src/`.
- Cygwin platform added (reorganized old cygwin files).
- Darwin platform added.
- compile line is 'gcc -c -g -O2 -I../include apcaction.c'.
- reviewed all the platform makefiles.
- use system `libgd`, searching for include files in system include dirs.
- in case system does not have `libgd`, uses provided `libgd`.
- put `gd1.2` into master's contrib directory and a message if `gd1.2` is not present into `src/gd1.2` (like default distribution will not)

APC UPS management under Linux

- is issued at configure time to get gdl.2 from contrib and extract it into the src/ directory. Re-run config and all will be good and happy.
- Sources reorganization.
 - Mandrake platform added.
-





Apcupsd Linux UPS daemon version 3.8.3

apcupsd a program for controlling APC UPSes

SYNOPSIS

`/sbin/apcupsd`
`/etc/apcupsd/apccontrol`
`/etc/apcupsd/apcupsd.conf`
`/sbin/apcaccess`
`/sbin/apcnisd`

DESCRIPTION

Apcupsd can be used for controlling APC UPSes. During a power failure, **apcupsd** will inform the users about the power failure and that a shutdown may occur. If power is not restored, a system shutdown will follow when the battery is exhausted, a timeout (seconds) expires, or runtime expires based on internal APC calculations determined by power consumption rates. If the power is restored before one of the above shutdown conditions is met, **apcupsd** will inform users about this fact.

Apcupsd runs on Linux systems, many Unix systems (Solaris, FreeBSD, OpenBSD, Alpha, ...) as well as on Windows systems (Win95/98/Me/NT/2000).

Apcupsd performs the shutdown by calling `/etc/apcupsd/apccontrol`, which is a shell script. Consequently, no changes to `/etc/inittab` are necessary. There is no communication between **apcupsd** and `init(1)` process. During installation, the system halt script was modified so that at the end of the shutdown process during a power failure, **apcupsd** will be re-executed in order to power off the UPS. This step is not mandatory, but is good practice as it avoids the possibility of your system being prematurely restarted if the power returns for a short period. It also permits your computer to be automatically rebooted when the power is restored providing you have configured your computer BIOS appropriately.

The **apcupsd** daemon now supports two networking modes that function independently, but if desired at the same time.

Most users will probably enable the first network mode, which permits **apcupsd** to serve [STATUS](#) and [EVENTS](#) information to clients over the network.

The second networking mode is for multiple networked machines that are powered by the same UPS. In this mode, one machine is configured as a master with the UPS attached to the serial port. The other machines (maximum 20) powered by the same UPS are configured as slaves. The master has a network connection with the slaves and sends them information about the UPS status. Please see the [UPS Sharing section](#) of this document for more details.

RedHat and SuSE. versions of Linux have direct install support, as does Solaris. All other flavors of Linux

and Unix machines may need some fussing with to get the install correct.

SUPPORTED MODELS

Please see the [Configuration section](#) of this document for more details.

OPTIONS

- `-c --configure`
Attempts to configure the UPS EPROM to the values specified in the configuration file **/etc/apcupsd/apcupsd.conf**.
- `-d --debug <level>`
Turns on debugging output for a NETSLAVE or a NETMASTER.
- `-f --config-file <file>`
Specifies the location of the configuration file. The default is: **/etc/apcupsd/apcupsd.conf**
- `-k --killpower`
Attempt to turn the UPS off. This option is normally only used by the daemon itself to shut the UPS off after a system shutdown has completed. Note, if the killpower is done by a netmaster, it will sleep for 30 seconds before issuing the power kill request to the UPS to allow the slaves to properly shutdown.
- `-n --rename-ups`
Attempts to change the UPS name stored in the UPS EPROM to that specified in your **/etc/apcupsd/apcupsd.conf** file.
- `-p, --kill-on-powerfail`
If this option is specified and a power failure occurs, immediately after **apcupsd** issues the system shutdown request, it will instruct the UPS to shut off the mains power. This option can be useful if your system does not let **apcupsd** regain control at the end of the shutdown procedure to shut off the mains power. However, if you use this option, you must insure that your UPS has a sufficiently long power off grace period to permit your computer to fully shutdown.
- `-t, --term-on-powerfail`
This option tells **apcupsd** to exit immediately when it issues the system shutdown command (i.e. calls **apccontrol**). This behavior can be useful for those systems where it is not possible to insert **apcupsd** commands in the termination script in order to do the killpower. Without this option, **apcupsd** will wait for the **SIGTERM** signal from the system shutdown before exiting.
- `-u --update-battery-date`
Attempts to update the battery date stored in the UPS EPROM. Normally done after a battery replacement.
- `-V --version`
Prints the **apcupsd** version number and the usage.
- `-? --help`
Prints a brief **apcupsd** help screen.

FILES

/etc/apcupsd/apcupsd.conf – configuration file.

/etc/apcupsd/apcupsd.status – STATUS file

/etc/apcupsd/apcupsd.events

where up to the last 50 events are stored for the network information server.

SEE ALSO

apcnisd(8)

For credits, please see the [Thanks Chapter](#) of this manual.



Quick Start for Beginners

For beginners, setting up **apcupsd** can be a bit overwhelming because of the new terminology and the large number of options available in **apcupsd**. As a consequence, the following is meant to help guide to the steps needed to get it up and running as painlessly as possible.

- First take a look at the [Configuration Chapter](#) of the manual. In particular, look at the [models and cables supported](#).
- Decide if you have a Smart UPS or a Dumb UPS.
- Find out what cable type you have by looking on the flat ends of the cable for a number such as 940-0020A stamped in the plastic.
- Read the [Building and Installing](#) section of the manual.
- Configure, build, and install **apcupsd**
- Tweak your `/etc/apcupsd/apcupd.conf` file as necessary.
- Read and follow the instructions in the [Testing](#) chapter of the manual.
- If you run into problems, read the [Trouble Shooting](#) section of the manual.
- If you still need help, send a message to the developer's email list describing your problem, what version of **apcupsd** you are using, what Operating System you are using, and anything else that would be helpful.



Building and Installing Apcupsd

The installation can be made several different ways depending on what system you are running. The basic procedure involves getting a source distribution, running the configuration, rebuilding, and installing. For RedHat systems, **apcupsd** is available in binary RPM format as well as source RPM format. Please see [RedHat RPM Installation](#) below for more details of the RPM installation. For Microsoft Windows systems, there are two forms of binary install (tar file, and setup.exe). Please see [Win32 Installation](#) below for more details of the Windows install.

The basic installation from a tar source file is rather simple:

1. Detar the source code.
2. **cd** to the directory containing the source code.
3. `./configure` (with appropriate options as described below)
4. `make`
5. `su` (i.e. become root)
6. Stop any running **apcupsd**
`<system-dependent-path>/apcupsd stop`
7. **uninstall any old **apcupsd****
 This is important since the default install locations may have changed.
8. `make install`
9. edit your `/etc/apcupsd/apcupsd.conf` file if necessary
10. ensure that your halt script is properly updated
11. Start the new **apcupsd** with:
`<system-dependent-path>/apcupsd start`
12. IMPORTANT! Test the installation as outlined in the [Testing Chapter](#) of this document.

If all goes well, the `./configure` will correctly determine which operating system you are running and configure the source code appropriately. **configure** currently recognizes the systems listed below in the [Operating System Specifics](#) section of this chapter and adapts the configuration appropriately. Check that the configuration report printed at the end of the `./configure` process corresponds to your choice of directories, options, and that it has correctly detected your operating system. If not, redo the `./configure` with the appropriate options until your configuration is correct.

Please note that a number of the `./configure` options preset **apcupsd.conf** directive values in an attempt to automatically adapt **apcupsd** as best possible to your system. You can change the values in **apcupsd.conf** at a later time without redoing the configuration process by simply editing the **apcupsd.conf** file.

For systems other than those mentioned above, you may need to do some tweaking.

In general, you will probably want to supply a more complicated **configure** statement to ensure that the modules you want are built and that everything is placed into the correct directories.

On RedHat, I use the following:

```
CFLAGS="-g -Wall" LDFLAGS="-g -Wall" ./configure \
--prefix=/usr \
--sbindir=/sbin \
--with-cgi-bin=/home/www/sibbald/new/cgi-bin \
--enable-cgi \
--with-css-dir=/home/www/sibbald/new/docs/css \
```

```
--with-log-dir=/etc/apcupsd \
--enable-pthreads \
--enable-powerflute
```

Default Options

By default, `make install` will install the executable files in `/sbin`, the manuals in `/usr/man`, and the configuration and script files in `/etc/apcupsd`. In addition, if your system is recognized, certain files such as the startup script and the system halt script will be placed in appropriate system directories (usually subdirectories of `/etc/rc.d`).

Checking the Installation

There are a number of things that you can do to check if the installation (`make install`) went well. The first is to check where the system has installed **apcupsd** using **which** and **whereis**. On my RedHat 6.1 system, I get the following (lines preceded with a \$ indicate what I typed):

```
$ which apcupsd
/sbin/apcupsd
```

and

```
$ whereis apcupsd
apcupsd: /sbin/apcupsd /etc/apcupsd /etc/apcupsd.conf /etc/apcupsd.status /usr/man/man8/apcupsd.8.gz
/usr/man/man8/apcupsd.8
```

If you find an **apcupsd** in `/usr/sbin`, `/usr/local/sbin`, `/usr/lib`, or another such directory, it is probably a piece of an old version of **apcupsd** that you can delete. If you are in doubt, delete it, then rerun the **make install** to ensure that you haven't deleted anything needed by the new **apcupsd**. Please note that the files specified above assume the default installation locations.

Final Installation Check

As a final check that the **make install** went well, you should check your halt script (in `/etc/rc.d` on SuSE systems, and in `/etc/rc.d/init.d` on RedHat systems) to see that the appropriate lines have been inserted in the correct place. Modification of the halt script is important so that at the end of the shutdown procedure, **apcupsd** will be called again to command the UPS to turn off the power. This should only be done in a power failure situation as indicated by the presence of the `/etc/powerfail` file, and is necessary if you want your machine to automatically be restarted when the power returns. On a RedHat system, the lines containing the *****apcupsd***** should be inserted just before the final halt command:

```
# Remount read only anything that's left mounted.
#echo "Remounting remaining filesystems (if any) readonly"
mount | awk '/ext2/ { print $3 }' | while read line; do
    mount -n -o ro,remount $line
done

# See if this is a powerfail situation.
if [ -f /etc/apcupsd/powerfail ]; then
    echo
    echo "APCUPSD will now power off the UPS"
    echo
    # ***apcupsd***
    # ***apcupsd***
    # ***apcupsd***
    # ***apcupsd***
    # ***apcupsd***
```

```

/etc/apcupsd/apccontrol killpower                                # ***apcupsd***
echo                                                            # ***apcupsd***
echo "Please ensure that the UPS has powered off before rebooting" # ***apcupsd***
echo "Otherwise, the UPS may cut the power during the reboot!!!" # ***apcupsd***
echo                                                            # ***apcupsd***
fi                                                            # ***apcupsd***

# Now halt or reboot.
echo "$message"
if [ -f /fastboot ]; then
    echo "On the next boot fsck will be skipped."
elif [ -f /forcefsck ]; then
    echo "On the next boot fsck will be forced."
fi

```

The purpose of modifying the system halt files is so that **apcupsd** will be recalled after the system is in a stable state. At that point, **apcupsd** will instruct the UPS to shut off the power. This is necessary if you wish your system to automatically reboot when the mains power is restored. If you prefer to manually reboot your system, you can skip this final system dependent installation step by specifying the **—disable—install—distdir** option on the **./configure** command (see below for more details).

The above pertains to RedHat systems only. There are significant differences in the procedures on each system, as well as the location of the halt script. Also, the information that is inserted in your halt script varies from system to system. Other systems such as Solaris require you to make the changes manually, which has the advantage that you won't have any unpleasant surprises in your halt script should things go wrong. Please consult the specific system dependent README files for more details.

Please note that if you install from RPMs for a slave machine, you will need to remove the changes that the RPM install script made (similar to what is noted above) to the halt script. This is because on a slave machine there is no connection to the UPS, so there is no need to attempt to power off the UPS. That will be done by the master.

Configure Options

All the available **configure** options can be printed by entering:

```
./configure --help
```

When specifying options for **./configure**, if in doubt, don't put anything, since normally the configuration process will determine the proper settings for your system. The advantage of these options is that it permits you to customize your version of **apcupsd**. If you save the **./configure** command that you use to create **apcupsd**, you can quickly reset the same customization in the next version of **apcupsd** by simply re-using the same **./configure** command.

If you are setting up a master/slave configuration, you will be required to make some modifications to the **apcupsd.conf** files after the configuration process. For more details on a master/slave setup (two computers powered by the same UPS), please see the [Configuration Examples Chapter](#) of this document. In addition, you will find some schematic diagrams of the possible configurations in the [Configuration Chapter](#).

The following command line options are available for **configure** to customize your installation.

```
--prefix=<path>
```

This defines the directory for the non-executable files such as the manuals. The default is **/usr**.

--*sbindir*=<path>

This defines the directory for the executable files such as **apcupsd**. The default is **/sbin**. You may be tempted to place the executable files in **/usr/sbin** or **/usr/local/sbin**. Please use caution here as these directories may be unmounted during a shutdown and thus may prevent the **halt** script from calling **apcupsd** to turn off the UPS power. Though your data will be protected, in this case, your system will probably not be automatically rebooted when the power returns.

--*enable-powerflute*

This option enables the building of the **powerflute** executable, which is a ncurses based program to monitor the UPS. This program is not necessary for the proper execution of **apcupsd**.

--*enable-cgi*

This enables the building of the CGI programs that permit Web browser access to **apcupsd** data. This option is not necessary for the proper execution of **apcupsd**.

--*with-cgi-bin*=<path>

The *with-cgi-bin* configuration option allows you to define the directory where the cgi programs will be installed. The default is **/etc/apcupsd**, which is not necessarily what you want.

--*with-css-dir*=<path>

This option allows you to specify where you want **apcupsd** to put the Cascading Style Sheet that goes with the **multimoncss.cgi** CGI program.

--*enable-pthreads*

This option enables pthreads support causing **apcupsd** to be built as a threaded program rather than forking to create separate processes. **apcupsd** built in this fashion is more efficient than the standard version being one third the data size and less overhead locking and coping shared memory. This option is **highly** recommended for Windows builds.

--*with-libwrap*=<path>

This option when enabled causes **apcupsd** to be built with the TCP WRAPPER library for enhanced security.

In most cases, the <path> is optional since configure will determine where the libraries are on most systems.

--*with-nologin*=<path>

This option allows you to specify where **apcupsd** will create the nologin file when logins are prohibited. The default is **/etc**

--*with-pid-dir*=<path>

This option allows you to specify where **apcupsd** will create the process id (PID) file to prevent multiple copies from running. The default is system dependent but usually **/var/run**.

--*with-log-dir*=<path>

This option allows you to specify where **apcupsd** will create the EVENTS and STATUS log files. The default is **/etc/apcupsd**. This option simply sets the appropriate path in the **apcupsd.conf** file, which can be changed at any later time.

--*with-lock-dir*=<path>

This option allows you to specify where **apcupsd** will create the serial port lock file. The default is system dependent but usually **/var/lock**. This option simply sets the appropriate path in the **apcupsd.conf** file, which can be changed at any later time.

--*with-pwrfail-dir*=<path>

This option allows you to specify where **apcupsd** will create the **powerfail** file when a power failure occurs. The default is system dependent but usually **/etc**.

--*with-serial-dev*=<device-name>

This option allows you to specify where **apcupsd** will look for the serial device. The default is system dependent, but often **/dev/ttyS0**. This option simply sets the appropriate device name in the **apcupsd.conf** file, which can be changed at any later time.

--*with-nis-port*=<port>

This option allows you to specify what port **apcupsd** will use for the Network Information Server (the CGI programs). The default is system dependent but usually 7000. This option simply sets the appropriate port in the **apcupsd.conf** file, which can be changed at any later time.

`--with-nisip=<IP-Address>`

This option allows you to specify the value that will be placed on then NISIP directive in the configuration file. The default is 0.0.0.0. No checking is done on the value entered, so you must ensure that it is a valid IP address.

`--with-net-port=<port>`

This option allows you to specify what port **apcupsd** will use for the Master and Slave communications. The default is system dependent but usually 6666. This option simply sets the appropriate port in the **apcupsd.conf** file, which can be changed at any later time.

`--with-upstype=<type>`

This option allows you to specify the type of UPS that will be connected to your computer. The default is: **smartups**. This option simply sets the appropriate UPS type in the **apcupsd.conf** file, which can be changed at any later time.

`--with-upscable=<path>`

This option allows you to specify what cable you are using to connect to the UPS. The default is: **smart**. This option simply sets the appropriate UPS cable in the **apcupsd.conf** file, which can be changed at any later time.

`--disable-install-distdir`

This option modifies the **apcupsd** Makefiles disable installation of the distribution (platform) directory. Generally, this used to do a full installation of **apcupsd** except the final modification of the operating system files (normally /etc/rc.d/halt, etc.). This is useful if your operating system is not directly supported by **apcupsd** or if you want to run two copies of **apcupsd** on the same system. This option can also be used by those of you who prefer to manually reboot your system after a power failure or who do not want to modify your system halt files.

Recommended Options for most Systems

For most systems, we recommend the following options:

```
./configure --prefix=/usr --sbindir=/sbin
```

and you can optionally build and install the CGI programs as follows:

```
./configure --prefix=/usr --sbindir=/sbin --enable-cgi --with-cgi-bin=/home/httpd/cgi-bin
```

Compilers and Options

Some systems require unusual options for compilation or linking that the **./configure** script does not know about. You can specify initial values for variables by setting them in the environment. Using a Bourne-compatible shell, you can do that on the command line like this:

```
CFLAGS="-O2 -Wall" LDFLAGS= ./configure
```

Or on systems that have the **env** program, you can do it like this:

```
env CPPFLAGS=-I/usr/local/include LDFLAGS=-s ./configure
```

Or for example on the Sun Solaris system, you can use:


```
setenv CFLAGS -xO2
```

```
setenv LDFLAGS -O
```

```
./configure
```

Operating System Specifics

With the exception of Linux SuSe and Linux RedHat systems used by the developers, we rely on users to help create installation scripts and instructions as well as to test that **apcupsd** runs correctly on their system. As you can imagine, most of these people are system administrators rather than developers so they are very busy and don't always have time to test the latest releases. With that in mind, we believe that you will find that a lot of very valuable work has been already done to make your installation much easier (and probably totally automatic).

Below, you will find a list of Operating Systems for which we have received installation files:

[Alpha](#)
[Caldera](#)
[Debian](#)
[FreeBSD](#)
[HPUX](#)
[NetBSD](#)
[OpenBSD](#)
[RedHat](#)
[RedHat RPM Installation](#)
[Slackware](#)
[SuSE](#)
[Solaris](#)
[Unknown](#)
[Windows](#)

Alpha

The Alpha V4.0 version of **apcupsd** builds without compiler errors with gcc version 2.95.2. It is unlikely that the native Alpha compiler will work because of varargs differences. Unless you are a system guru, we recommend that you connect your UPS to the second serial port **/dev/tty01** to avoid conflicts with the console device.

```
DEVICE /dev/tty01
```

In addition, you should ensure serial port lock file in **apcupsd.conf** is defined as:

```
LOCKFILE /var/spool/locks
```

Unlike the Linux systems, the system halt routine is located in **/sbin/rc0**, so after the **make install**, please check that this file has been correctly updated.

The start/stop script can be found in:

/sbin/init.d/apcupsd

Caldera

This port appears to be complete, but we have had no feedback from users.

Debian

This port is complete and is operation by several users. Since Debian build and install procedures are somewhat particular, we have put the extra Debian information into the following two subdirectories:

`<src>/distributions/debian/examples/` and `<src>/distributions/debian/packageinfo`

You can also find the official Debian packages on the Debian site at:

<http://packages.debian.org/stable/admin/apcupsd.html>

<http://packages.debian.org/testing/admin/apcupsd.html>

<http://packages.debian.org/unstable/admin/apcupsd.html>

FreeBSD

This port is complete and is being used by several users. As of version 3.8.3, we do not recommend that you compile **apcupsd** with pthreads enabled. This is because the current FreeBSD implementation of pthreads runs as a single process, and thus is less efficient (consumes more CPU time) than the forking version of **apcupsd**. We hope to rectify this in a future version by using the FreeBSD LinuxThreads implementation of pthreads.

HPUX

We have no reports of testing this yet on version 3.8.4, but worked fine on 3.8.1

NetBSD

Submitted during development of 3.8.2, this should be a complete distribution.

Please read the comments on the pthreads implementation in the FreeBSD section above as they may apply equally to OpenBSD.

OpenBSD

Ensure that you read the distributions/openbsd/README file before running apcupsd. There are some critical differences in how the OpenBSD implemenation operates when the UPS batteries are exhausted. Failure to take this into account may result in the system not being fully halted when power is lost.

Please read the comments on the pthreads implementation in the FreeBSD section above as they may apply equally to OpenBSD.

RedHat Systems

RedHat systems are fully supported, and by following the standard installation instructions given above, you should experience few or no problems.

RedHat RPM Installation

For RedHat systems 6.0, 6.1, and 6.2, and 7.0, there are binary and source RPMs available. Follow standard procedures for installing them. Please note, that the neither the 6.x nor the 7.0 RPMs can be installed on a RedHat 7.1 system. These binary RPMs can be installed with:

```
rpm -Uhv <release>
```

where <release> is the release to be installed, and is typically something like **apcupsd-3.8.0.i386.rpm** (or perhaps **apcupsd-3.8.0-pre6.i386.rpm** for a pre-release).

IMPORTANTIf you are doing a binary RPM upgrade, please remove the previous version of **apcupsd** because for some unknown reason, the rpm does not always update the halt script. To do the upgrade, use the following two commands:

```
rpm -e apcupsd
rpm -Uhv <release>
```

After installation of the binary RPM, please verify carefully that **/etc/rc.d/init.d/halt** was properly updated and contains new script lines flagged with *****APCUPSD*****.

Since there is no standard location for **cgi-bin**, the rpm will place the binary CGI programs in the directory **/etc/apcupsd/cgi**. To actually use them, you must copy or move them to your actual cgi-bin directory, which on many systems is located in **/home/httpd/cgi-bin**.

Slackware

Slackware systems are fully supported, and by following the standard installation instructions given above, you should experience few or no problems.

SuSE

SuSE systems are fully supported, and by following the standard installation instructions given above, you should experience few or no problems.

Sun Solaris

Please read this before attempting to compile or install the beta software. It contains important information that will make your efforts easier.

If you find bugs, or run into problems that seem to be related to the version of Solaris that you run, please feel free to contact me by email, or through the development mailing list. I'll attempt to help with problems getting the beta running, although I can't promise a quick response.

As always, remember testing UPSes can be hazardous to you system, and, APCUPSD MAY CONTAIN BUGS THAT CAN DAMAGE YOUR SYSTEM AND DATA FILES! You must accept all responsibility for running this software. An unexpected power-off of a running system can be a disaster. As always, make backups of any critical information before you install this software.

Remember, we told you. we'll listen sympathetically if you lose data, but there will be nothing we can do to help you.

Sincerely,
Carl Erhorn <cerhorn@hyperion.com> <apcupsd-devel@ro.com>

Please read the general installation instructions given above before continuing on with these Solaris-specific instructions. Then come back and read this section before attempting to build the package.

For building the system, we suggest that you run the configure and make processes as your normal UNIX user ID. The **make install** must be run as root. But if your normal ID has an environment setup for using the c compiler, it's simpler to do that than setup root to have the correct environment.

Normally, we support the GCC compiler, but we have also attempted to support the Solaris workshop compilers and EGCS compilers. Please be aware that if you do not use GCC, you may experience a few problems.

Whichever compiler you do have, please insure that you can execute the compiler from the command line before running configure. If you do not have an environment setup to run the compiler first, configure will fail.

Before running ./configure, please be sure that you do not have /usr/ucb on your path. This may cause the ./configure to chose the wrong shutdown program. If ./configure detects that /usr/usb is on your path, it will print a warning message. Please follow the advice to avoid shutdown problems.

Your normal UNIX user ID must own the source tree directories, and you must have the normal development tools in your path. This includes make, the compiler, the M4 preprocessor, the linker, and ar or ranlib. If the user you are logged in as can compile and link a c program from a source file, then you have all the required tools available.

You will want to install the executables in a directory that remains mounted during the shutdown. Solaris will unmount almost everything except the root directories. Since the ability to power the UPS off requires access to the executable programs, they need to be in a directory that will never be unmounted. And since they should also be in a directory that normal users cannot get into, /sbin is the default. However, please be aware that if you want to follow Sun's filesystem conventions you would use the following:

```
./configure \
  --prefix=/opt/apcupsd \
  --sbindir=/etc/opt/apcupsd/sbin \
  --sysconfdir=/etc/opt/apcupsd \
  --with-cgi-bin=/opt/apcupsd/cgi-bin
```

The way to setup the /sbin directory as the executables directory is to pass configure the **---sbindir=/sbin** option. No other arguments should be required, and your setup and platform should be detected automatically by configure.

Once you have run configure, you will need to do a **make**. Once the make has completed with no errors, you must su to root to complete the install. After the su, you may not have a path to the make program anymore. In that case, you should do the **make install** step as:

```
/usr/ccs/bin/make install
```

Once the install completes, you must edit the `/sbin/rc0` script as detailed below, then exit from the `su`'ed shell.

In order to support unattended operation and shutdown during a power failure, it's important that the UPS remove power after the shutdown completes. This allows the unattended UPS to reboot the system when power returns by repowering the system. Of course, you need autoboot enabled for your system to do this, but all Solaris systems have this by default. If you have disabled this on your system, please re-enable it.

To get the UPS to remove power from the system at the correct time during shutdown, i.e., after the disks have done their final sync, we need to modify a system script. This script is `/sbin/rc0`.

We do not have access to every version of Solaris, but we believe this file will be almost identical on every version. Please let us know if this is not true.

At the very end of the `/sbin/rc0` script, you should find lines just like the following:

```
# unmount file systems. /usr, /var and /var/adm are not unmounted by umountall
# because they are mounted by rcS (for single user mode) rather than
# mountall.
# If this is changed, mountall, umountall and rcS should also change.
/sbin/umountall
/sbin/umount /var/adm >/dev/null 2>1/sbin/umount /var >/dev/null 2>1/sbin/umount /usr >/dev/null
echo 'The system is down.'
```

We need to insert the following lines just before the last `'echo'`:

```
#see if this is a powerfail situation
if [ -f /etc/apcupsd/powerfail ]; then
    echo
    echo "APCUPSD will power off the UPS"
    echo
    /etc/apcupsd/apccontrol killpower
    echo
    echo "Please ensure that the UPS has powered off before rebooting"
    echo "Otherwise, the UPS may cut the power during the reboot!!!"
    echo
fi
```

We have included these lines in a file called `rc0.solaris` in the `distributions/sun` subdirectory of the source tree. You can cut and paste them into the `/sbin/rc0` file at the correct place, or yank and put them using `vi` or any other editor. Note that you must be root to edit this file.

You must be absolutely sure you have them in the right place. If your `/sbin/rc0` file does not look like the lines shown above, do not modify the file. Instead, email a copy of the file to me, and I will attempt to figure out what you should do. If you mess up this file, the system will not shut down cleanly, and you could lose data. Don't take the chance.

This feature has only been tested with APC SmartUPS models. If you do not have a SmartUPS, you will be one of the first testers to try this feature. Please send me email to let me know if it works with your UPS model, what model you have, and if possible, the event logs located in `/etc/apcupsd`. We'd be very interested in your results, and would be glad to work with you to get this feature working correctly with all the APC models. A detailed description of the screen output during the shutdown would be very helpful if you see problems.

You will then need to make the normal changes to the `/etc/apcupsd/apcupsd.conf` file. This file contains the configuration settings for the package. It is important that you set the values to match your UPS model and cable type, and the serial port that you have attached the UPS to. I have used both `/dev/ttya` and `/dev/ttyb` with no problems. You should be sure that logins are disabled on the port you are going to use, otherwise you will not be able to communicate with the UPS. If you are not sure that logins are disabled for the port, run the 'admintool' program as root, and disable the port. The 'admintool' program is a GUI administration program, and required that you are running CDE, OpenWindows, or another XWindows program such as KDE.

Solaris EEPROM Changes

Solaris probes the serial ports during boot, and during this process, it toggles some handshaking lines used by the UPS. As a result, particularly for simple signalling "dumb" UPSes it seems to kick it into a mode that makes the UPS think it's either in a calibration run, or some self-test mode. Since at this point we are really not communicating with the UPS, it's pretty hard to tell what happened. But it's easy to prevent this, and you should. Disconnect the UPS, and boot the system. When you get to a login prompt, log in as root. Type the following command:

```
eeeprom com1-noprobe=true
```

or

```
eeeprom com2-noprobe=true
```

depending on which com port your UPS is attached to. Then sync and shutdown the system normally, reattach the UPS, and reboot. This should solve the problem. However, we have some reports that recent versions of Solaris (7 & 8) appear to have removed this eeprom option and there seems to be no way to suppress the serial port probing during boot.

At this point, you should have a complete installation. The daemon will load automatically at the next boot. Watch for any error messages during boot, and check the event logs in `/etc/apcupsd`. If everything looks OK, you can try testing the package by removing power from the UPS. NOTE! if you have a simple signaling UPS, please run the first power tests with your computer plugged into the wall rather than into the UPS. This is because simple signaling UPSes have a tendency to power off if your configuration or cable are not correct.

As a user, your input is very helpful in solving problems with the package, and providing suggestions and future directions for the development of the package. We are striving to provide a useful package that works across all platforms, and welcome your feedback.

Best regards, and thanks for your interest and help, The Apcupsd Development Team.

Unknown Operating System

During the `./configure`, if **apcupsd** does not find one of the systems for which it has specific installation programs, it will set the Operating System to **unknown** and will use the incomplete installation scripts that are in `<src>/distributions/unknown/`. You will be on your own, or you can ask the developers list (`apcupsd-devel@apcupsd.org`) for installation instructions. This directory also contains a hint file for **Linux From Scratch**, which could be helpful for other systems as well.

Windows Systems with CYGWIN Installed

If you have a binary release of the **Win32 apcupsd**, please see the instructions in the [Win32 section](#) of this manual.

If you wish to build from the source, and if you have CYGWIN version 1.3.2 and GCC 2.95.3–5 installed, it is possible to build the Win32 version of **apcupsd**. Note, **apcupsd** version 3.8.0 as distributed was built with CYGWIN version 1.1.2 but works fine when built and run with CYGWIN 1.1.5. Version 3.8.2 was built with CYGWIN version 1.3.1–1, and versions 3.8.3 and 3.8.4 of **apcupsd** are built with CYGWIN version 1.3.2. Please don't try any other versions of CYGWIN as there were known problems.

To date, the Win32 version has only been build on a Win98 SR2 system with the above CYGWIN environment and all the available CYGWIN tools loaded. In addition, the builds were done running under the **bash** shell. As time permits, we will experiment with other environments, and if any of you do build it from source, please let us know. The current CYGWIN environment was loaded using the CYGWIN setup.exe program, downloading ALL the latest binaries and installing them.

We recommend that you run the **./configure** command with the following options:

```
./configure \
  --prefix=/apcupsd \
  --sbindir=/apcupsd/bin \
  --sysconfdir=/apcupsd/etc/apcupsd \
  --with-pid-dir=/apcupsd/etc/apcupsd \
  --mandir=/apcupsd \
  --with-cgi-bin=/apcupsd/etc/apcupsd/cgi \
  --enable-pthreads
```

After which, you can do a:

```
make
```

And to install **apcupsd**, do:

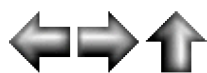
```
make install
```

During linking of **popup.exe** and **apcupsd.exe**, the following warning message is printed:

```
/USR/BIN/ld: warning: cannot find entry symbol _WinMainCRTStartup; defaulting to 00401000
```

This warning causes no harm. If there is some CYGWIN guru out there who knows how to eliminate this error, please contact us at: apcupsd-devel@apcupsd.org

Finally, you should follow the installation instructions in the [Win32 Installation](#) section of this document, skipping the part that describes unZipping the binary release.



Apcupsd Post Installation Configuration

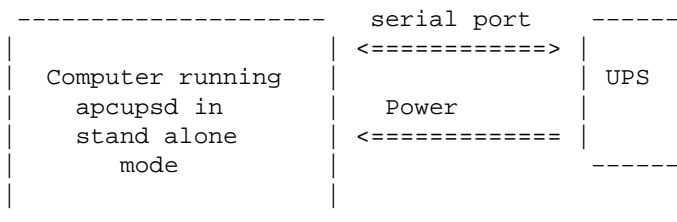
It may be necessary to change the configuration information in the file `/etc/apcupsd/apcupsd.conf` to meet your needs and to correspond to your configuration. This file is a plain ASCII file and you can use your favorite editor to change it. However, please take a careful look at the [installation chapter](#) in this document, because many of these directives can be correctly set during installation using the `./configure` program. This simplifies the task of editing the configuration file, and ensures that other files (CGI) are appropriately modified as well.

Three Major Configuration Possibilities for Apcupsd

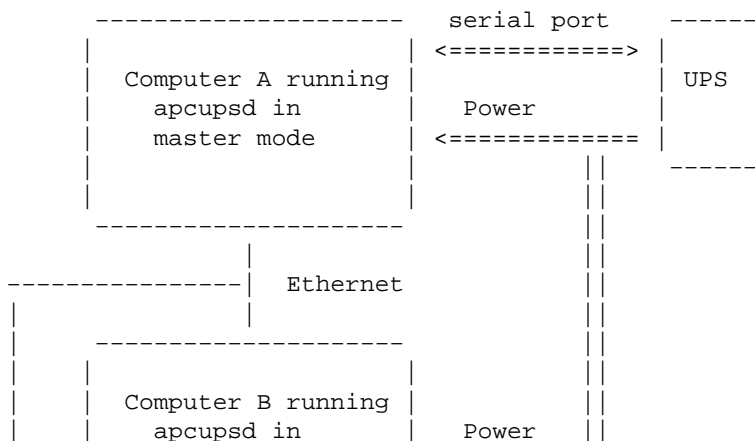
There are three major ways of running **apcupsd** on your system. The first is a standalone configuration where **apcupsd** controls a single UPS, which powers the computer. This is the most common configuration. The second configuration is a master/slave configuration, where one UPS powers several computers, each of which runs a copy of **apcupsd**. The computer that controls the UPS is called the master, and the other computers are called slaves. The master copy of **apcupsd** communicates with and controls the slaves via an ethernet connection. The third configuration (new with version 3.8.3), is where a single computer controls multiple UPSes. In this case, there are several copies of **apcupsd** on the same computer each controlling a different UPS. One copy of **apcupsd** will run in standalone mode, and the other copy or copies will normally run in master/slave mode.

These three possibilities can be represented by the following diagrams:

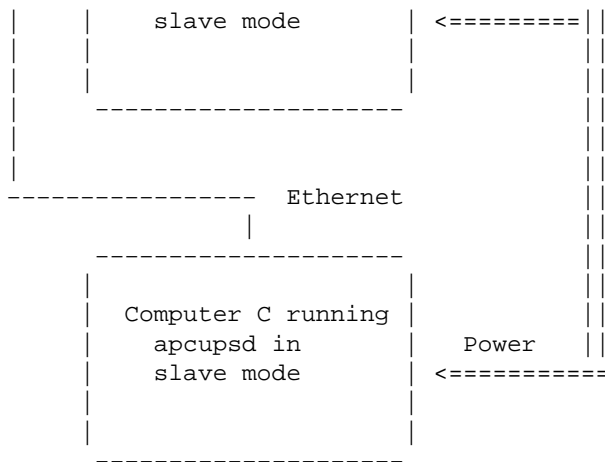
Stand Alone Configuration



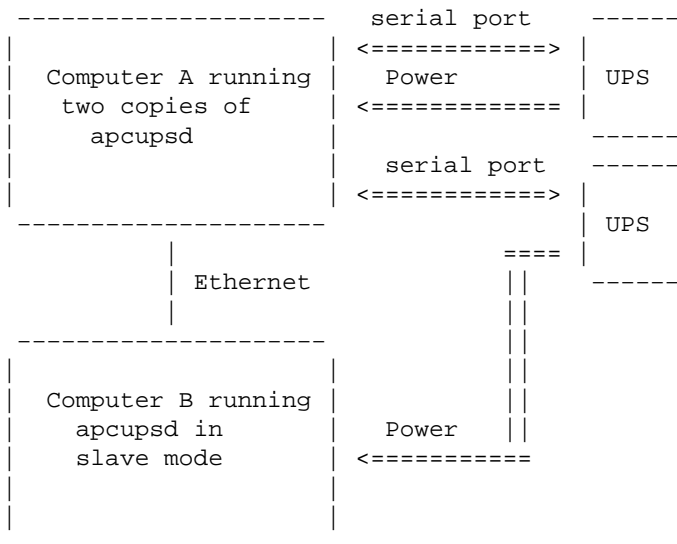
Typical Master/Slave Configuration



APC UPS management under Linux



Multi-UPS Configuration



If you wish to see some simple examples of possible configuration files, please see the [Configuration Examples Chapter](#) of this document.

For more details about the Multi-UPS Configuration, please see the Chapter entitled [Configuration for Controlling Multiple UPSes](#) in this manual.

Configuration Directives

Configuration directives in `/etc/apcupsd/apcupsd.conf` are:

General Configuration Directives

In general, each of these directives is required (the `DEVICE` directive is ignored for UPSCABLE ether).

UPSTYPE <type of APC UPS you have>

One of the big problems is understanding what kind of UPS you have and knowing what cable to use with what UPS. If you received a cable with your UPS, the cable number is stamped on the side of the connector (usually in the plastic on both cable ends), and it is most likely the correct cable for your UPS.

In the table below, we attempt to show what cables are known to work with each UPS. This information is a bit sketchy and so should not yet be considered definitive. Any comments or corrections would be appreciated.

The UPSTYPE directive can be defined during installation by using the **--with-upstype=** option on the **./configure** program.

Apcupsd UPSTYPE Keyword	APC Model	UPS Signaling	Cables Supported	Status
backups	BackUPS	Simple	*Simple-Custom, 940-0020B, 940-0020C, 940-0119A, 940-0023A	Supported
backups	BackUPS Office	Simple	940-0119A	Supported
backups	BackUPS ES	Simple	940-0119A	Supported (apparently identical to the BackUPS Office)
smartups	BackUPS CS (serial mode)	SubSmart	Smart (*Custom RJ45)	Supported using the Custom RJ45 cable
usb	BackUPS CS USB, BackUPS Pro USB	SubSmart	USB (using APC cables 940-0127A/B)	Supported in apcupsd version 3.9.4 and later
backups	BackUPS CS (serial mode)	Simple	940-0128A	Supported
sharebasic	ShareUPS Basic Port	Simple	940-0020B, 940-0020C, 940-0023A	Supported

APC UPS management under Linux

dumb	All models listed above	Simple	Cables listed above	Supported
backupspro	BackUPS Pro	SubSmart	940-0095A	Supported (note, see newbackupspro)
smartvsups	SmartUPS VS	SubSmart	940-0095A??	Supported
newbackupspro	Smarter BackUPS Pro	SubSmart	940-0095A	Supported
backupspropnp	Smarter BackUPS Pro	SubSmart	940-0095A	Supported
smartups	SmartUPS, PowerStack 450	Smart	*Smart-Custom, 940-0024C	Supported
apcsmart	SmartUPS	Smart	*Smart-Custom, 940-0024C	Supported
matrixups	MatrixUPS	Smart	*Smart-Custom, 940-0024C	Supported
sharesmart	ShareUPS Advanced Port	Smart	*Smart-Custom, 940-0024C	Unknown status
usb	SmartUPS USB	Smart	USB (using APC cable)	Supported in apcupsd version 3.9.8 and later

* => see the [Cables](#) chapter of this manual for instructions on how to build this cable (Simple-Custom, Smart-Custom, or Smart-RJ45).

UPSCABLE <type of cable you are using>>

[simple | 940-0020B | 940-0023A]

[smart | 940-0024B | 940-0024C]

[940-1524C | 940-0024G | 940-0095A | 940-0095B | 940-0095C | 940-0119A]

[ether | usb]

If you have a Smart or a SubSmart UPS (see table above), and you build your own cable, build a **Smart-Custom** cable (see the [Cables Chapter](#) of this manual). If you have a **Simple** UPS, build a **Simple-Custom** cable. If you have a BackUPS CS with a RJ45 connector, you can build your own **Custom-RJ45** cable.

The **--with-upscable=** option can be used on the

`./configure`

program to set this directive during the installation.

DEVICE *<name of device>*

Please specify which device is used for UPS communications (normally a serial port or a USB port). The default is platform dependent, and is usually something like `/dev/ttyS0`. For USB ports, you may specify a port range specification of the form `/dev/usb/hid/hidev[-0-9]`. Normally, the `./configure` program will set an appropriate default value, otherwise, you may also specify the **--with-serial-dev=** option on the `./configure` program to set this directive during the installation.

LOCKFILE *<path to lockfile>*

By supplying this argument, "apcupsd" tries to create a lockfile for the serial port in the specified directory. This is important to keep two programs from reading or writing the serial port at the same time. Please note that although the directive name is **LOCKFILE**, you are actually specifying the lock file path. Apcupsd automatically appends the name of the device when creating the file. On most systems, this directive is automatically set by the `./configure` program. You may also explicitly set it during the installation process by using the **--with-lock-dir=** option on the `./configure` program.

Configuration Directives Used by the Network Information Server

None of these directives are required for proper operation of **apcupsd**. For the Network Information Server to work, it must be enabled in the configuration (default) with **--enable-nis**.

NETSERVER *[on | off]*

This configuration directive turns the network information server on or off. If it is on, **apcupsd** will spawn a child process that serves **STATUS** and **EVENTS** information over the network. This information is currently used by the Web based CGI programs. The default is on. In some cases, for added security, you may want to invoke a separate information server daemon from the `inetd` daemon. In that case, **NETSERVER** should be **off**.

NISIP *<IP-address>*

This directive specifies an IP address on which NIS server will listen for incoming connections. Default value is 0.0.0.0 that means any incoming request will be serviced but if you want it to listen to a single subnet you can set it up to that subnet address, for example 192.168.10.0. Additionally you can listen for a single IP like 192.168.10.1. You may also use the **--with-nisip=** option on the `./configure` program to set this directive during the configuration phase.

This directive does not work on Win32 machines because `inet_ipton()` is not implemented there.

NISPORT *<port>*

This configuration directive specifies the port to be used by the **apcupsd** Network Information Server. The default is platform dependent, but typically 3551, which we have received from IANA as the official **apcupsd** networking port. If you change this port, you must manually change the `#define SERV_TCP_PORT` in `cgi/upsfetch.c` and rebuild the CGI programs. An alternative is to use the **--with-nis-port=** option on the `./configure` program during installation. In this case, all the appropriate locations will be automatically changed.

EVENTSFILE *<filename>*

If you want the **apcupsd** network information server to provide the last 10 events via the network, you must specify a file where **apcupsd** will save these events. The default is: `/etc/apcupsd/apcupsd.events`. Currently, **apcupsd** will save at most the last 50 events. Periodically (once an hour by default), **apcupsd** will check the size of this file. When more than 50 events are recorded, **apcupsd** will truncate the file to the most recent 10 events. Consequently this file will not

grow indefinitely. Although we do not recommend it, you may change these values by editing `apcevents.c` and changing the appropriate defines. Be aware that if you set these values to very large numbers, **apcupsd** may make excessive memory demands on the system during the data access and file truncation operations. < p>This filename may also be specified at installation time by using the **--with-log-dir=** option on the `./configure` program.

Configuration Directives used during Power Failures

In general, none of these directives are required. However, if you have a simple signaling (dumb) UPS with a cable that does not support the Low Battery signal, you must set the **TIMOUT** directive to force a shutdown. Please see the [Simple Signaling features Supported](#) section of this manual for more details.

ANNOY <time in seconds>

Specify the time in seconds between messages requesting logged in users to get off the system during a power failure. This timer starts only when the UPS is running on batteries. The default is 300 seconds (5 minutes). **apcupsd** sends the annoy messages by invoking the **apccontrol** script with the **annoyme** argument. The default is to send a **wall** message on Unix systems and a **popup** in Windows. The value of **ANNOYDELAY** must be greater than the value of **ANNOY** in order to receive annoy messages (this doesn't make sense, and means that the default values do not generate annoy messages --- KES).

Note that if **NOLOGON** is set to **disable** the annoy messages will also be disabled.

ANNOYDELAY <time in seconds>

Specify delay time in seconds before **apcupsd** begins requesting logged in users to get off the system during a power failure. This timer starts only after the UPS is running on batteries. This timer is reset when the power returns. The default is 60 seconds. Thus, the first warning to log off the system occurs after 60 seconds on batteries, assuming that **NOLOGON** is not set to **disable**.

NOLOGON <specifies when **apcupsd** should prevent user logins>

[**disable** | **timeout** | **percent** | **minutes** | **always**] are valid types.

The type specified allows you define the point when **apcupsd** will create the `/etc/nologin` file and thus when user logins are prohibited. Once the `/etc/nologin` file is created, normal users are prevented from logging in. Control of when this file is created is important for allowing systems with BIG UPSes to run as normally until the system administrator determines the need for preventing user logins. The feature also allows the system administrator to hold the "ANNOY" factor until the `/etc/nologin` file is created. The default is **always** if no **NOLOGON** directive is specified.

As far as I can tell, the only useful types are **disable** and **always** since the difference in the time when the logout warning is given and shutdown occurs for the other types is very short (KES).

disable

prevents **apcupsd** from creating the nologin file. Consequently, any user can login during a power failure condition. Also, the **ANNOY** feature is disabled so users will not be warned to logoff the system.

timeout

specifies that **apcupsd** should prohibit logins after the UPS is on batteries for 90% of the time specified on the **TIMOUT** configuration directive. Note! Normally you don't want to specify a **TIMOUT** value, so this option is probably not too useful (KES).

percent

specifies that **apcupsd** should prohibit logins when the remaining battery charge percentage reaches 110% or less than the value specified on the **BATTERYLEVEL** configuration directive. Thus if the **BATTERYLEVEL** is specified as 15, **apcupsd** will prohibit logins when the battery charge drops below 16% (15% X 110% = 16%).

minutes

specifies that **apcupsd** should prohibit logins when the remaining runtime in minutes reaches 110% or less than the value specified on the **MINUTES** configuration directive. Thus if **MINUTES** is set to 3, **apcupsd** will prohibit logins when the remaining runtime is less than 3 minutes ($3 \times 110\% = 3.3$).

always

causes **apcupsd** to immediately prohibit logins when a power failure occurs. This will also enable the ANNOY feature.

BATTERYLEVEL *<percent of battery>*

If **BATTERYLEVEL** is specified, during a power failure, **apcupsd** will halt the system when the remaining battery charge falls below the specified percentage. The default is 5 percent. This directive is ignored for simple signaling UPSes. To totally disable this counter, set **BATTERYLEVEL -1** in your **apcupsd.conf** file.

MINUTES *<battery runtime in minutes>*

If **MINUTES** is specified, during a power failure, **apcupsd** will shutdown the system when the remaining runtime on batteries as internally calculated by the UPS falls below the time specified. The default is 3. This directive is ignored for simple signaling UPSes. It should be noted that some UPSes report an incorrect value for remaining runtime when the battery is fully charged. This can be checked by examining the **TIMELEFT** value as printed in the output of an **apcaccess status** command. If the value is zero or otherwise unreasonable, your UPS is probably broken. In this case, we recommend that you disable this timer by setting **MINUTES -1** in your **apcupsd.conf** file.

TIMEOUT *<time in seconds>*

After a power failure, **apcupsd** will halt the system when **TIMEOUT** seconds have expired. A value of zero disables this timer. Normally for all Smart UPS models and dumb UPSes with cables that support low battery detection, this should be zero so that the shutdown time will be determined by the battery level and/or remaining runtime (see above) or in the case of a simple signaling UPS, when the battery is exhausted. This command is required for dumb UPSes that do not provide a battery exhausted signal (only testing can determine this point). For more information, see the [testing section](#) of this manual. This timer can also be useful if you want some slave machines to shutdown before other machines to conserve battery power. It is also useful for testing **apcupsd** because you can force a rapid shutdown by setting a small value (e.g. 60) and pulling the plug to the UPS.

When **apcupsd** is running in master mode (UPSCCLASS netmaster), and a shutdown condition is determined, **apcupsd** will notify each of the slaves to perform a shutdown then **apcupsd** will sleep for 30 seconds before issuing the shutdown of its own computer. If you need the master to wait additional time before shutting down (to allow for shutdown of slower slaves or of slaves running software that requires more time to shutdown — e.g. databases), you can do so by adding additional **sleep()** commands to **/etc/apcupsd/apccontrol** in each case that causes a shutdown.

TIMEOUT, **BATTERYLEVEL**, and **MINUTES** can be set together without problems. **apcupsd** will react to the first case or test that is valid. Normally SmartUPS users will set **TIMEOUT** to zero so that the system is shutdown depending on the percentage battery charge remaining (**BATTERYLEVEL**) or the remaining battery runtime (**MINUTES**).

KILLDELAY *<time in seconds>*

If **killdelay** is set, **apcupsd** will continue running after a shutdown has been requested, and after the specified time in seconds, **apcupsd** will attempt to shut off the UPS the power. This directive should normally be disabled by setting the value to zero, but on some systems such as Win32 systems **apcupsd** cannot regain control after a shutdown to force the UPS to shut off the power. In this case, with proper consideration for the timing, the **KILLDELAY** directive can be useful. Please be aware, if you cause **apcupsd** to kill the power to your computer too early, the system and the disks may not have been properly prepared. In addition, **apcupsd** must continue running after the shutdown is requested, and on Unix systems, this is not normally the case as the system will terminate all processes during the shutdown.

Configuration Directives used to Control System Logging

None of these directives are required.

STATTIME <time>

This directive supplies the time interval between writes to the STATUS file. If set to zero, the STATUS file will not be written. Please note that in a future version of **apcupsd** the STATUS file code will disappear since its functionality has been replaced by the Network Information Server and by **apcaccess status**, as a consequence, it is normally disabled by setting it to zero.

STATFILE <file>

This directive specifies the file to be used when writing the STATUS information. The default is `/etc/apcupsd/apcupsd.status`

DATETIME <time>

This directives supplies the time interval between writes of PowerChute™ like data information to the log file. See the [DATA format specification](#) section of this manual for additional details.

FACILITY <log-facility>

The facility directive can be used to change the system logging class or facility. The default is **DAEMON**. This parameter can be useful if you wish to direct the **apcupsd** system logging information to other than your system default files. See the [logging section](#) of this manual for additional details.

Configuration Directives for Sharing a UPS

The following directives apply to the master/slave networking mode of **apcupsd** where multiple machines can be powered by the same UPS. One machine, the master, will have a serial port connection to the UPS, and the other machines, the slaves, will obtain their information via the network from the master.

Note, as of version 3.10.x, the old master/slave code is by default turned off in the configuration. You must explicitly enable it by including a **--enable-master-slave** option on your `./configure` command before building the source.

In addition to the old master/slave code, there is now a new network driver enabled with **--enable-net** (default disabled) that can be used to control a slave from any version of **apcupsd** running NIS. This is a much more flexible system of controlling slaves because a slave machine that also has NIS turned on can thus act as a master for another slave with **--enable-net** turned on. With this mode turned on, the slave obtains the address of the master from the **DEVICE** directive, which takes the form **hostname[:port]** as a consequence, none of the directives apply for this form of networking. In addition, for this mode to work, you must specify **UPSTYPE net** so that the proper driver is loaded.

The remainder of this section presents directives that apply to the old master/slave code that must be enabled by the **enable-master-slave** configuration option.

UPSCCLASS <class of operation>

[standalone | shareslave | sharemaster] and
[netslave | netmaster] are valid types.
[standalone | netslave | netmaster] are tested classes.
[shareslave | sharemaster] classes are being tested.

The default is **standalone** and should be used for all machines powered by the UPS and having a

serial port connection to the UPS, but where there are no other computers dependent on power from the same UPS. This is the **normal** case.

Use **netmaster**, if and only if you have a serial port connection to the UPS and there are other machines deriving power from the same UPS. This is required in all master configuration files.

Use **netslave** if and only if you have no serial port connection to the UPS, but you derive power from it. This is required in all slave configuration files, and in this case, you will also have UPSCABLE set to **ether**.

Use **shareslave** if and only if you are using a ShareUPS and connected to a BASIC Port with Simple Signal. This code is not fully tested.

Use **sharemaster**, if and only if you are using a ShareUPS and connected to the ADVANCED Port Smart Signal control. This code is not fully tested.

UPSMODE [*disable* | *share* | *net* | *sharenet*] are valid types.

[*disable* | *net*] are the only known and tested classes.

[*share* | *sharenet*] classes are being tested.

For normal standalone operations, you will set **UPSMODE** to **disable** to indicate that you are disabling the master/slave networking.

However, if you are using a single UPS to power several computers and you have configured master and slave computers, then set this value to **net**.

Use *share* for two or seven (2/7) additional simple signal ports on a SmartAccessories(tm) (internal/external box) for SmartUPSes. The *share* and *sharenet* code is not fully tested.

NETTIME <*time in seconds*>

The interval in seconds that the master uses to send information to slave machines. This rate is automatically set to 1 second if the UPS goes on batteries and reset to your specified value when the mains power returns. A typical value might be 60 seconds.

NETPORT <*IP port number*>

This port number is used for communications in the master/slave networking code. Note that the master and each slave must have the same port number specified on the **NETPORT** directive in the configuration file. This port may also be specified during installation by using the **--with-net-port=** option on the **./configure** program.

The **NETPORT** should not be confused with the port number for the Network Information Server which is specified with the **SERVERPORT** configuration directive.

MASTER <*name of the master*> for Slave machines.

Used in slave configuration files, this is the network name of the master which is authorized to send commands to this slave. In all cases (of which I am aware), when you specify a **MASTER** directive, you will also specify **UPSCABLE ether** since your information about the UPS will come via the network from a master.

The slave machine will be shutdown whichever occurs first: either at the request of the master when it does a shutdown or when the values you have specified for **TIMEOUT**, **BATTERYLEVEL**, or **MINUTES** expire (these should work but have not been fully tested). Consequently, if you want the slaves to begin shutting down before the master, you can do so by adjusting the values in the configuration file. If you want the slave to remain up until the master shuts down, you should set **TIMEOUT**, **BATTERYLEVEL**, and **MINUTES** all to zero.

For proper functioning of the slave, you must specify the same **UPSTYPE** in the slave configuration file as is in the master configuration file.

It should be noted that the master and slaves continue to communicate over the network even after the master has issued a shutdown command to the slaves. This is because the master **apcupsd** continues

to run until it receives the shutdown signal from the system. This is important to ensure that all the slaves have been properly notified of the shutdown.

We recommend that the machine names used on the **MASTER** and **SLAVE** directives be put in your **/etc/hosts** file so that **apcupsd** will be able to resolve the machine name during startup and shutdown even if DNS is not running. Alternatively, you can use IP addresses on the **MASTER** and **SLAVE** directives, but this is less flexible.

SLAVE *<name of slave(s)> used only in MASTER configuration files.*

Used in master configuration files, this is the name of a slave machine that depends on this master.

There can be a maximum of 20 slaves attached to one master. Thus you can specify multiple **SLAVE** directives in a master configuration file. Only one slave name can be specified per **SLAVE** directive, thus for multiple slaves, specify multiple **SLAVE** directives.

As noted above the master and slaves continue to communicate over the network even after the master has issued a shutdown command to the slaves. This is because the master **apcupsd** continues to run until it receives the shutdown signal from the system. This is important to ensure that all the slaves have been properly notified of the shutdown.

We recommend that the machine names used on the **MASTER** and **SLAVE** directives be put in your **/etc/hosts** file so that **apcupsd** will be able to resolve the machine name during startup and shutdown even if DNS is not running. Alternatively, you can use IP addresses on the **MASTER** and **SLAVE** directives, but this is less flexible.

USERMAGIC *< user defined magic> used only in SLAVE configuration files.*

The **USERMAGIC** directive is a sort of password that gives a second level of identification security in a slave configuration file. It is a character string up to 17 characters in length. It should be unique for each slave. When the slave makes initial contact with the master, this string is passed to the master. Then on each transmission from the master to the slave, the string is passed back to the slave, which checks that it is the correct string before accepting the master's information. This string should be different for each and every slave on the network. This directive is not required.

Configuration Directives Used to Set the UPS EPROM

The values specified with the following directives are only used if the **--configure** option is specified on the **apcupsd** command line, and the UPS is capable of internal EPROM programming. In that case, **apcupsd** attempts to set the values into the UPSes EPROM.

Under normal operations, the values for these parameters specified in the configuration file are not used. Instead, they are read from the UPS EPROM by **apcupsd**. See the [EEPROM programming section](#) of this manual for further details before attempting to reprogram your EEPROM.

SENSITIVITY *<sets sensitivity level>*

(H)igh, (M)edium, (L)ow

This value determine how sensitive the UPS is to the mains quality and voltage fluctuations. The more sensitive it is, the quicker the UPS will switch to battery power when the mains line quality is bad. Normally, this should be set to H, but if you find your UPS switching to batteries frequently, you might want to try a less sensitive setting, providing that your computer equipment tolerates the poor quality mains. This value is written to the UPS EPROM when the **configure** option is specified.

Under normal **apcupsd** operations (no **--configure** option), **apcupsd** will read the value stored in the UPS and display it in the STATUS output.

WAKEUP <set wakeup delay>

The UPS power restart delay value in [0,60,180,300] in seconds after the UPS has shut down during a power failure. This is to prevent the power from coming back on too quickly after a power down, and is important for those who have high rpm drives that need to spindown before powering them up again. Some older SCSI models are very sensitive to this problem. Default is zero. This value is written to the UPS EPROM when the **---configure** option is specified. Under normal **apcupsd** operations (no **---configure** option), **apcupsd** will read the value stored in the UPS and display it in the STATUS output.

SLEEP <set sleep delay>

The UPS delay or grace period in [20,180,300,600] seconds before the UPS cuts the power to your equipment. The default is 20 seconds. This value is written to the UPS EPROM when the **---configure** option is specified. Under normal **apcupsd** operations (no **---configure** option), **apcupsd** will read the value stored in the UPS and display it in the STATUS output.

LOTTRANSFER <lower limit of ups batt. transfer>

This sets the low line voltage point at which to switch over to batteries. Different values are permitted based on the UPS model, classification, and manufacture date. Use **apcaccess eeprom** to show you which values are permitted. This value is written to the UPS EPROM when the **---configure** option is specified.

Under normal **apcupsd** operations (no **---configure** option), **apcupsd** will read the value stored in the UPS and display it in the STATUS output.

HITRANSFER <upper limit of ups batt. transfer>

This sets the high line voltage point to switch over to batteries.

Different values are permitted based on the UPS model, classification, and manufacture date. Use **apcaccess eeprom** to show you which values are permitted. This value is written to the UPS EPROM when the **---configure** option is specified.

Under normal **apcupsd** operations (no **---configure** option), **apcupsd** will read the value stored in the UPS and display it in the STATUS output.

RETURNCHARGE <min. batt. charge level>

This parameter specifies what battery percentage charge is necessary before the UPS will supply power to your equipment after a power down. Different values are permitted based on the UPS model, classification, and manufacture date. Use **apcaccess eeprom** to show you which values are permitted. This value is written to the UPS EPROM when the **---configure** option is specified.

Under normal **apcupsd** operations (no **---configure** option), **apcupsd** will read the value stored in the UPS and display it in the STATUS output.

BEEPSTATE <alarm beep state>

This parameter tells the UPS when it can sound its audio alarm. These settings are based on discrete events related to the remaining capacity of the UPS.

0

immediately upon power failure

T

power failure + 30 seconds /DD>

L

low battery power

N

never

UPSNAME <string>

This is an eight character string. This is the UPS name that will be stored in the UPS EPROM.

This value is written to the UPS EPROM when the **---rename-ups** option is specified. Under normal **apcupsd** operations (no **---configure** option), **apcupsd** will read the value stored in the UPS and display it in the STATUS output.

BATTDATE <string>

This is an eight character string that is the last date the batteries were changed.

This value is written to the UPS EPROM when the `--update-battery-date` option is specified. Under normal **apcupsd** operations (no `--configure` option), **apcupsd** will read the value stored in the UPS and display it in the STATUS output.



Configuration Examples

A Simple Configuration

You have a Smart UPS using the cable supplied by APC. A very simple configuration file would look like the following:

```
## apcupsd.conf v1.1 ##
UPSCABLE smart
UPSTYPE smartups
DEVICE /dev/ttyS0
LOCKFILE /var/lock
UPSCCLASS standalone
UPSMODE disable
```

Normally you would have many more configuration directives to completely customize your installation, but this example shows you the minimum required.

A Simple Master Configuration

You have a Smart UPS using the cable supplied by APC and you want it to act as a master for another computer, which is powered by the same UPS. A very simple configuration file would look like the following:

```
## apcupsd.conf v1.1 ##
UPSCABLE smart
UPSTYPE smartups
DEVICE /dev/ttyS0
LOCKFILE /var/lock
UPSCCLASS netmaster
UPSMODE net
NETTIME 10
NETPORT 6666
SLAVE slave1.mynetwork.com
SLAVE slave2.mynetwork.com
```

Note, the main difference from the stand alone configuration is that you have specified **UPSCCLASS netmaster** and **UPSMODE net**. In addition, you have specified one or more slave machines.

A Simple Slave Configuration

You have a Smart UPS using the cable supplied by APC that is connected to the master machine configured above. This slave machine has no serial port connection to the UPS, but is powered by the same UPS as the master. A very simple configuration file would look like the following:

```
## apcupsd.conf v1.1 ##
UPSCABLE ether
UPSTYPE smartups
LOCKFILE /var/lock
UPSCCLASS netslave
UPSMODE net
NETPORT 6666
MASTER master.mynetwork.com
```

The main difference from the master configuration is that you have specified **UPSCABLE ether** and **UPSCLASS netslave**. In addition, you have specified a single controlling master.

In this configuration, the shutdown will be initiated by the master. It is also possible to specify **BATTERYLEVEL**, **MINUTES**, and **TIMEOUT** configuration directives in the Slave machine that will cause the slave to shutdown before the master. This can often be useful if the slave is less important than the master and you wish to reduce battery power consumption so that the master can remain up longer during a power outage.

Variation on the Master/Slave Configuration

It is also possible to have a Master/Slave configuration where the Slave is powered by a different UPS (or any other power source), but is nevertheless controlled (i.e. shutdown) by the master. The setup would be identical to the Master/Slave configuration files shown above. The only difference is where the slave actually receives its power. In effect, **apcupsd** does not know or care where the power really comes from.



Cables

First, you will need a serial port cable between the APC UPS and your computer running **apcupsd**. You can use either the cable that came with your UPS (the easiest if we support it) or you can make your own cable. We recommend that you obtain a supported cable directly from APC.

If you already have an APC cable, you can determine what kind it is by examining the flat sides of the two connectors where you will find the cable number embossed into the plastic. It is generally on one side of the male connector.

To make your own cable, first, you must know whether you have a Smart UPS that sends ASCII characters called **Smart Signaling**, or a "dumb" UPS that uses serial port line voltage signaling, called **Simple Signaling**.

The "dumb" UPSes are older models such as the BackUPS (not BackUPS Pro) and the ShareUPS Basic Port that use **Simple Signaling**. Most other UPSes use **Smart Signaling**. If in doubt consult the [Configuration Section](#) of this manual, or the documentation that came with your UPS.

Smart Signaling Cable for SmartUPSes

If you must build your own cable, and you have a Smart UPS, we recommend building the cable as follows:

SMART-CUSTOM CABLE

Signal	Computer		UPS	
	DB9F		DB9M	
RxD	2	-----	2	TxD Send
TxD	3	-----	1	RxD Receive
GND	5	-----	9	Ground

When using this cable with **apcupsd** specify the following in **apcupsd.conf**:

```
UPSCABLE smart
UPSTYPE smartups
DEVICE /dev/ttyS0 (or whatever your serial port is)
```

Smart Signaling Cable for BackUPS CS Models

If you have a BackUPS CS, you are probably either using it with the USB cable that is supplied or with the 940-0128A supplied by APC, which permits running the UPS in dumb mode. By building your own cable, you can now run the BackUPS CS models (and perhaps also the ES models) using Smart signaling and have all the same information that is available as running it in USB mode.

The jack in the UPS is actually a 10 pin RJ45. However, you can just as easily use a 8 pin RJ45 connector, which is more standard (ethernet TX, and ISDN connector). >Below, you will find a diagram for the CUSTOM-RJ45 cable:

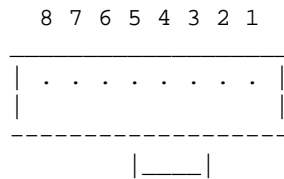
CUSTOM-RJ45 CABLE

Signal	Computer		UPS	UPS
	DB9F		RJ45-8	RJ45-10

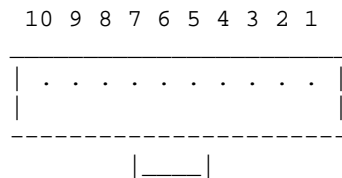
APC UPS management under Linux

RxD	2	-----	1	2	TxD	Send
TxD	3	-----	7	8	RxD	Receive
GND	5	-----	6	7		Ground

The RJ45-8 pins are: looking at the end of the connector:



The RJ45-10 pins are: looking at the end of the connector:



When using this cable with **apcupsd** specify the following in **apcupsd.conf**:

```

UPSCABLE smart
UPSTYPE smartups
DEVICE /dev/ttyS0 (or whatever your serial port is)
  
```

The information for constructing this cable was discovered and transmitted to us by slither_man. Many thanks!

Simple Signaling Cable for "dumb" UPSes

NOTE. YOU DO NOT HAVE THIS CABLE UNLESS YOU BUILT IT YOURSELF. THE SIMPLE-CUSTOM CABLE IS NOT AN APC PRODUCT.

For "dumb" UPSes using Simple Signaling, if you are going to build your own cable, we recommend to make the cable designed by the **Apcupsd** team as follows:

```

SIMPLE-CUSTOM CABLE

Signal Computer                UPS
      DB9F  4.7K ohm          DB9M
DTR    4  --[####]--*          DTR set to +5V by Apcupsd
      |
CTS    8  -----*-----  5  Low Battery
GND    5  -----  4  Ground
DCD    1  -----  2  On Battery
RTS    7  -----  1  Kill UPS Power
  
```

List of components one needs to make the Simple cable:

1. One (1) male DB9 connector, use solder type connector only.
2. One (1) female DB9/25F connector, use solder type connector only.
3. One (1) 4.7K ohm 1/4 watt 5% resistor.

4. resin core solder.
 5. three (3) to five (5) feet of 22AWG multi-stranded four or more conductor cable.
- Solder the resistor into pin 4 of the female DB9 connector.
 - Next bend the resistor so that it connects to pin 8 of the female DB9 connector.
 - Pin 8 on the female connector is also wired to pin 5 on the male DB9 connector. Solder both ends.
 - Solder the other pins, pin 5 on the female DB9 to pin 4 on the male connector; pin 1 on the female connector to pin 2 on the male connector; and pin 7 on the female connector to pin 1 on the male connector.
 - Double check your work.

We use the DTR (pin 4 on the female connector) as our +5 volts power for the circuit. It is used as the Vcc pull-up voltage for testing the outputs on any "UPS by APC" in Simple Signaling mode. This cable may not work on a BackUPS Pro if the default communications are Smart Signaling mode. This cable is also valid for "ShareUPS" BASIC Port mode and is also reported to work on SmartUPSes. However, the Smart Cable described above is much simpler. To have a better idea of what is going on inside **apcupsd**, for the SIMPLE cable **apcupsd** reads three signals and sets three:

Reads:

CD, which apcupsd uses for the On Battery signal when high.

CTS, which apcupsd uses for the Battery Low signal when high.

RxD (SR), which apcupsd uses for the Line Down signal when high. This signal isn't used for much.

Sets:

DTR, which apcupsd sets when it detects a power failure (generally 5 to 10 seconds after the CD signal goes high). It clears this signal if the CD signal subsequently goes low -- i.e. power is restored.

TxD (ST), which apcupsd clears when it detects that the CD signal has gone low after having gone high - i.e. power is restored.

RTS, which apcupsd sets for the killpower signal -- to cause the UPS to shut off the power.

Please note that these actions apply only to the SIMPLE cable, the signals used on the other cables are different.

Finally, here is another way of looking at the CUSTOM-SIMPLE cable:

APCUPSD SIMPLE-CUSTOM CABLE

Computer Side		Description of Cable	UPS Side	
DB9f	DB25f		DB9m	DB25m
4	20	DTR (5vcc) *below	n/c	
8	5	CTS (low battery) *below	<- 5	7
2	3	RxD (no line voltage) *below	<- 3	2
5	7	Ground (Signal)	4	20

APC UPS management under Linux

1	8	CD (on battery from UPS)	<-	2	3
7	4	RTS (kill UPS power)	->	1	8
n/c	1	Frame/Case Gnd (optional)		9	22

Note: the <- and -> indicate the signal direction.

Optional connections of original SIMPLE-CUSTOM specification that are not used.

		4.7K ohm		
DTR	4	--[####]--*		Note needed
RxD	2	-----*	3	Not used by Apcupsd

When using this cable with **apcupsd** specify the following in **apcupsd.conf**:

```
UPSCABLE simple
UPSTYPE backups
DEVICE /dev/ttyS0 (or whatever your serial port is)
```

Other APC Cables that Apcupsd Supports

Apcupsd will also support the following off the shelf cables that are supplied by APC:

940-0020B/C Simple Signal Only, all models.
 940-0023A Simple Signal Only, all models.
 940-0119A Simple Signal Only, Back-UPS Office, and BackUPS ES.
 940-0024[B/C/G] SmartMode Only, SU and BKPro only.
 940-0095[A/B/C] PnP (Plug and Play), all models.
 940-1524C SmartMode Only
 940-0127A/B USB Cables
 940-0128A Simple Signal Only, Back-UPS CS in serial mode.

Simple Signaling Features Supported by Apcupsd for Various Cables

The following table shows the features supported by the current version of **Apcupsd** (3.8.5 or later) for various cables running the UPS in Simple Signaling mode.

Cable	Power Loss	Low Battery	Kill Power	Cable Disconnected
940-0020B	Yes	No	Yes	No
940-0020C	Yes	Yes	Yes	No
940-0023A	Yes	No	No	No
940-0119A	Yes	Yes	Yes	No
940-0127A	Yes	Yes	Yes	No

940-0128A	Yes	Yes	Yes	No
940-0095A/B/C	Yes	Yes	Yes	No
simple	Yes	Yes	Yes	No

Simple UPS Signaling

Apparently, all APC signaling UPSes have the same signals on the output pins of the UPS. The difference at the computer end is due to different cable configurations. Thus, by measuring the connectivity of a cable, one can determine how to program the UPS. This is to be verified.

The signals presented or accepted by the UPS on its DB9 connector using the numbering scheme listed above is:

UPS Pin		Signal meaning
1	<-	Shutdown when set by computer for 1-5 seconds.
2	->	On battery power (this signal is normally low but goes high when the UPS switches to batteries).
3	->	Mains down (line fail) See Note 1 below.
5	->	Low battery. See Note 1 below.
6	->	Inverse of mains down signal. See Note 2 below.
7	<-	Turn on/off power (only on advanced UPSes only)

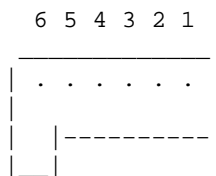
Note 1: these two lines are normally open, but close when the appropriate signal is triggered. In fact, they are open collector outputs which are rated for a maximum of +40VDC and 25 mA. Thus the 4.7K ohm resistor used in the Custom Simple cable works quite well.

Note 2: the same as note 1 except that the line is normally closed, and opens when the line voltage fails.

The Back-UPS Office 500 signals

The Back-UPS Office UPS has a telephone type jack as output, which looks like the following:

Looking at the end of the connector:



It appears that the signals work as follows:

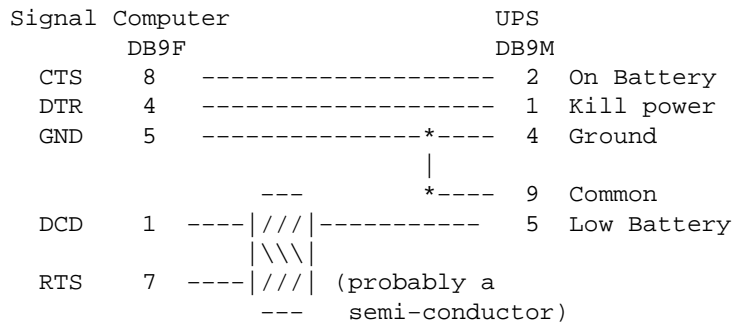
UPS		Signal meaning
1 (brown)	<-	Shutdown when set by computer for 1-5 seconds.
2 (black)	->	On battery power
3 (blue)	->	Low battery
4 (red)		Signal ground
5 (yellow)	<-	Begin signaling on other pins
6 (none)		none

940-0020B Cable Wiring

This diagram is for informational purposes and is not complete. Although we do not know what the black box semi-conductor contains, we believe that we understand its operation (many thanks to Lazar M. Fleysher for working this out).

This cable can only be used on simple signaling UPSes, and provides the On Battery signal as well as kill UPS power. Most recent evidence (Lazar's analysis) indicates that this cable under the right conditions may provide the Low Battery signal. This is to be confirmed.

APC Part# - 940-0020B

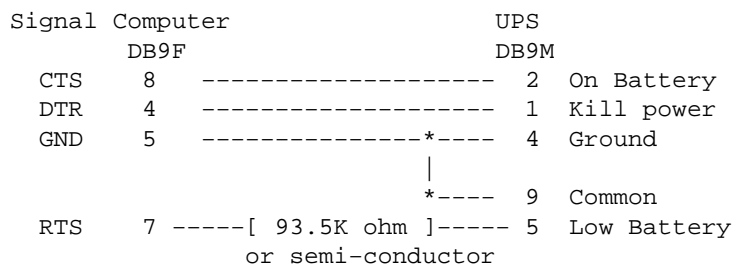


Thanks to Lazar M. Fleysher for proper

940-0020C Cable Wiring

This diagram is for informational purposes and may not be complete, we don't recommend that use it to build you build one yourself. This cable can only be used on simple signaling UPSes, and provides the On Battery signal, the Low Battery signal as well as kill UPS power. In **apcupsd** versions 3.8.2 and prior, please set your UPSCABLE to 940-0020B. In version 3.8.3 and later, you may specify the cable as 940-0020C. Please note that this diagram may not be accurate.

APC Part# - 940-0020C



940-0023A Cable Wiring

This diagram is for informational purposes and may not be complete, we don't recommend that use it to build you build one yourself. This cable can only be used on simple signaling UPSes, and apparently only provides the On Battery signal. As a consequence, this cable is pretty much useless, and we recommend that you find a better cable because all APC UPSes support more than just On Battery. Please note that we are not sure the following diagram is correct.

APC UPS management under Linux

* DTR is "cable power" and must be held at SPACE. DSR or CTS may be used as a loopback input to determine if the cable is plugged in.

* DCD is the "battery low" signal to the computer. A SPACE on this line means the battery is low. This is signalled by BATTERY-LOW being pulled down (it is probably open circuit normally).

Normally, the transistor is turned off, and DCD is held at the MARK voltage by TxD. When BATTERY-LOW is pulled down, the voltage divider R2/R1 biases the transistor so that it is turned on, causing DCD to be pulled up to the SPACE voltage.

* TxD must be held at MARK; this is the default state when no data is being transmitted. This sets the default bias for both DCD and SHUTDOWN. If this line is an open circuit, then when BATTERY-LOW is signalled, SHUTDOWN will be automatically signalled; this would be true if the cable were plugged in to the UPS and not the computer, or if the computer were turned off.

* RTS is the "shutdown" signal from the computer. A SPACE on this line tells the UPS to shut down.

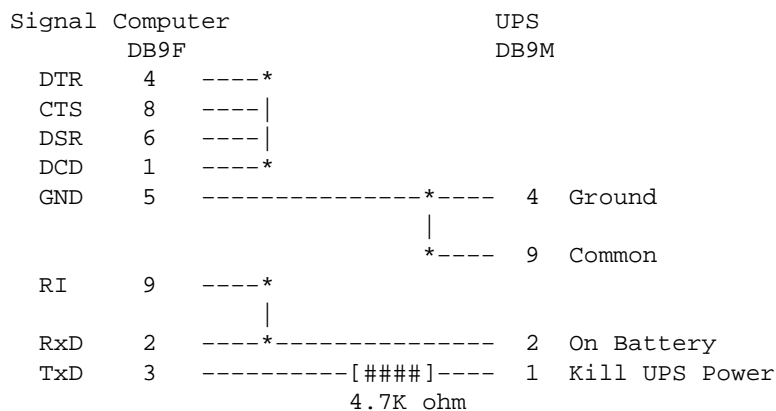
* RxD and RI are both the "power-fail" signals to the computer. A MARK on this line means the power has failed.

* SPACE is a positive voltage, typically +12V. MARK is a negative voltage, typically -12V. Linux appears to translate SPACE to a 1 and MARK to a 0.

940-0095B Cable Wiring

This diagram is for informational purposes and may not be complete, we don't recommend that use it to build you build one yourself.

APC Part# - 940-0095B



940-0119A Cable Wiring

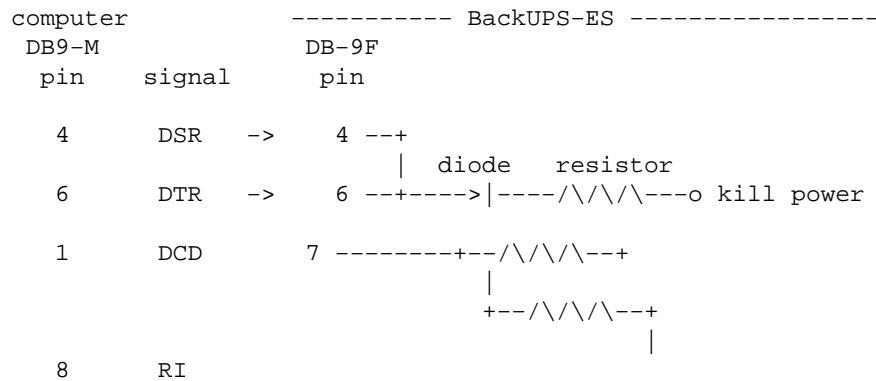
This diagram is for informational purposes and may not be complete, we don't recommend that use it to build you build one yourself. This cable is used with the BackUPS Office UPSes.

APC Part# - 940-0119A

UPS pins	Computer pins	Signal	Signal meaning
1 (brown)	4,6	DSR DTR	<- Shutdown when set by computer for 1-5 seconds.
2 (black)	8,9	RI CTS	-> On battery power
3 (blue)	1,2	CD RxD	-> Low battery
4 (red)	5	Ground	
5 (yellow)	7	RTS	<- Begin signaling on other pins
6 (none)	none		

BackOffice ES

The BackUPS ES has a straight through serial cable with no identification on the plugs. To make it work with **apcupsd**, specify the **UPSCABLE 940-0119A** and **UPSTYPE backups**. The equivalent of cable 940-0119A is done on a PCB inside the unit. Thanks to William Stock for supplying us with the information about the straight through cable, the PCB, and the following diagram:

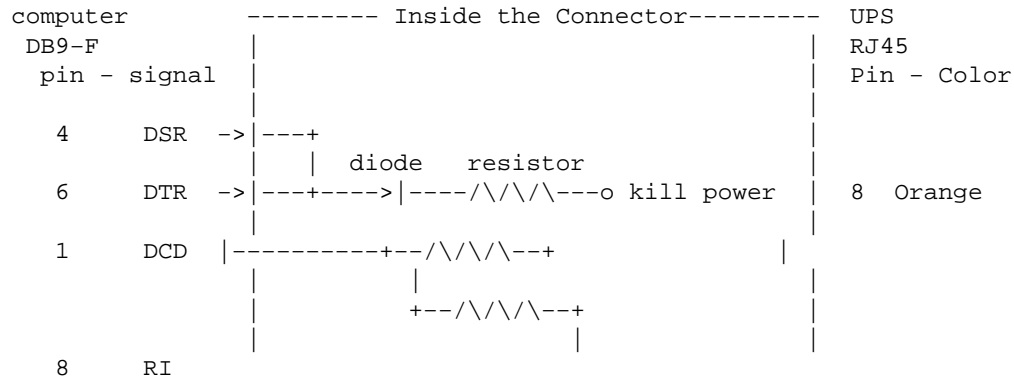


BackUPS ES and CS in Serial mode with Cable 940-0128A

Though these UPSes are USB UPSes, APC supplies a serial cable (typically with a green DB9 F connector) that has 940-0128A stamped into one side of the plastic serial port connector. The other end of the cable is a 10 pin RJ45 connector that plugs into the UPS (thanks to Dean Waldow for sending me a cable!). Apcupsd version 3.8.5 and later supports this cable when specified as **UPSCABLE 940-0128A** and **UPSTYPE backups**. However, running in this mode much of the information that would be available in USB mode is lost. In addition, when **apcupsd** attempts to instruct the UPS to kill the power, it begins cycling about 4 times a second between battery and line. The solution to the problem (thanks to Tom Suzda) is to unplug the UPS and while it is still chattering, press the power button (on the front of the unit) until the unit beeps and the chattering stops. After that the UPS should behave normally and power down 1-2 minutes after requested to do so.

An amazing discovery by slither_man allows one to build a CUSTOM-RJ45 cable (documented above) and run the BackUPS CS (and probably also the ES) in Smart mode. Running it this way provides all the same information that you would get by running it in USB mode. As a consequence, we recommend that you either purchase (where I don't know) or build your own CUSTOM-RJ45 cable rather than use the 940-0128A cable.

Thanks to all the people who have helped test this and have provided information on the cable wiring, our best guess for the cable schematic is the following:



Win32 Implementation Restrictions for Simple UPSes

Due to inadequacies in the Win32 API, it is not possible to set/clear/get all the serial port line signals. **apcupsd** can detect: CTS, DSR, RNG, and CD. It can set and clear: RTS and DTR.

This imposes a few minor restrictions on the functionality of some of the cables. In particular, LineDown on the Custom Simple cable, and Low Battery on the 0023A cable are not implemented.

Internal Apcupsd Actions for Simple Cables

This section describes how apcupsd 3.8.5 (March 2002) treats the serial port line signals for simple cables.

```

apcaction.c:
condition = power failure detected
cable = CUSTOM_SIMPLE
action = ioctl(TIOCMBIS, DTR)          set DTR (enable power bit?)

apcaction.c:
condition = power back
cable = CUSTOM_SIMPLE
action = ioctl(TIOCMBIC, DTR)          clear DTR (clear power bit)
action = ioctl(TIOCMBIC, ST)           clear ST (TxD)

apcserial.c:
condition = serial port initialization
cable = 0095A, 0095B, 0095C
action = ioctl(TIOMBIC, RTS)           clear RTS (set PnP mode)

cable = 0119A, 0127A, 0128A
action = ioctl(TIOMBIC, DTR)           clear DTR (killpower)
action = ioctl(TIOMBIS, RTS)           set  RTS (ready to receive)

apcserial.c:
condition = save_dumb_status
cable = CUSTOM_SIMPLE
action = ioctl(TIOMBIC, DTR)           clear DTR (power bit?)
action = ioctl(TIOMBIC, RTS)           clear RTS (killpower)

cable = 0020B, 0020C, 0119A, 0127A, 0128A
action = ioctl(TIOMBIC, DTR)           clear DTR (killpower)

```

APC UPS management under Linux

```
cable = 0095A, 0095B, 0095C
action = ioctl(TIOMBIC, RTS)      clear RTS (killpower)
action = ioctl(TIOMBIC, CD)       clear DCD (low batt)
action = ioctl(TIOMBIC, RTS)      clear RTS (killpower) a second time!
```

apcserial.c:

```
condition = check_serial
```

```
cable = CUSTOM_SIMPLE
action = OnBatt = CD
action = BattLow = CTS
action = LineDown = SR
```

```
cable = 0020B, 0020C, 0119A, 0127A, 0128A
action = OnBatt = CTS
action = BattLow = CD
action = LineDown = 0
```

```
cable = 0023A
action = Onbatt = CD
action = BattLow = SR
action = LineDown = 0
```

```
cable = 0095A, 0095B, 0095C
action = OnBatt = RNG
action = BattLow = CD
action = LineDown = 0
```

apcserial.c

```
condition = killpower
```

```
cable = CUSTOM_SIMPLE, 0095A, 0095B, 0095C
action = ioctl(TIOMCBIS, RTS)    set RTS (kills power)
action = ioctl(TIOMCBIS, ST)     set TxD
```

```
cable = 0020B, 020C, 0119A, 0127A, 0128A
action = ioctl(TIOMCBIS, DTR)    set DTR (kills power)
```

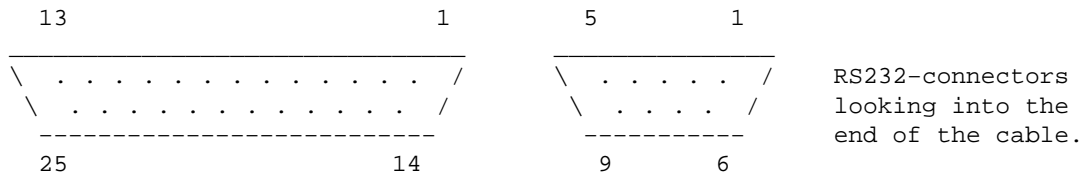
RS232 Wiring and Signal Conventions

DB-25 Pin #	DB-9 Pin #	Name	DTE-DCE Description
1	---	FG	--- Frame Ground/Chassis GND
2	3	TD	----> Transmitted Data, TxD
3	2	RD	<---- Received Data, RxD
4	7	RTS	----> Request To Send

5	8	CTS	<---- Clear To Send
6	6	DSR	<---- Data Set Ready
7	5	SG	----- Signal Ground, GND
8	1	DCD	<---- Data Carrier Detect
9	---	---	---- Positive DC test voltage
10	---	---	---- Negative DC test voltage
11	---	QM	<---- Equalizer mode
12	---	SDCD	<---- Secondary Data Carrier Detect
13	---	SCTS	<---- Secondary Clear To Send
14	---	STD	----> Secondary Transmitted Data
15	---	TC	<---- Transmitter (signal) Clock
16	---	SRD	<---- Secondary Receiver Clock
17	---	RC	----> Receiver (signal) Clock
18	---	DCR	<---- Divided Clock Receiver
19	---	SRTS	----> Secondary Request To Send
20	4	DTR	----> Data Terminal Ready
21	---	SQ	<---- Signal Quality Detect
22	9	RI	<---- Ring Indicator
23	---	---	----> Data rate selector

24	--	--	<---- Data rate selector
25	--	TC	<---- Transmitted Clock

Pin Assignment for the Serial Port (RS-232C), 25-pin and 9-pin, Female End



The diagram above represents the Female end of the cable. The male end is the same, but looking from inside the cable.

DTE : Data Terminal Equipment (i.e. computer)
 DCE : Data Communications Equipment (i.e. UPS)
 RxD : Data received; 1 is transmitted "low", 0 as "high"
 TxD : Data sent; 1 is transmitted "low", 0 as "high"
 DTR : DTE announces that it is powered up and ready to communicate
 DSR : DCE announces that it is ready to communicate; low=modem hang-up
 RTS : DTE asks DCE for permission to send data
 CTS : DCE agrees on RTS
 RI : DCE signals the DTE that an establishment of a connection is attempted
 DCD : DCE announces that a connection is established

ioctl to RS232 Correspondence

```

#define TIOCM_LE      0x001
#define TIOCM_DTR     0x002
#define TIOCM_RTS     0x004
#define TIOCM_ST      0x008
#define TIOCM_SR      0x010
#define TIOCM_CTS     0x020
#define TIOCM_CAR     0x040
#define TIOCM_RNG     0x080
#define TIOCM_DSR     0x100
#define TIOCM_CD      TIOCM_CAR
#define TIOCM_RI      TIOCM_RNG
#define TIOCM_OUT1    0x2000
#define TIOCM_OUT2    0x4000
  
```



Invoking Apcupsd

The simplest way to invoke **apcupsd** is from the command line by entering:

```
/sbin/apcupsd
```

To do so, you must be root. However, normally, you will want **apcupsd** started automatically when your system boots. On some systems with installation support (e.g. S.u.S.E and RedHat), the installation procedure will create a script file that you will be automatically invoked when your system reboots. On other systems, you will have to invoke **apcupsd** from your **rc.local** script.

Invoking Apcupsd on RedHat systems

On RedHat systems, this script file that automatically invokes **apcupsd** on system start and stops is:

```
/etc/rc.d/init.d/apcupsd
```

To start **apcupsd** manually (as you will probably do immediately following the installation), enter the following:

```
/etc/rc.d/init.d/apcupsd start
```

To understand how this file is automatically invoked at system startup and shutdown, see the man pages for **chkconfig**.

Invoking Apcupsd on SuSE systems

On SuSE systems, the script file that automatically invokes **apcupsd** on system start and stops is:

```
/etc/rc.d/apcupsd
```

To start **apcupsd** manually (as you will probably do immediately following the installation), enter the following:

```
/etc/rc.d/apcupsd start
```



Stopping Apcupsd

Normally, when properly installed, **apcupsd** will be started and stopped automatically by your system. Unfortunately, the details are different for each system. Below, we give the commands for selected systems. Alternatively, there are simple **stopapcupsd** and **startapcupsd** scripts in the **examples** directory, or you can modify one of the scripts in the **distributions** directory to meet your needs.

To stop **apcupsd** you can do the following:

On RedHat systems:

```
/etc/rc.d/init.d/apcupsd stop
```

On SuSE systems:

```
/etc/rc.d/apcupsd stop
```



Testing Apcupsd

Simple Signaling UPSes (sometimes called Dumb UPSes)

The following testing procedures apply only to Smart UPSes. If you have a simple signaling UPS such as a BackUPS or other non-Smart UPS, your testing procedures will be somewhat different (Smart UPSes include SmartUPS and MatrixUPS models). For example, for simple signaling (Dumb) UPSes, **apcupsd** is not currently able to detect whether or not the serial cable is connected. In addition, some simple signaling UPSes with certain cable combinations are not able to detect the low battery condition. For more details please see the [Cable Chapter](#) of this manual, and more specifically, the [Cable Modes section](#). Many of the other testing features should work similar to described below. However, since it is easy to configure the cable incorrectly and thus have premature shutdowns of the UPS power, we **strongly** recommend, especially for simple signaling (Dumb) UPSes, that you do most of the initial testing with your computer plugged into the wall rather than your UPS. Thus if the UPS power is suddenly shutoff, your computer will continue to run. We also recommend using the safe-apccontrol as described below, until you are sure that the signaling is correct.

Please note that if launch the execution of **apcupsd** while your simple signaling UPS is on battery power, it is very likely that your UPS will immediately shut off the power. This is due to the initialization of the serial port line signals.

Determining Which Simple Signaling Cable You Have

For a simple signaling (dumb) UPS, it is important to know what cable you have. All cables furnished by APC have the cable number stamped on the side of the computer connector end of the cable. Using this number with **apcupsd** will normally work fine.

If you do not know what cable you have, you can use the **apctest** program to determine the type of cable that you have. Please see the [apctest Chapter](#) of this manual for the details of running **apctest**.

Smart UPSes

Most of rest of this chapter concerns testing Smart UPSes. As noted above many of these tests will apply with certain restrictions to Simple Signaling (dumb) UPSes.

USB port

Please see the [USB Chapter](#) of this manual for details of testing connections to a USB port. Once you have the UPS and the computer connected, most of the testing procedures described here apply equally well to USB UPSes.

Checking the Installation

Before continuing, please first read the section [Checking the Installation](#) in the Installation Section of this manual.

Establishing Serial Port Connection

Once you have compiled, installed, and invoked **apcupsd**, you should wait to allow **apcupsd** to configure itself and establish contact with the UPS.

If you see the following message about 30 seconds after starting **apcupsd**:

```
apcupsd FATAL ERROR in apcserial.c at line 156
PANIC! Cannot communicate with UPS via serial port.
```

it means that **apcupsd** tried for about 30 seconds to establish contact with the UPS via the serial port, but was unable to do so. Before continuing, you must correct this problem. Some of the possible sources of the problem are:

- You have not configured the correct serial port name on the DEVICE directive in your **/etc/apcupsd/apcupsd.conf** configuration file.
- The serial port that you have chosen has logins enabled. You must disable logins on that port, otherwise, the system prevents **apcupsd** from using it. Normally, the file **/etc/inittab** specifies the ports for which a **getty** process is started (on Sun machines, the serial port program equivalent to **getty** is called **ttymon**). You must disable this for the port that you wish to use.
- Make sure you are doing your testing as **root** otherwise, you may have permissions problems accessing the serial port.
- You may have cabling problems, either with an incorrect cable, or the incorrect cable specification directive in the configuration file.
- You may have a problem with the **/etc/apcupsd/apcupsd.conf** file. For example, check that you have specified the correct type of UPS and the correct networking directives. For more details, see the [Configuration Section of this manual](#).
- If you have a SmartUPS 5000 RM 15U or similar model, that comes with a "Web/SNMP management card" in one of the "Smart Slots", this card may interfere with the serial port operation. If you are having problems, please remove this card and try again. Supposedly V3.0 of the card firmware has been corrected to properly release the serial port.
- Ensure that you have no other programs that are using the serial port. One user reported that he had problems because the serial port mouse (gpm) was using the same port as **apcupsd**. This causes intermittent seemingly random problems.
- If you are using a WinNT or Win2000 machine, the OS is probably attempting to attach a serial mouse to the port you are using (COM1 or COM2). To prevent this, edit your **c:\boot.ini** file, and you will find a line that looks something like the following:

```
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Workstation Version 4.00"
```

Add the following to the end of the line: **/NoSerialMice:COM1** (or **COM2**) so that the new line looks like:

```
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Workstation Version 4.00"
/NoSerialMice:COM1
```

- If you are using a WinNT or Win2000 machine, try connecting **apcupsd** to COM2 rather than COM1 (be sure to change your **c:\apcupsd\etc\apcupsd\apcupsd.conf** to reflect the change).
- If you are using a Solaris machine, you may have similar problems as described above for the WinNT machine. A possible fix is documented in the Sun section of the Configuration chapter of this manual.

- Try connecting your UPS to another machine. If it works, then you probably have a bad serial port card. As unlikely as this may sound, at least two of our users have had to replace bad serial port cards.
- Try doing an **lsot /dev/ttyS0** where you replace the **/dev/ttyS0** with your serial port name. If you get no output, the port is free (or there is no physical port). If you get output, then another program is using the port, and you should see which one.
- Try doing a **dmesg | grep tty**. This may show you if a program has grabbed the port. (Thanks to Joe Acosta for the suggestion.)
- If all else fails, make sure your system is configured for serial port support.
- If you are running Linux, check your **/proc** file system. For example: **cat /proc/devices** should print something like **4 ttyS** if you have a serial port. If your serial port is working, a **cat /proc/interrupts** should show the serial port usage (e.g. **4: 294553 XT-PIC serial**). Also, **cat /proc/ioports** should show up something like **03f8-03ff : serial(auto)**. Or, **cat /proc/tty** should print a line like **serial /dev/ttyS 4 64-127 serial**. Finally, a **cat /proc/tty/driver/serial** should print something like the following:
serinfo:1.0 driver:5.05c revision:2001-07-08
0: uart:16550A port:3F8 irq:4 baud:9600 tx:1503168 rx:1461721 fe:8 RTS|DTR|RI

The first thing to do is to look at your log file, usually **/var/log/messages** because **apcupsd** writes more detailed information to the log file whenever there is an error.

If you have a UPS that uses SubSmart or Smart protocol (see the [Configuration](#) section for a list of the UPSes using these protocols), you can manually test the serial communications with the UPS by starting a serial port communications program (such as **minicom**, **tip**, or **cu**) with the settings 2400 8N1 (2400 baud, 8 bits, no parity, 1 stop bit). Be extremely careful what you send to your UPS as certain characters may cause it to power down or may even cause damage to the UPS. Try sending an upper case Y to the UPS (without a return at the end). It should respond with SM. If this is not the case, review the possible problems listed above. If you fat finger the Y and enter y instead, no cause for alarm, you will simply get the APC copyright notice.

Once you are sure that serial port communications is working, proceed to the next test.

PS Test

After you start **apcupsd**, execute the following command:

```
ps fax
```

or the equivalent for your system. If you are running on Linux and using the fork()ing version of **apcupsd**, you should something similar to the following output.

```
4492 ?      S      0:00 apcmain      -f /etc/apcupsd/apcupsd.conf
4496 ?      S      0:00  \_ apcser      -f /etc/apcupsd/apcupsd.conf
4497 ?      S      0:00  \_ apcnis      -f /etc/apcupsd/apcupsd.conf
```

This indicates that **apcupsd** is up and running and has started the two (default) child processes.

apcmain

is the main program that waits until it receives a termination signal (SIGTERM) or one of the child processes dies.

apcser

is the process that manages the serial port and takes any actions (generates events) that are necessary as a result of a change of state of the UPS.

apcnis

is the Network information server process that provides EVENTS and STATUS information over the network. This information is used by the CGI programs.

If you are running on a non-Linux system, or using pthreads on a Linux system (recommended), your output will probably not show the names of the processes and will appear more like the following:

```
632 ?      S      0:00 /sbin/apcupsd -f /etc/apcupsd/apcupsd.conf
841 ?      S      0:00 \_ /sbin/apcupsd -f /etc/apcupsd/apcupsd.conf
842 ?      S      0:00 \_ /sbin/apcupsd -f /etc/apcupsd/apcupsd.conf
```

Logging Test

Once you have established that the proper processes are running, do a tail of the system log file, normally **/etc/var/messages**:

```
tail /etc/var/messages
```

You should see output that looks similar to the following:

```
Dec 5 17:01:05 matou apcupsd[5917]: apcupsd 3.7.2 startup succeeded
```

And if you have configured the network information server, you should also see:

```
Dec 5 17:01:05 polymatou apcupsd[5975]: apcserver startup succeeded
```

These messages should also appear in the temporary [EVENTS](#) file (**/etc/apcupsd/apcupsd.events**) if you are using the default configuration file.

apcaccess Test

This test consists of running **apcaccess** to see if **apcupsd** is properly updating its internal variables. Please note that if you are running a pthreaded version of **apcupsd** (installed from rpm or **--enable-pthreads** on the **./configure** line), you must enable the **apcupsd** Network Information Server in your configuration file for **apcaccess** to work.

To run the apcaccess test, use the following command:

apcaccess status

Depending on the type of UPS you have, you will get slightly different output, but an example For a Smart-UPS is as follows:

```
APC      : 001,048,1088
DATE     : Fri Dec 03 16:49:24 EST 1999
HOSTNAME : daughter
RELEASE  : 3.7.2
CABLE    : APC Cable 940-0024C
MODEL    : APC Smart-UPS 600
UPSMODE  : Stand Alone
UPSNAME  : SU600
LINEV    : 122.1 Volts
```


APC UPS management under Linux

```
MAXLINEV : 123.3 Volts
MINLINEV : 122.1 Volts
LINEFREQ : 60.0 Hz
OUTPUTV  : 122.1 Volts
LOADPCT  : 32.7 Percent Load Capacity
BATTV    : 26.6 Volts
BCHARGE  : 095.0 Percent
MBATTCHG : 15 Percent
TIMELEFT : 19.0 Minutes
MINTIMEL : 3 Minutes
SENSE    : Medium
DWAKE    : 000 Seconds
DSHUTD   : 020 Seconds
LOTRANS  : 106.0 Volts
HITRANS  : 129.0 Volts
RETPCT   : 010.0 Percent
STATFLAG : 0x08 Status Flag
STATUS   : ONLINE
ITEMP    : 34.6 C Internal
ALARMDEL : Low Battery
LASTXFER : Unacceptable Utility Voltage Change
SELFTEST : NO
STESTI   : 336
DLOWBATT : 05 Minutes
DIPSW    : 0x00 Dip Switch
REG1     : N/A
REG2     : N/A
REG3     : 0x00 Register 3
MANDATE  : 03/30/95
SERIALNO : 13035861
BATTDATA : 05/05/98
NOMOUTV  : 115.0
NOMBATTV : 24.0
HUMIDITY : N/A
AMBTEMP  : N/A
EXTBATTs : N/A
BADBATTs : N/A
FIRMWARE : N/A
APCMODEL : 6TD
END APC  : Fri Dec 03 16:49:25 EST 1999
```

For a simple signaling or dumb UPS such as BackUPS, your output will be very minimal as follows:

```
APC      : 001,012,0319
DATE     : Mon Feb 18 09:11:50 CST 2002
RELEASE  : 3.8.5
UPSNAME  : UPS_IDEN
CABLE    : APC Cable 940-0128A
MODEL    : BackUPS
UPSMODE  : Stand Alone
STARTTIME: Mon Feb 18 09:11:45 CST 2002
LINEFAIL : OK
BATTSTAT : OK
STATFLAG : 0x008 Status Flag
END APC  : Mon Feb 18 09:15:01 CST 2002
```

If you see the above output, it is a good sign that **apcupsd** is working. Assuming that the output looks reasonable, check the following variables:

LINEV

This is the line voltage and it should be a value that is appropriate for your equipment. In the USA, it is typically about 120 Volts while in Europe, it is about 220 Volts.

BATTV

Unless you have additional battery packs, this should be near 24 Volts plus or minus 5 Volts.

STATUS

This is the status of the UPS and it should normally be **ONLINE**.

If you see a message to the effect of:

```
attach_shmarea: shared memory version mismatch (or UPS not yet ready to report)
```

or if all the displayed values are zero, you have not waited long enough. Wait a bit longer and then re-execute the **apcaccess status** command.

If you see a message to the effect of:

```
APCACCESS FATAL ERROR in apcaccess.c at line 336
tcp_open: cannot connect to server localhost on port 7000.
```

It means that you have probably not enabled NIS in **apcupsd**.

Communications Test

At this point, you should ensure that **apcupsd** is handling the serial port correctly. This test assumes you have a Smart UPS. If you have a simple signaling UPS, please skip to the next section (Simulated Power Fail Test).

When **apcupsd** detects a problem, it generates an EVENT, which consists of sending a message to the system log then invoking the **apccontrol** script (normally in **/etc/acpupsd/apccontrol**) to handle the event.

In order to create an event, remove the serial port plug from the back of your computer or from the back of the UPS. Within 6 seconds, **apcupsd** should detect the lack of serial port communications and broadcast a **wall** message indicating that the serial port communications was lost:

Warning serial port communications with UPS lost.

At the same time, it sends the same message to the system log and to the temporary EVENTS file (**/etc/apcupsd/apcupsd.events**).

Plug the serial port plug back into your computer, and within about 12 seconds, **apcupsd** should reestablish communications and broadcast and log the following message:

Serial communications with UPS restored.

If these messages are logged but not broadcast, either you have your **mesg** permission set to **no** (see **man wall**) or there is a problem with **apccontrol**. If you are running a window manager such as GNOME and don't have a console window open, you may not receive the **wall** messages. However, you should find them in your system log file (normally **/var/log/messages** and in the temporary EVENTS file, **/etc/apcupsd/apcupsd.events**. For example, to observe these events in the temporary EVENTS file, you might do a

```
tail -f /etc/apcupsd/apcupsd.events
```

before running the test.

If you do not observe these messages, you should correct this problem before proceeding with additional tests.

Simulated Power Fail Test

At this point, you should verify that in the event of a power fail **apcupsd** properly calls **apccontrol**. This test is appropriate for all models of UPSes (Smart or dumb).

To avoid the possibility that **apcupsd** might shutdown your system, locate where **apccontrol** resides on your system (normally, **/etc/apcupsd/apccontrol**). Move this script to another location e.g. **apccontrol.save** and replace it with the script found in **examples/safe.apccontrol**. When that is done, ensure that your UPS battery is fully charged and that you have at least 5 minutes of remaining runtime on the batteries. This can be done by examining the values of the **BATTCHG** and **TIMELEFT** variables in the printout of **apcaccess status**.

Although this should not be necessary, as an extra precaution, you can shutdown your machine, remove the plug from the UPS you are testing, and plug your machine into another UPS or directly into the wall. Doing so, will ensure that the UPS doesn't cut the power to your machine at a bad time. Remember at the end of the testing to plug your machine back into the UPS.

To begin the test, pull the power plug from the UPS. The first time that you do this, psychologically it won't be easy, but after you have pulled the plug a few times, you may even enjoy it as I do. If all goes well, **apcupsd** should detect the power failure and print several warning messages. The first should appear after 5 to 6 seconds and read:

Warning power loss detected.

Then generally 6 seconds later, **apcupsd** is sure that it isn't a transient effect, so it sends:

Power failure. Running on UPS batteries.

After a few more seconds (total around 15 seconds), plug the power cord back in and ensure that **apcupsd** is aware that the power has returned. It should print:

Power has returned...

If you do not observe the above messages, please correct the situation before proceeding. The most likely cause of problems are:

- **apcupsd** doesn't recognize the power failure because the configuration directives are not correct. E.g. wrong cable.
- The file **/etc/apcupsd/apccontrol** doesn't exist or is not marked as executable.

At this point, we recommend that you do a simulated power down of your system. If you are adventuresome or have been through this before, skip to the next section in this manual and do the real power fail shutdown. If you continue with the simulated power down and if all goes well, **apcupsd** will go through all the motions without actually shutting down the system. Continue using the safe **apccontrol** that you installed. Edit the

configuration file `/etc/apcupsd/apcupsd.conf` and change the value of **TIMEOUT** from 0 to something like 30. Doing so will cause **apcupsd** to attempt to shutdown the system 30 seconds after it detects a power failure. Once this change has been made, you must stop and restart **apcupsd** for the new configuration value to take effect.

Once again, pull the power plug, and if all goes as expected, **apcupsd** should attempt to shutdown the system about 30 seconds after it detects the power failure. All the messages should be displayed by **wall** or by the **tail -f** command. The precise message is determined by what is printed in `/etc/apcupsd/apccontrol` for the **doshutdown** event. Though it varies from system to system, it will generally be something like:

Beginning Shutdown Sequence

When **apcupsd** this message prints, reconnect the power. **apcupsd** should detect that the power has been restored and attempt to cancel the shutdown.

IMPORTANT after this test, please replace the changed **apccontrol** and **apcupsd.conf** with the original files.

System Shutdown Test

This is an intermediate test that you can do, for all UPS models before doing the Full Power Down Test. First modify the `/etc/apcupsd/apccontrol` file so that in the **killpower** case, the line that re-executes **apcupsd** with the **--killpower** option is commented out. The original line probably looks something like:

```
${APCUPSD} --killpower
```

when it is commented out, it looks like:

```
# ${APCUPSD} --killpower
```

Now when you pull the power plug, and either the timer expires or the batteries are exhausted (see the next section for more details), the system should be fully shutdown.

After performing this test, please be sure to restore `/etc/apcupsd/apccontrol` to its previous state.

Full Power Down Test

To complete the testing, you should do a power fail shutdown of your system. This test is applicable to all UPS models. Please do a backup of your system or take other precautions before attempting this to avoid the possibility of lost data due to a problem (I have been through this at least 10 times and never once had problems, but we all know that someday something will go wrong).

Before proceeding, please ensure that your halt script or the equivalent has been properly updated by the install process to contain the logic to call **apcupsd --killpower** when it detects a power failure situation (the presence of a `/etc/powerfail` file). See the [install section](#) of this manual, or the README files for additional details about the halt modifications necessary.

When you are ready to do the test, either simply pull the plug and wait for the batteries to become exhausted, or set the **TIMEOUT** configuration directive to something like 60 so that the system will shutdown before the batteries are exhausted. We recommend doing the full shutdown without using **TIMEOUT** to correctly

simulate a real power failure, but the choice is yours (I did it once here, but now use `TIMEOUT 30`).

If all goes well, your system should be shutdown before the batteries are completely exhausted and the UPS should be powered off by **apcupsd**. Please be aware that if you do the full power down, you must ensure that your UPS is totally powered off. Otherwise, it may have been given the command to power off, but due to a long grace period it is still waiting. If you were to reboot your computer during the grace period, the UPS could then suddenly turn off the power (this happened to me). To avoid this problem, always wait for your UPS to power itself off, or power it off manually before restarting your computer. On my system, the UPS is configured as at the factory to have a 180 second grace period before shutting off the power. During this type of testing, 180 seconds **seems** like an eternity, so please take care to either wait or manually power off your UPS. To determine what grace period is programmed into your UPS EEPROM, run **apcaccess eprom** and look at the "Shutdown grace delay".

Shutdown Sequence

If you experienced so problems with the above testing procedures, or if you are porting **apcupsd** to another system, or you are simply curious, you may want to know exactly what is going on during the shutdown process. If so, please see the [Shutdown Chapter](#) of this manual.

Testing the CGI Programs

Please see the [CGI Programs](#) chapter of this manual for how to test the Network Information Server and the CGI programs.

Recalibrating the UPS Runtime

This section does not apply to simple signaling or dumb UPSes such as the BackUPS.

Smart UPSes internally compute the remaining runtime, and **apcupsd** uses the value supplied by the UPS. As the batteries age (after say two or three years), the runtime computation may no longer be accurate since the batteries no longer hold the same charge. As a consequence, in the event of a power failure, the UPS and thus **apcupsd** can report a runtime of 5 minutes remaining when in fact only one minute remains. This can lead to a shutdown before you might expect it, because regardless of the runtime remaining that is reported, the UPS will always correctly detect low batteries and report it, thus causing **apcupsd** to correctly shutdown your computer.

If you wish have the UPS recalibrate the remaining runtime calculations, you can do so manually as the current version of **apcupsd** does not support this feature. To do so,

- Shutdown **apcupsd**
- contact your UPS directly using some terminal program such as minicom, tip, or cu with the settings 2400 8N1 (2400 baud, 8 bits, no parity, 1 stop bit). Be extremely careful what you send to your UPS as certain characters may cause it to power down or may even cause damage to the UPS. Try sending an upper case Y to the UPS (without a return at the end). It should respond with SM. If this is not the case, read the chapter on testing. If you fat finger the Y and enter y instead, no cause for alarm, you will simply get the APC copyright notice.
- when you are sure you are properly connected send an upper case D (no cr). This will put the UPS into calibration mode, and it will drain the battery down to 25% capacity (35% for a Matrix) at which point it will go back on the mains. In doing so, it will recompute the runtime calibration.

- If you wish to abort the calibration, enter a second D command.
- When you are done, restart apcupsd.

In principle, you should be able to do this with the computer powered by the UPS, but if you wish to be completely safe, you should plug your computer into the wall prior to performing the runtime calibration. In that case, you will need to artificially load the UPS with light bulbs or other means. You should supply a load of about 30 to 35% but not more than 50%. You can determine the load by looking at the output of the **apcaccess status** command while **apcupsd** is running.

You should not run the recalibration command more than once or twice per year as discharging these kinds of batteries tends to shorten their life span.



Trouble Shooting Apcupsd

Testing

The first step in trouble shooting **apcupsd** is to read the [Testing Apcupsd](#) section of this manual.

Network Problems with Master/Slave Configurations

When working with a master/slave configuration (one UPS powering more than one computer), the master and slave communicate via the network. In many configurations, **apcupsd** is started before the network is initialized. In this case, it is possible that the master will be unable to contact the slave. On **apcupsd** versions prior to 3.8.0, this could cause **apcupsd** to error off. The solution to this problem is to either force **apcupsd** to be started after the network and the DNS (fiddle the symbolic links in `/etc/rc.d`), or put the names of the slave machines in your `/etc/hosts` file, or even more preferable, use IP addresses rather than machine names. On some configurations, you may need to use fully qualified names (`host.domain.xxx`) rather than simple host names.

Error Messages from a Master Configuration

In a master/slave configuration, you can get the following error messages from a master. The error message is followed by a possible explanation:

Cannot resolve slave name XXX

To contact the slave, the slave name given in the configuration file must be resolved to an IP address. In this case, **apcupsd** could not get the IP address. Either the slave name is incorrect, your DNS may not be working, or you have started **apcupsd** during the boot process before the network is operational.

Got slave shutdown from SSS

This message should not be printed as it is not yet used.

Cannot write to slave SSS

This message occurs when the master attempts to send a message to the slave SSS and gets an error. It indicates that either the slave machine is not responding (**apcupsd** died, the system crashed, ...) or that the network is down.

Cannot read magic from slave SSS

This message indicates that the master attempted to read the code key from the slave SSS and it did not match the value expected. A common cause of this problem is that the master and slave versions of **apcupsd** are not the same. Please be sure you are running the same version of **apcupsd** on all your master and slave machines.

Connect to slave SSS failed

This message is logged when the master attempts to connect to slave SSS and no connection is accepted. The most common cause of this problem is that the slave copy of **apcupsd** is not yet ready to accept connections or is not running. Generally, **apcupsd** will retry the connection a bit later. If the problem is persistent, it can

indicate a network problem or the slave name on the SLAVE directive of the master's configuration file is incorrect.

Cannot open stream socket

This indicates a fundamental networking problem on your system — either a lack of sufficient resources or you have not configured TCP/IP operations.

Error Messages from a Slave Configuration

In a master/slave configuration, you can get the following error messages from a slave. The error message is followed by a possible explanation:

Can't resolve master name MMM

This message is logged when the slave attempts to resolve the name given on the MASTER configuration directive to an IP address. It probably means that the master name MMM is not defined, your DNS is not properly working, or you have started **apcupsd** in the boot process before the network is initialized. Check the name MMM, or use an explicit IP address on the MASTER configuration directive in the slave's configuration file.

Cannot bind local address, probably already in use

This means that the slave has attempted to bind the port number so that it can listen for messages from the master. This can occur if already have a copy of **apcupsd** running, or you have previously run **apcupsd** in the past 5 or 10 minutes, because occasionally the operating system will not shutdown a port correctly for 5 to 10 minutes after a program exits. In this case, you can either wait a few minutes for the problem to go away, or use a different port in both your master and slave configuration files.

Socket accept error

The slave got an error waiting on the accept() system call. This is probably due to a fundamental networking problem.

Unauthorized attempt from master MMM

The master named MMM (probably an IP address) contacted the slave but MMM is not the master that was listed on the MASTER configuration directive in `/etc/apcupsd.conf`, and consequently, it is not authorized to communicate with the slave. Please check that your MASTER and SLAVE names in your slave and master configuration files respectively are correct.

Read failure from socket

The slave got an error reading the socket open to the master. This indicates a fundamental networking problem.

Bad APC magic from master: MMM

The slave received a code key from the master that does not correspond to the one expected by the slave. The most common cause of this problem is that you are running a different version of **apcupsd** on the master and

the slave. Please ensure that you are running the same version of **apcupsd** on all your master and slaves.

Bad user magic from master: MMM

This message indicates that the master and slave have previously communicated, but that the code key transmitted with the most recent message from the master does not correspond to what the slave expects. This problem is probably due to a network error or some other user or machine contacting the slave on the network port.

Master/Slave Connection Not Working

Master/slave problems are usually related to one of the following items:

1. Improper `apcupsd.conf` files. A good starting point are the master/slave example files in the examples subdirectory of the source.
2. Master or slave IP address or name incorrect. Try ping'ing each machine from the other using the names or addresses that you have put in the respective `apcupsd.conf` files.
3. Make sure no other program is using socket number 6666 or change the `NETPORT` directive in both `apcupsd.conf` files.
4. Make sure you are using the same version of `apcupsd` on both the master and slave machines.

CGI Programs Do Not Work

Try checking the following:

1. Did you successfully compile and link the cgi programs without errors? If not sure, cd to the cgi directory, do a "make clean" followed by a "make"
2. Did you move or copy all the .cgi programs in the cgi directory to your Web server cgi-bin directory on the SAME machine?
3. Did you verify that the cgi programs located in the cgi-bin all have execute permission?
4. Have you tried any other cgi programs and proven that they work?
5. Have you verified that the Network Information Server process of **apcupsd** is running as described in this manual?
6. Have you verified that your **apcupsd.conf** file is properly configured for the Network Information Server and that the port is defined as 7000? I.e. "NETSERVER on" and "SERVERPORT 7000"
7. If one or more machines does not show up in the multimon output, it is most likely due to a configuration error in the **hosts.conf** file in your **/etc/apcupsd** directory.

Battery Problems

Please see the [Battery Chapter](#) of this document for more details.

Cable or Connection Problems

Frequently during the initial installation, users don't know what cable they have or have problems connecting to the serial port. If this is your case, one means of diagnosing the problem can be to use the **apctest** program. To do so, you must first build it with:

```
make apctest
```

Then, you simply execute it with:

```
./apctest
```

and follow the instructions. It will place the output from the session in the file **apctest.output**. If you are not able to resolve your problem, sometimes we can help if you email us this output file along with your **apcupsd.conf** file. Please see the [Testing Chapter](#) of this document for additional details on how to build and use **apctest**.

Bizarre Intermittent Behavior

In one case, a user reported that he received random incorrect values from the UPS in the status output. It turned out that **gpm**, the mouse control program for command windows, was using the serial port without using the standard Unix locking mechanism. As a consequence, both **apcupsd** and **gpm** were reading the serial port. Please ensure that if you are running **gpm** that it is not configured with a serial port mouse on the same serial port.



Shutdown Sequence

If you experienced so problems with the testing procedures, or if you are porting **apcupsd** to another system, or you are simply curious, you may want to know exactly what is going on during the shutdown process.

The shutdown sequence is as follows:

- Apcupsd detects that there is a power problem and it calls **/etc/apcupsd/apccontrol powerout**, which normally sends a message to all users informing them of a potential problem.
- After approximately 5 seconds in the power problem mode, Apcupsd calls **/etc/apcupsd/apccontrol onbattery**, which normally sends a message to all users informing them that the UPS is on batteries.
- When one of the conditions listed below occurs, **apcupsd** issues a shutdown command by calling **/etc/apcupsd/apccontrol doshutdown**, which should perform a shutdown of your system using the system **shutdown** command. You can modify the behavior by editing the **/etc/apcupsd/apccontrol** script, but doing so will make it more complicated to upgrade to the next **apcupsd** version.

The conditions that trigger the shutdown can be: running time on batteries have expired (TIMEOUT), the battery runtime remaining is below the configured value (BATTERYLEVEL), the estimated remaining runtime is below the configured value (MINUTES), or the UPS signals that the batteries are exhausted.

A shutdown could also be initiated if **apcupsd** detects that the batteries are no longer functioning correctly. This case, though very unusual, can happen at any time even if there is proper mains voltage, and **/etc/apcupsd/apccontrol emergency** is called.

Just before initiating any shutdown through the apccontrol script, **apcupsd** will create the file **/etc/apcupsd/powerfail**. This file will be used later in the shutdown sequence to recall **apcupsd** after syncing of the disks to initiate a power off of the UPS.

If the **/etc/nologin** file has not already been created, it will normally be created during the shutdown sequence to prevent additional users from logging in (see the NOLOGIN configuration directive).

Even though **apcupsd** has requested the system to perform a shutdown, it continues running. If it is a master with slaves, it will inform the slaves to do a shutdown. They perform their shutdown by calling **/etc/apcupsd/apccontrol remoteshutdown**.

- When the system signals **apcupsd** to do exit, it does so. This is part of the normal system shutdown (at least on Unix and Linux systems) and the exact time that **apcupsd** receives the termination signal depends on how the shutdown links (usually in **/etc/rc.d**) are set.

Note that on Windows NT systems, **apcupsd** apparently continues to run as a Service even though the machine is "shutdown".

- During the shutdown of the system after **apcupsd** has been forced to exit, one of the last things done by the system shutdown is to call the halt script, which is usually in **/etc/rc.d/halt** or **/etc/rc.d/init.d/halt**, or possibly in **/sbin/init.d/rc.0** depending on your system. If apcupsd was properly installed, this standard halt script was modified to include a bit of new logic just before the final halt of the system. It first tests if the file **/etc/apcupsd/powerfail** exists, and if it does, it executes

/etc/apcupsd/apccontrol killpower. It is this last step that will cause **apcupsd** to be re-executed with the **--killpower** option on the command line. This option tells **apcupsd** to inform the UPS to kill the power.

This final step is important if you want to ensure that your system will automatically reboot when the power comes back on. The actual code used on the RedHat version is:

```
# See if this is a powerfail situation.
if [ -f /etc/apcupsd/powerfail ]; then
    echo
    echo "APCUPSD will now power off the UPS"
    echo
    /etc/apcupsd/apccontrol killpower
    echo
    echo "Please ensure that the UPS has powered off before rebooting"
    echo "Otherwise, the UPS may cut the power during the reboot!!!"
    echo
fi
```

```
# ***apcupsd***
# ***apcupsd***
# ***apcupsd***
# ***apcupsd***
# ***apcupsd***
# ***apcupsd***
# ***apcupsd***
# ***apcupsd***
# ***apcupsd***
# ***apcupsd***
# ***apcupsd***
# ***apcupsd***
```

The above code must be inserted as late as possible in the halt script. On many systems, such as RedHat, all the disk drives were unmounted, then remounted read-only, thus permitting access to the /etc files and the **apcupsd** executable. If your system does not explicitly remount the disks, you must remount them in read-only mode in the code that you add. Examples of code fragments that do this can be found in the distributions/suse subdirectory of the source.

If you are not able to insert the above code in your halt script because there is no halt script, or because your halt script calls the **init** program as some Unix systems do, you can either just forget about powering off the UPS, which means that your machine will not automatically reboot after a power failure, or there is yet another alternative, though not at all as satisfying as inserting code in the halt script.

Only if you cannot insert the appropriate code in the halt script, when you start **apcupsd**, normally from the /etc/rc.d/init.d/apcupsd script, use the **--kill-on-powerfail** option. This will cause **apcupsd** to program the UPS to shutoff the power just before it (**apcupsd**) does the system shutdown. Please note that this is not the most ideal solution. Read on to understand why.

A very important consideration is that you must set the EEPROM in your UPS so that it waits a sufficient time for the system to halt before it shuts off the UPS power. The current value as well as the permitted values for your UPS can be determined by executing:

```
apcaccess eeprom
```

The output should look something like the following:

```
apcaccess eeprom
```

```
Valid EPROM values for the SMART-UPS 1000
```

Description	Config Directive	Current Value	Permitted Values
Upper transfer voltage	HITRANSFER	253	253 264 271 280
Lower transfer voltage	LOTRANSFER	196	196 188 208 204
Return threshold	RETURNCHARGE	0	00 15 50 90

Output voltage on batts	OUTPUTVOLTS	230	230	240	220	225
Sensitivity	SENSITIVITY	H	H	M	L	L
Low battery warning	LOWBATT	2	02	05	07	10
Shutdown grace delay	SLEEP	20	020	180	300	600
Alarm delay	BEEPSTATE	0	0	T	L	N
Wakeup delay	WAKEUP	0	000	060	180	300
Self test interval	SELFTTEST	336	336	168	ON	OFF

The line of interest for you is the **Shutdown grace delay**, which can be changed using the [SLEEP](#) directive in your **apcupsd.conf** file. The default value is 20 seconds, but generally, you can set it to 180, 300, or 600 seconds depending on your UPS. See the [EEPROM programming section](#) of this manual for further details on how to change this EPROM value. --kill-on-powerfail option, you run the risk of having the computer power cut before the system has shutdown. Even if the grace period is rather long, if something goes wrong in the shutdown, well, it is up to you to decide.

Automatic Reboot of your Computer after a Power Shutdown

If **apcupsd** has successfully shutdown your computer and powered off the UPS during a power outage, you can control whether or not your computer is automatically rebooted when the power returns.

The UPS contains two internal EPROM values that determine when it will restore power to your computer after a full power shutdown. They are the [RETURNCHARGE](#) percentage and the [WAKEUP](#) delay. Briefly, the **RETURNCHARGE** specifies what percentage charge the battery must have before the power is restored. Higher values are recommended in regions where the power goes up and down frequently. The **WAKEUP** delay is a simple time delay. Most sites will have both of these at zero, or perhaps the **RETURNCHARGE** set to 15. Please follow the links to the **Configuration** section of this manual for more information. See the [EEPROM programming section](#) of this manual for further details on how to change these EPROM values.

The final consideration for a automatic reboot after a full power down is to ensure that your computer will automatically reboot when the power is restored. This is not the normal behavior of most computers as shipped from the factory. Normally after the power is cut and restored, you must explicitly press a button for the power to actually be turned on. You can test your computer by powering it down; shutting off the power (pull the plug); then plugging the cord back in. If your computer immediately starts up, good. There is nothing more to do. If your computer does not start up, manually turn on the power (by pressing the power on button) and enter your computer's SETUP program (often by pressing DEL during the power up sequence; sometimes by pressing F10). You must then find and change the appropriate configuration parameter to permit instant power on. Normally, this is located under the **BOOT** menu item, and will be called something such as **Restore on AC/Power Loss** or **Full-On**. The exact words will vary according to the ROM BIOS provider. Generally you will have three options: **Last State**, **Power On**, and **Power Off**. Although **Last State** should normally work, I set my computers to **Power On**. This means that whenever the power is applied they are on. The only way to shut them off is to pull the plug or to have a special program that powers them off (**/sbin/poweroff** on Linux systems).

Shutdown Problems

Obviously if your halt script is not properly modified, **apcupsd** will not be able to shut off the power to the UPS, and if the power returns before the batteries are exhausted your system will not automatically reboot. In any case, your machine should have been cleanly shutdown.

Master/Slave Shutdown

In master/slave configurations, however, the master cannot be 100 percent sure that the slaves have all shutdown before it performs the power off. As a consequence, it is possible that the master will shut off the power before the slave has finished shutdown. If this is the case, the best procedure is to put an appropriate sleep command in the **/etc/apcupsd/apccontrol** file on the master. For example to give the slaves 30 additional seconds to shutdown, one would add:

```
sleep 30
```

just after the line that reads

```
doshutdown)
```

in the apccontrol file (approximately line 79 – depending on your system version).

Also, on a slave machine, you do not want to use the modified halt script since it will recall **apcupsd**, which will detect that it is a slave (i.e. no connection to the UPS) and will complain that it cannot do the killpower. This situation is not harmful just annoying and possibly confusing.

One possible problem during shutdown can be caused by remnants of old versions. Please be sure to delete or rename all prior versions (**/usr/local/sbin/apcupsd** or **/sbin/powersec**).

Startup

Normally, **apcupsd** is automatically started when your system is rebooted. This normally occurs because the startup script **apcupsd** is linked into the appropriate places in **/etc/rc.d**. On most Linux systems, there is a program called **chkconfig** that will automatically link the startup script. This program is invoked by the **make install** scripts, or it is explicitly done for those systems that do not have **chkconfig**. If this is not the case, you can either link it in appropriately yourself or explicitly call it from your **rc.local** file. The appropriate manual way to startup **apcupsd** is by executing:

```
<path>/apcupsd start
```

where **<path>** is normally **/etc/rc.d** or **/etc/rc.d/init.d** depending on your system (isn't Unix wonderful? :-)). Using this script is important so that any files remaining around after a power failure are removed. Likewise, shutting down **apcupsd** should be done with the same script:

```
<path>/apcupsd stop
```

Windows Considerations

Please see the end of [Apcupsd Under Windows](#) chapter of this manual for conderations pertaining to shutdown and killpower on Windows.



apcaccess

Apcaccess is a program (normally found in `/sbin/apcaccess`) that permits you to print out the complete status of your UPS. Although there are a number of command line arguments (**eprom**, **reconfig**, **status**, **slave**, **shutdown**), all except **eprom** and **status** are under development and hence do not work.

If you have built **apcupsd** with pthreads enabled, **apcaccess** will use the Network Information Server to obtain the necessary information for the **status** and **eeeprom** commands. This is because in the pthreaded version, there is no IPC shared memory. In this case (pthreads enabled), you can specify a second optional argument to **apcaccess** in the form of `host:port`, where the `:port` is optional. The default is **localhost:7000**. In the current version (3.8.2) specifying `host:port` works only if the host specified is running the NIS server and an **apcupsd** identical (version, architecture, ...) to the localhost (i.e. this code is apcupsd version dependent as well as machine dependent).

apcaccess status

The **status** command line option of **apcaccess** will produce a full printout of all the **STATUS** variables used by **apcupsd**. This can be very helpful for checking the condition of your UPS and to know whether or not **apcupsd** is properly connected to it. For a complete description of the variables and their meanings, please read the [Apcupsd STATUS](#) document.

Please note that if you invoke **apcaccess** within the first 30 seconds of launching **apcupsd**, you will likely get an error message such as:

```
APCACCESS FATAL ERROR in apcipc.c at line 325
attach_shmarea: shared memory version mismatch
```

This is because **apcupsd** is still in the process of initializing the shared memory segment used to communicate between the two processes. There is also a small window of time after which the memory segment is properly initialized but before the UPS has been completely polled. If you invoke **apcaccess** during this period, you will get the STATUS output, but with many of the values zero. The solution is to wait at least 30 seconds after starting **apcupsd** before launching **apcaccess**.

To invoke apcaccess, enter:

```
apcaccess status
```

and for a SmartUPS 1000 **apcaccess** prints the following output:

```
DATE       : Fri Dec 03 12:34:26 CET 1999
HOSTNAME   : matou
RELEASE    : 3.7.0-beta-1
CABLE      : Custom Cable Smart
MODEL      : SMART-UPS 1000
UPSMODE    : Stand Alone
UPSNAME    : UPS_IDEN
LINEV      : 232.7 Volts
MAXLINEV   : 236.6 Volts
MINLINEV   : 231.4 Volts
LINEFREQ   : 50.0 Hz
```

```

OUTPUTV : 232.7 Volts
LOADPCT : 11.4 Percent Load Capacity
BATTV : 27.7 Volts
BCHARGE : 100.0 Percent
MBATTCHG : 5 Percent
TIMELEFT : 112.0 Minutes
MINTIMEL : 3 Minutes
SENSE : Low
DWAKE : 060 Seconds
DSHUTD : 180 Seconds
LOTRANS : 204.0 Volts
HITRANS : 253.0 Volts
RETPCT : 050.0 Percent
STATFLAG : 0x08 Status Flag
STATUS : ONLINE
ITEMP : 29.2 C Internal
ALARMDEL : Low Battery
LASTXFER : U command or Self Test
SELFTST : NO
STESTI : 336
DLOWBATT : 02 Minutes
DIPSW : 0x00 Dip Switch
REG1 : 0x00 Register 1
REG2 : 0x00 Register 2
REG3 : 0x00 Register 3
MANDATE : 01/05/99
SERIALNO : GS9902009459
BATTDATE : 01/05/99
NOMOUTV : 230.0
NOMBATTV : 24.0
HUMIDITY : N/A
AMBTEMP : N/A
EXTBATT : 0
BADBATT : N/A
FIRMWARE : 60.11.I
APCMODEL : IWI
END APC : Fri Dec 03 12:34:33 CET 1999

```

apcaccess eeprom

The **eeprom** command line option for **apcaccess** allows you to examine the current values of your UPS' EPROM as well as to know the permitted values that can be set in the EPROM. For information about changing these values, see the section on [Apcupsd EEPROM Configuration](#).

A typical output from **apcaccess eeprom** is:

Valid EPROM values for the SMART-UPS 1000

Description	Config Directive	Current Value	Permitted Values
=====			
Upper transfer voltage	HITRANSFER	253	253 264 271 280
Lower transfer voltage	LOTRANSFER	208	196 188 208 204
Return threshold	RETURNCHARGE	15	00 15 50 90
Output voltage on batts	OUTPUTVOLTS	230	230 240 220 225
Sensitivity	SENSITIVITY	H	H M L L
Low battery warning	LOWBATT	2	02 05 07 10
Shutdown grace delay	SLEEP	180	020 180 300 600

APC UPS management under Linux

Alarm delay	BEEPSTATE	T	0	T	L	N
Wakeup delay	WAKEUP	60	000	060	180	300
Self test interval	SELFTEST	336	336	168	ON	OFF



apctest

apctest is a program that allows you to directly talk to your UPS and run certain tests. It will function either for Simple Signaling UPSes (dumb UPSes) or for Smart UPSes.

Running apctest for a Simple Signaling UPS

Shutdown `apcupsd` if it is running.

Make sure your `/etc/apcupsd/apcupsd.conf` file has **UPSTYPE backups** and **UPSCABLE simple**

Normally **apctest** will have been built and installed by default, otherwise, you can explicitly build it on Unix with:

```
cd <apcupsd-source-directory>
make apctest
./apctest
```

on Win32 systems, use:

```
make apctestwin32
./apctest
```

It will present you with the following output

```
2001-02-07 04:08:26 apctest 3.8.5 (3 January 2002) redhat
Checking configuration ...
sharenet.type = DISABLE
cable.type = CUSTOM_SIMPLE
mode.type = BK
Setting up serial port ...
Creating serial port lock file ...
Doing prep_serial() ...
Hello, this is the apcupsd Cable Test program.
This part of apctest is for testing dumb UPSes (ones that uses signaling rather than commands.
Most tests enter a loop polling every second for 10 seconds.
```

Then it will present you with the following list of choices:

- 1) Test 1 - normal mode
- 2) Test 2 - no cable
- 3) Test 3 - no power
- 4) Test 4 - low battery (requires test 3 first)
- 5) Test 5 - battery exhausted
- 6) Test 6 - kill UPS power
- 7) Test 7 - run tests 1 through 5
- 8) Guess which is the appropriate cable
- 9) quit

Select test number:

Run tests 1, 2, and 3. Note, none of the currently supported cables will indicate a change for test 2. You can then run test 8 to see what cable it thinks you should be using. Finally run test 4.

`apctest` can also be run for Smart UPSes.

The print out of your testing will be written to the file **apctest.output**. If you are unable to solve your problem, you can try posting that file to the development mailing list, and perhaps we can help you. In this case, please also include information on your operating system, which version of **apcupsd** you are using, your UPS model, and also your **apcupsd.conf** file.

Expected apctest Signals for a Dumb UPS

If you have configured your UPS as:

```
UPSTYPE backups
UPSCABLE APC_940_0119A
      or APC_940_0127A
      or APC_940_0128A
      or APC_940_0020B
      or APC_940_0020C
```

here are typical signals you would expect to see in the output from the various tests of apctest:

```
Test 1 normal:           RTS for cables (0119A 0127A 0128A)
Test 2 no serial cable:  not important
Test 3 no AC power:      CTS for all cables
Test 4 batteries exhausted: CTS and CD for all cables
```

Note: **RTS** if set in Test 1 will probably also be set in all the other tests. This is not important, what counts is the appearance of **CTS** when the power fails and additionally **CD** when the batteries are low.

Expected apctest Signals for a BackUPS Pro

If you have configured your UPS as:

```
UPSTYPE backupsp
UPSCABLE APC_940_0095A
      or APC_940_0095C
```

here are the typical signals you would expect to see in the output from the various tests of apctest:

```
Test 1 normal:           RTS not set
Test 2 no serial cable:  not important
Test 3 no AC power:      RNG
Test 4 batteries exhausted: RNG and CD
```

Note: **RTS** should never be set in any of the tests as it is the killpower signal. What is important is the appearance of **RNG** when the power fails and additionally **CD** when the batteries are low.

Running apctest for a Smart UPS

Shutdown apcupsd if it is running.

Make sure your **/etc/apcupsd/apcupsd.conf** file has **UPSTYPE backups** and **UPSCABLE simple**

Normally **apctest** will have been built and installed by default, otherwise, you can explicitly build it on Unix with:

```
cd <apcupsd-source-directory>
```

```
make apctest
./apctest
```

on Win32 systems, use:
make apctestwin32
./apctest

It will present you with the following output

```
2002-01-03 21:04:57 apctest 3.8.5 (3 January 2002) redhat
Checking configuration ...
sharenet.type = DISABLE
cable.type = CUSTOM_SMART

You are using a SMART cable type, so I'm entering SMART test mode
mode.type = SMART
Setting up serial port ...
Creating serial port lock file ...
Hello, this is the apcupsd Cable Test program.
This part of apctest is for testing Smart UPSes.
Please select the function you want to perform.

1) Query the UPS for all known values
2) Perform a Battery Runtime Calibration
3) Abort Battery Calibration
4) Monitor Battery Calibration progress
5) Quit

Select function number:
```

Item 1 will probe the UPS for all values known to **apcupsd** and present them in rather raw format. This output can be useful for providing technical support if you are having problems with your UPS.

Item 2 will perform a Battery Runtime Calibration. This test will only be performed if your battery is 100% charged. Running the test will cause the batteries to be discharged to approximately 30% of capacity. The exact number depends on the UPS model. In any case, **apctest** will abort the test if it detects that the battery charge is 20% or less.

The advantage of doing this test is that the UPS will be able to recalibrate the remaining runtime counter that it maintains in its firmware. As your batteries age, they tend to hold less of a charge, so the runtime calibration may not be accurate after several years.

We recommend that perform a Battery Calibration about once a year. You should not perform this calibration too often since discharging the batteries tends to shorten their lifespan.

Item 3 can be used to abort a Battery Calibration in progress, if you some how became disconnected.

Item 4 can be used to restart the monitoring of a Battery Calibration if you should some how become disconnected during the test.

Item 5 will terminate **apctest**.



The Apcupsd Network Information Server

Apcupsd maintains STATUS and EVENTS data concerning the UPS and its operation. This information can be obtained over the network using either **apcnetd** or **apcupsd**'s internal network information server, which is essentially the same code as **apcnisd** but compiled into apcupsd. Clients on the network make a connection to the information server and send requests for [STATUS](#), or [EVENTS](#) data, which the server then transmits to them.

The information served to the network by this interface should not be confused with sharing a UPS between two or more computers. That code is described in the [configuration section](#) of this documentation.

There are three different ways to run the information server daemon depending on your requirements and preferences. It can be run as 1. a standalone program, 2. a standalone program invoked by the inetd daemon, or 3. as a child process of **apcupsd** (default configuration).

Running the Network Information Server Directly within Apcupsd

This is probably the simplest way to run the network information server. To do so, you simply modify the **NETSTATUS** directive in **/etc/apcupsd/apcupsd.conf** to be **on**, and then [stop](#) and [restart](#) **apcupsd**. It will automatically spawn an additional child process named **apcnis** to handle network clients. In the case where pthreads are enabled, a new thread will be created rather than a child process to handle the network information requests. Note, the above modification should not be necessary if you use the default **apcupsd.conf**, since it is already turned on.

Although this method is simple, it affords no protection from the outside world accessing your network server unless, like me, you are behind a firewall. In addition, if there is a bug in the network server code, or if a malicious user sends bad data, it may be possible for **apcnis** to die, in which case, though it is not supposed to, **apcupsd** may also exit, thus leaving your machine without shutdown protection. That being said, most of us prefer to run the server this way.

With **apcupsd** version 3.8.2, you may enable the TCP Libwrap subroutines to add additional security. In this case, access to the network server will be controlled by the statements you put in **/etc/hosts.allow**.

Running apcnisd from INETD

This is probably the most secure and most desirable way of running the network information server. Unfortunately, it is a bit more complicated to setup. However, once running, the server remains unexecuted until a connection is attempted, at which point, inetd will invoke **apcnisd**. Once **apcnisd** has responded to the client's requests, it will exit. None of the disadvantages of running it standalone apply since **apcnisd** runs only when a client is requesting data.

An additional advantage of this method of running the network information server is that you can call it with a TCP wrapper and thus use access control lists (ACL) such as **hosts.allow**. See the man pages for **hosts.allow** for more details.

To configure **apcnisd** to run from INETD, you must first put an entry in **/etc/services** as follows:

```
apcnisd          7000/tcp
```

This defines the port number (7000) and the service (TCP) that **apcnisd** will be using. This statement can go anywhere in the services file. Normally, one adds local changes such as these to the end of the file.

Next, you must modify **/etc/inetd.conf** to have the following line:

```
apcnisd stream tcp nowait root /usr/sbin/tcpd /sbin/apcnisd -i
```

If you do not want to run the TCP wrapper, then the line should be entered as follows (not tested):

```
apcnisd stream tcp nowait root /sbin/apcnisd -i
```

Please check that the file locations are correct for your system. Also, note that the **-i** option is necessary so that **apcnisd** knows that it was called by INETD. Before restarting INETD, first ensure that the **NETSTATUS** directive in **/etc/apcupsd/apcupsd.conf** is set to **off**. This is necessary to prevent **apcupsd** from starting a child process that acts as a server. If you change **NETSTATUS**, you must [stop](#) and [restart](#) **apcupsd** for the configuration change to be effective.

Finally, you must restart INETD for it to listen on port 7000. On a RedHat system, you can do so by:

```
/etc/rc.d/init.d/inet reload
```

At this point, when a client attempts to make a connection on port 7000, INETD will automatically invoke **apcnisd**.

Running apcnisd Standalone

This is probably the least desirable of the three ways to run an **apcupsd** network information server because if **apcupsd** is stopped, you must also stop **apcnisd** before you can restart **apcupsd**. This is because **apcnisd**, when run standalone, holds the shared memory buffer by which **apcnisd** and **apcupsd** communicate. This prevents a new execution of **apcupsd** from creating it.

To execute **apcnisd** in standalone mode, first ensure that the **NETSTATUS** directive in **/etc/apcupsd/apcupsd.conf** is set to **off**. This is necessary to prevent **apcupsd** from starting a child process that acts as a server. Restart **apcupsd** normally, then:

```
/sbin/apcnisd
```

The advantage of running the network information server standalone is that if for some reason, a client causes the network server to crash, it will not affect the operation of **apcupsd**.





Apcupsd Bugs

Apcupsd Version 3.9.8 Bugs

Since the software has major modifications, there are probably quite a few bugs, many of which are simply unfinished implementations. For example, the EEPROM programming feature is disabled in this version while we are deciding how to do it the **right** way.

The 3.8.5 Slackware bug reported below for version 3.8.5 does not apply to version 3.9.8.

The killpower Feature Does Not Work On USB UPSes

We have not yet been able to make a USB UPS shut off the power. This applies both to the BackUPS CS models as well as the SmartUPS models. Though this means the implementation is incomplete, it should not be a major issue if after performing a system halt, your computer does not power off (this is the normal behavior). In that case, the computer will continue to drain the UPS batteries and generally within 2 minutes the UPS will shutoff. If this happens, your computer will automatically reboot when the mains power returns. Unfortunately, leaves a 2 minute (or longer) window where the mains power can return and your computer will be left in a halted state.

Battery Voltages Are Not Correct

On all USB UPSes, the battery voltage is incorrectly scaled in STATUS output. In addition, on Back-UPS CSeS, the BATTV value is not reported in STATUS output. Though annoying, this causes no harm as the battery voltages are used for display only.

Apcupsd Version 3.8.5 Bugs

All previous bugs reported against **apcupsd** have been fixed in this version.

Slackware -- After a Power Failure, It Reboots

This bug concerns the Slackware platform only. When power is lost and then restored, **apcupsd** will cause the system to reboot. This is due to an omission (error) in the file **distributions/slackware/apccontrol.sh.in**. The patch is as follows:

```
--- apccontrol.sh.in.orig      Wed Mar 13 17:51:05 2002
+++ apccontrol.sh.in          Wed Mar 13 17:53:40 2002
@@ -82,9 +82,11 @@
     ;;
     mainsback)
         printf "Power has returned..." | wall
-        printf "Attempting to cancel shutdown." | wall
-        ${SHUTDOWN} -c
```



```
-      ${SHUTDOWN} -r now "apcupsd initiated reboot"
+      if [ -f @PWRFAILDIR@/powerfail ] ; then
+          printf "Attempting to cancel shutdown." | wall
+          ${SHUTDOWN} -c
+          ${SHUTDOWN} -r now "apcupsd initiated reboot"
+      fi
+
+      ;;
+      annoyme)
+          printf "Power problems please logoff." | wall
```

After applying this patch, you must either do:

```
make Makefiles
make install
```

or simply rerun the whole build process starting from **./configure**

Apcupsd Version 3.8.4 Bugs

Version 3.8.4 is a bug fix version to 3.8.3 that corrects improper placement of the subsystem lock file that crept in between 3.8.2 and 3.8.3. It also corrects an oversight in the `multimoncsc.c` code that didn't include the temperature and humidity columns in the **Normal** class.

Apcupsd Version 3.8.3 Bugs

All known bugs in version 3.8.2 have been fixed with the exception of the following two:

Networking may not work in a mixed vendor setup

Bug: In a mixed vendor setup (RedHat, Debian, ...), the default networking (master/slave or Network Information Server) port assignments are different. Reason: Due to port conflicts on some machines, we set the default port numbers on each distribution to the values recommended by our experts. This solves a lot of problems, but results in incompatibilities if you use the **apcupsd** defaults. Fix: If you run a mixed vendor setup, either add a port specification **:<port>** to the machine name in **hosts.conf** or use the **--with-nis-port=nn** and **--with-net-port=nn** options on your **./configure** command and ensure that all your machines use the same port numbers. After the **./configure**, you will need to redo your **make** and **make install**. The default values for most systems are:

```
./configure --with-nis-port=7000 --with-net-port=6666
```

Unusual Case of Apcupsd Hanging during Boot

On some systems (kernel version 2.4.5-acxx), **apcupsd** may hang during the boot process. This appears to be a probably of invoking **apcupsd** before the network is initialized. The simplest solution is to ensure that **apcupsd** is started after the network functions, or to put an explicit background (ampersand) request in the **apcupsd** startup script.

If this occurs and your **apcupsd** is a master, the problem can be because you master/slave networking port is used by another program on the slave machine. The master/slave networking code has been modified to timeout in 10 seconds in this case.

Apcupsd Version 3.8.2 Bugs

All version 3.8.0 and 3.8.1 bugs have been corrected in version 3.8.2.

The Lock Directory in the Solaris is Incorrect

Bug: The apcupsd script is using /var/lock as the directory for lock files even though the configure script figures out that it is supposed to be /var/spool/locks (/var/lock does not exist).

Fix: Apply the following patch then re-run your ./configure:

```
--- distributions/sun/old.apcupsd.in      Sat Jul  7 10:20:30 2001
+++ distributions/sun/apcupsd.in          Sat Jul  7 10:18:59 2001
@@ -17,7 +17,7 @@
     rm -f @PWRFAILDIR@/powerfail
     echo "Starting apcupsd power management ... \c"
     @sbindir@/apcupsd || return="  Failed."
-    touch /var/lock/apcupsd
+    touch @LOCKDIR@/apcupsd
     echo "$return"
;;
stop)
@@ -29,7 +29,7 @@
     else
         return="  Failed."
     fi
-    rm -f /var/lock/apcupsd
+    rm -f @LOCKDIR@/apcupsd
     echo "$return"
;;
restart)
```

Solaris doesn't Shutdown

Bug: Solaris detects power failures and seems to work fine, but the machine is not shutdown.

Reason: You probably executed the ./configure for **apcupsd** with **/usr/ucb** on your path before **/usr/sbin**. Thus **apcupsd** is using the Berkeley shutdown program but with SysV arguments.

Fix: remove **/usr/ucb** from your path and rerun your ./configure, make, and make install. Alternative: edit **/etc/apcupsd/apccontrol** and set the correct path for the SysV shutdown. We are working on a permanent solution.

examples/safe.apccontrol has bad wall command

Bug: On non-Linux systems, the examples/safe.apccontrol doesn't work because it uses **wall "message"** whereas on other systems **wall** only accepts the message from stdin.

Fix: We have modified **examples/safe.apccontrol** to use **wall <<EOF** input, which should work fine on all systems. You can download the new version of safe.apccontrol from [safe.apccontrol.tar.gz](http://www.apc.com/secure/patches/safe.apccontrol.tar.gz), but you may want to edit the paths to correspond to your system.

Future: For the next version of **apcupsd**, we have also created a **examples/safe.apccontrol.in** so that all the paths will be correctly set by configure for your system.

Networking does not work in a mixed vendor setup

Bug: In a mix vendor setup (RedHat, Debian, ...), the default networking (master/slave or Network Information Server) port assignments are different. Reason: Due to port conflicts on some machines, we set the default port numbers on each distribution to the values recommended by our experts. This solves a lot of problems, but results in incompatibilities if you use the **apcupsd** defaults. Fix: If you run a mixed vendor setup, use the **---with-nis-port=nn** and **---with-net-port=nn** options on your **./configure** command and ensure that all your machines use the same port numbers. After the **./configure**, you will need to redo your **make** and **make install**. The default values for most systems are:

```
./configure --with-nis-port=7000 --with-net-port=6666
```

Unusual Case of Apcupsd Hanging during Boot

On some systems (kernel version 2.4.5-acxx), **apcupsd** may hang during the boot process. This appears to be a probably of invoking **apcupsd** before the network is initialized. The simplest solution is to ensure that **apcupsd** is started after the network functions, or to put an explicit background (ampersand) request in the **apcupsd** startup script.

Apcupsd Version 3.8.1 Bugs

Unfortunately, it seems that every program has some bugs. We do our best to keep the bugs to a minimum by extensive testing. However, because of our inherent nature to occasionally overlook things and the fact that we don't have all the UPS models nor the APC documentation on those models, **apcupsd** will have some bug.

As the bugs become known to us, we will post them here with any possible fixes or workarounds that we have.

Apcupsd may Hang when Attempting to Initialize the Serial Port

This problem has been reported on NetBSD systems and on a Solaris8 (64bit) UltraSparc60 system. The problem is that the **open()** of the serial port does not return. The solution is to use the **O_NDELAY** flag when opening the port. The **open()** at 85 of **apcserial.c** should be replaced with the following:

```
if ((ups->fd = open(ups->device, O_RDWR | O_NOCTTY | O_NDELAY)) < 0) {
    Error_abort2(_("Cannot open UPS tty %s: %s\n"),
                ups->device, strerror(errno));
}
/* Cancel the no delay we just set */
cmd = fcntl(ups->fd, F_GETFL, 0);
fcntl(ups->fd, F_SETFL, cmd ~O_NDELAY);
```

Please note the addition of the two **fcntl()** calls to remove the **O_NDELAY** so that **apcupsd** will function properly. The patch relative to **apcupsd-3.8.1** is:

```
@@ -77,18 +81,21 @@
```

```

        Error_abort0(_("Serial port already initialized.\n"));
    }

-    /* Open the the device */
-    if ((ups->fd = open(ups->device, O_RDWR | O_NOCTTY)) < 0) {
+    /* Open the serial port device */
+    if ((ups->fd = open(ups->device, O_RDWR | O_NOCTTY | O_NDELAY)) < 0) {
        Error_abort2(_("Cannot open UPS tty %s: %s\n"),
                      ups->device, strerror(errno));
    }
+    /* Cancel the no delay we just set */
+    cmd = fcntl(ups->fd, F_GETFL, 0);
+    fcntl(ups->fd, F_SETFL, cmd ~O_NDELAY);

```

Version 3.8.1 May Cause Networking Problems on Win95

We have a report by a user that installing version 3.8.1 on a Windows 95 machine caused the loss of all networking capabilities on that machine. Re-installation of version 3.8.0 restored the networking capabilities. It should be noted that the major difference between the two versions was addition of support on Win32 for simple signaling UPSes (no change for Smart UPSes or networking) and upgrading from version 1.1.2 to version 1.1.5 of CYGWIN. We would appreciate hearing of your experiences with Win95 and **apcupsd**.

STATUS Output for BackUPS Pro and SmartUPS VS Incorrect

The STATUS Output for the BackUPS Pro and SmartUPS VS was unfortunately truncated due to a misplaced "break" statement. To fix this problem and restore the full STATUS output, for version 3.8.1, delete line 161 of apcstatus.c, which should be "break;"

```

    } else {
        s_write("LINEFAIL : DOWN\n");
        if (ups->BattLow == 0) {
            s_write("BATTSTAT : RUNNING\n");
            ups->Status = UPS_ONBATT;
        } else {
            s_write("BATTSTAT : FAILING\n");
            ups->Status = UPS_BATTLow;
        }
    }
    s_write("STATFLAG : 0x%02X Status Flag\n", ups->Status);
    break;
case NBKPRO:
case SMART:
case SHARESMART:
case MATRIX:
    if (ups->UPS_Cap[CI_IDEN]) {
        s_write("UPSNAME : %s\n", ups->name);
    } else {
        s_write("UPSNAME : N/A\n");
    }
}

```

<===== delete this line

Thanks to Joe Acosta for reporting this bug and testing the fix. The patch relative to apcupsd-3.8.1 is:

```

@@ -158,7 +158,7 @@
    }

```

```

        }
        s_write("STATFLAG : 0x%02X Status Flag\n", ups->Status);
-       break;
+       /* Note! Fall through is wanted */
        case NBKPRO:
        case SMART:
        case SHARESMART:

```

Automatic Self Test is Reported as Power failure

Depending on your EEPROM setting, the UPS will enter an automatic self test mode for approximately 30 seconds every two weeks (the default). The self test involves a switch to battery power, and **apcupsd** reports this as a power failure. We hope to correct this in a future version.

Improper Shutdown of Apcupsd on WinNT Systems

If you attempt to shutdown **apcupsd** on a WinNT system either through the system tray icon or via the Services Manager, two of the three **apcupsd** processes may become stuck in the computer. One of these processes may consume all available CPU time. Even as the system administrator, you will be unable to kill these processes (that's Microsoft for you!). Normally, there should be no need to stop **apcupsd**. However, should you want to do so, for example, to make an upgrade, go to the Services dialog from the Control Panel, and deactivate **apcupsd** then reboot your computer.

Name Resolution Does Not Work on Win32 Systems

In a master/slave configuration, normally one uses the master and slave machine names as a qualified domain name in **apcupsd.conf**. Unfortunately, on Win32 systems, **apcupsd** is unable to resolve these names to an IP address needed to communicate with the other end. To resolve the problem, please use an explicit IP address written as a dotted quadruple (e.g. 192.168.1.100) instead of the machine name.

The apccontrol Script Doesn't Work with All User Scripts

apcupsd allows customization of the actions taken during **events** (e.g. onbattery, mainsback, etc). One method is to modify the **/etc/apcupsd/apccontrol** script directly. However, this makes it more difficult to upgrade to a new version of **apcupsd**. An alternate method is to place your own script in the **/etc/apcupsd** directory with the name of the event that you wish to control. For example, you may want to shutdown an Oracle database prior to issuing the system shutdown command. If the script you create is a shell script, everything is OK. However, if you use a Perl script the script may not run. To correct this problem, change the following line in **apccontrol**:

```
${SCRIPTSHELL} ${SCRIPTDIR}/${1}
```

at line 38 of **apccontrol** (depending on your version), to:

```
${SCRIPTDIR}/${1}
```

For this to work, your script file must have the executable bit set.

Denial of Service Security Problem

One of our users emailed us with the following information (with which, we agree):

I noticed that its (apcupsd's) handling of temp files during e-mail notification was prone to denial-of-service attacks. I found this vulnerability within these scripts found in the /etc/apcupsd directory in the above RPM:

```
changeme
commfailure
commok
mainsback
onbattery
```

Since the notification scripts blindly write to a \$\$-selected /tmp filename, any user on the box could cause root to overwrite any file on the system (/etc/passwd, /boot/vmlinuz) with the UPS error message by making lots and lots of /tmp/apcupsd.onbattery.##### symlinks which point to the victim file and waiting. (Yes, it's fairly unlikely, but a hole's a hole.)

As work-arounds, I'd recommend either moving the temp file to within /etc/apcupsd (which is writable only by root or else the sysadmin has bigger problems than temp file creation) or piping from a subshell like this:

```
(
    echo "$MSG"
    echo " "
    /sbin/apcaccess status
) | $APCUPSD_MAIL -s "$MSG" $SYSADMIN
```

Apcupsd version 3.8.0 Bugs

In addition to the bugs reported above, version 3.8.0 has the following problems. Please note that all of these bugs have been corrected in version 3.8.1.

Win32 apcupsd Does Not Work with Simple Signaling UPSes

The Win32 version of apcupsd will not support simple signaling UPSes (sometimes called dumb UPSes) such as the Back-UPS. Please do not attempt to use it as if there is a power failure, it will most likely cause the UPS to suddenly shut off the power to your computer. This is due to the fact that CYGWIN does not support serial line signal level IOCTL calls. We hope to rectify this situation in the not so distant future (no promises because of the inadequacies of the Windows API in this area, which is probably why CYGWIN does not support the signal level IOCTLs).

Bad Path to Shutdown in apccontrol Script

A Japanese user has reported that the call to /usr/bin/shutdown in the apccontrol script must be changed to /sbin/shutdown on his RedHat 6.2J system (/usr/bin/shutdown is valid on my English RedHat 5.2, 6.0, 6.1, and 7.0 systems). This kind of problem underscores the necessity to test the installation, as this user wisely did.

CGI Programs May Error When Called Directly

If instead of calling the CGI program `multimon.cgi`, the user calls one of the helper programs directly, that program may detect bad arguments and error. This causes Apache to report an "Internal Error" to the user. This is a bit unpleasant for some users but it causes no harm. All the CGI programs have been reworked to provide less dramatic error messages, and they will be posted to this site as updates shortly.

Alpha Tru64 32/64 Bit Bug

On Alpha machines, one instance of a 32/64 bit problem escaped our notice during the port. It apparently only affects the CGI programs, and causes **apcupsd** to be unable to resolve names. A fix to this problem will be posted.



Apcupsd Network Monitoring (CGI) Programs

Configuration

With this release, there are five [CGI programs](#) (**multimon.cgi**, **multimoncss.cgi**, **upsstats.cgi**, **upsfstats.cgi**, and **upsimage.cgi**). To have them properly installed, you must run the **./configure** command with **--enable-cgi** and you should specify an installation directory with **--with-cgi-bin=** or load them manually. To install the Cascading Style Sheet, which is used by **multimoncss.cgi**, you must use the **--with-css-dir=** option. The default directory for installation of the CGI programs is **/etc/apcupsd**, which is not really where you want them if you are going to use them. Normally, they should go in the **cgi-bin** of your Web server.

Once built and loaded, they will give you the status of your UPS or UPSes over the network.

Normally only **multimon.cgi** or **multimoncss.cgi** is directly invoked by the user. However, it is possible to directly invoke **upsstats.cgi** and **upsfstats.cgi**. **upsimage.cgi** should never be directly invoked as it is used by **upsstats.cgi** to produce the bar charts.

Setting up and Testing the CGI Programs

Before using **multimon** and the other CGI programs, first ensure that **apcupsd** is configured to run the Network Information Server. This is done by setting **NETSERVER on** in **/etc/apcupsd/apcupsd.conf**. See the [Network Information Server](#) section of the configuration section of this manual for additional details. Also, see the section at the end of this chapter concerning the Client test program.

Next you must edit the **hosts** file **/etc/apcupsd/hosts.conf** and at the end, add the name of the hosts you want to monitor and a label string for them. On my site, I use **multimon.conf** unmodified from what is on the source distribution. However, I have modified the **hosts.conf** file to contain the following three lines:

```
MONITOR matou "Server"
MONITOR polymatou "Backup server"
MONITOR deuter "Disk server"
```

matou, **polymatou**, and **deuter** are the network names of the three machines currently running **apcupsd**.

Please note that the network names may either be IP addresses or fully qualified domain names. The network name (or IP address) may optionally be followed by **:<port>**, where the port is the NIS port address you wish to use. This is useful if you are running multiple copies of **apcupsd** on the same system or if you are running in a mixed vendor environment where the NIS port assignments differ. An example could be the following:

```
MONITOR matou "Server"
MONITOR polymatou "Backup server"
MONITOR deuter "Disk server"
MONITOR polymatou:7001 "APC USB UPS"
```

where the USB copy of **apcupsd** has been configured to use port 7001 (with **--with-nis-port=7001** on the **./configure** or by modifying **apcupsd.conf**). Note, the default NIS port is 7000 on most platforms.

To test **multimon.cgi**, you can execute it as non-root directly from the source **cgi** build directory. To do so, enter at a shell prompt:

./multimon.cgi

If everything is setup correctly, it will print a bunch of html with the values of the machines that you have put in the **hosts.conf** file. It should look something like the following (note, only a small portion of the output is reproduced here):

```
Content-type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
    "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML>
<HEAD><TITLE>Multimon: UPS Status Page</TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
<TABLE BGCOLOR="#50A0A0" ALIGN=CENTER>
<TR><TD>
<TABLE CELLPADDING=5>
<TR>
<TH COLSPAN=10 BGCOLOR="#60B0B0">
<FONT SIZE="+2">APCUPSD UPS Network Monitor</FONT>
<BR>Sun Jan 16 12:07:27 CET 2000</TH>
</TR>
<TR BGCOLOR="#60B0B0">
<TH COLSPAN=1>System</TH>
<TH COLSPAN=1>Model</TH>
<TH COLSPAN=1>Status</TH>
...
```

If you do not get similar output, check the permissions of the /etc/apcupsd directory and of those of /etc/apcupsd/hosts.conf to ensure that your web server can access it. At many sites such as mine, the Apache server is not running as root, so you must be careful to ensure that that /etc/apcupsd/hosts.conf and /etc/apcupsd/multimon.conf are world readable.

To invoke **multimon** in your Web browser, enter:

`http://<your-site>/cgi-bin/multimon.cgi`

You should get something similar to the screenshot shown below.

If you wish additional control over the colors, type faces, and sizes of the multimon output, you might wish to use **multimoncss.cgi** in place of multimon. In this case, you simply edit the multimon.css file to specify the styles you prefer. There are several sample Style Sheet files in the **cgi** subdirectory of the source tree.

To see a working example of the these programs, visit <http://www.sibbald.com/cgi-bin/multimon.cgi>

or <http://www.sibbald.com/cgi-bin/multimoncss.cgi>

multimon.cgi

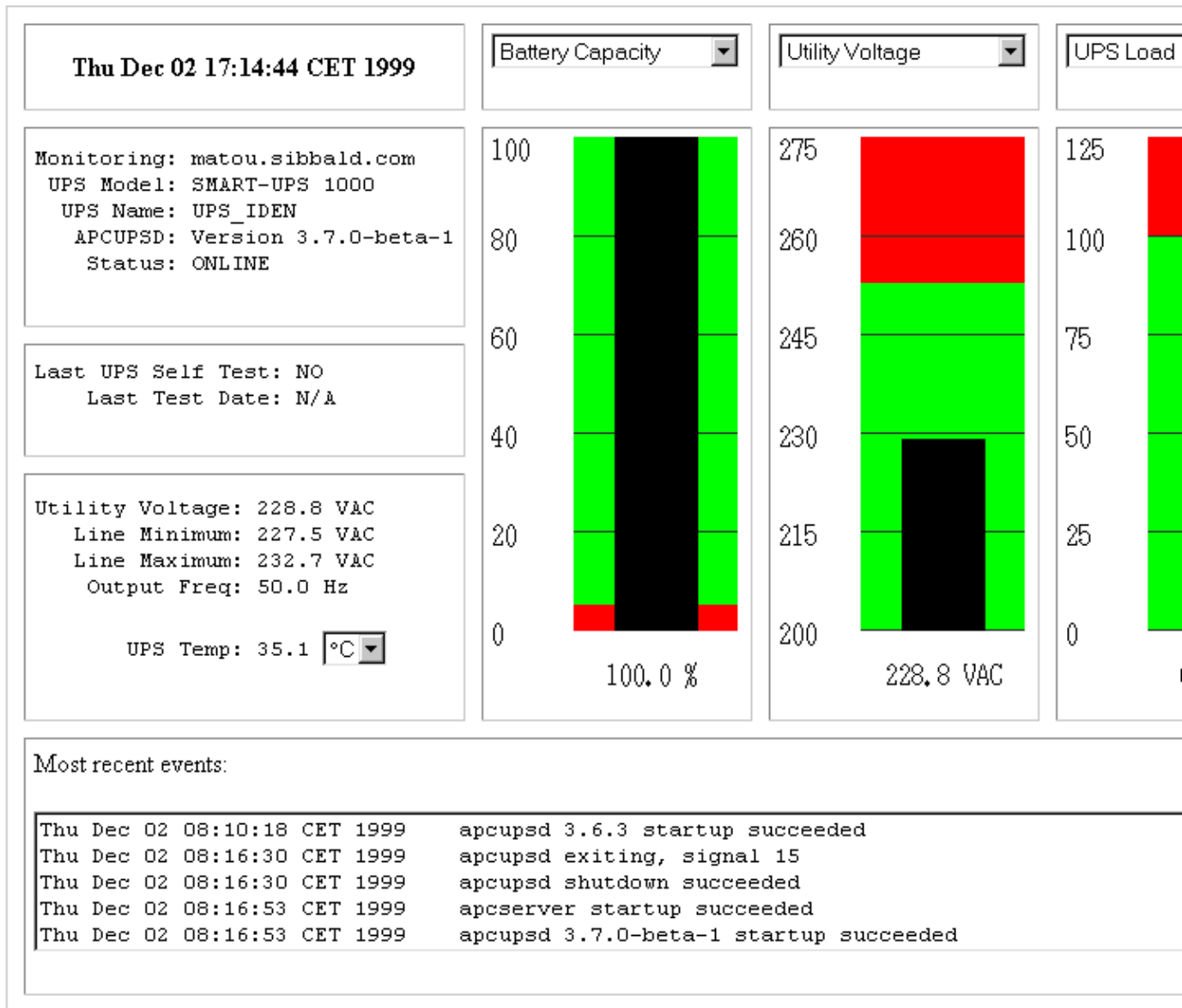
This program monitors multiple UPSes at the same time. A typical output of **multimon.cgi** as displayed in your Web browser might look like the following:

APCUPSD UPS Network Monitor							
Thu Dec 02 17:08:27 CET 1999							
System	Model	Status	Battery Chg	Utility	UPS Load	UPS Temp	Batt. Ru
Server	SMART-UPS 1000	ONLINE	100.0 %	228.8 VAC	6.2 %	35.1 °C	167.0
Backup server	SMART-UPS 1000	ONLINE	100.0 %	228.8 VAC	11.4 %	29.2 °C	112.0
Disk server	SMART-UPS 1000	ONLINE	100.0 %	230.1 VAC	6.2 %	35.1 °C	167.0

The machines monitored as well as the values and their column headings are all configurable (see `/etc/apcupsd/hosts.conf` and `/etc/apcupsd/multimon.conf`)

upsstats.cgi

By clicking on the **system** name in the **multimon.cgi** display, you will invoke **upsstats.cgi** for the specified system, which will produce a bar graph display of three of the monitored values. For example,



You can display different bar graphs by selecting different variables from the drop down menus at the top of each of the three bar graphs.

As with **multimon**, if you have your local host configured in the `/etc/apcupsd/hosts.conf` file, you can execute it from a Unix shell from the source cgi directory as follows:

```
./upsstats.cgi
```

As with **multimon**, quite a few lines of html should then be displayed.

upsfstatus.cgi

If you would like to see all of the STATUS variables available over the network, click on the **Data** field of the desired system, and your browser will display something like the following:

APC UPS management under Linux

```
APC      : 001,048,1109
DATE     : Thu Dec 02 17:27:21 CET 1999
HOSTNAME : matou.sibbald.com
RELEASE  : 3.7.0-beta-1
CABLE    : Custom Cable Smart
MODEL    : SMART-UPS 1000
UPSMODE  : Stand Alone
UPSNAME  : UPS_IDEN
LINEV    : 223.6 Volts
MAXLINEV : 224.9 Volts
MINLINEV : 222.3 Volts
LINEFREQ : 50.0 Hz
OUTPUTV  : 223.6 Volts
LOADPCT  : 6.2 Percent Load Capacity
BATTV    : 27.9 Volts
BCHARGE  : 100.0 Percent
MBATTCHG : 5 Percent
TIMELEFT : 167.0 Minutes
MINTIMEL : 3 Minutes
SENSE    : High
DWAKE    : 060 Seconds
DSHUTD   : 020 Seconds
LOTRANS  : 196.0 Volts
HITRANS  : 253.0 Volts
RETPCT   : 050.0 Percent
STATFLAG : 0x08 Status Flag
STATUS   : ONLINE
ITEMP    : 35.1 C Internal
ALARMDEL : Low Battery
LASTXFER : U command or Self Test
SELFTEST : NO
STESTI   : 336
DLOWBATT : 02 Minutes
DIPSW    : 0x00 Dip Switch
REG1     : 0x00 Register 1
REG2     : 0x00 Register 2
REG3     : 0x00 Register 3
MANDATE  : 01/11/99
SERIALNO : GS9903001147
BATTDATA : 01/11/99
NOMOUTV  : 230.0
NOMBATTV : 24.0
HUMIDITY : N/A
AMBTEMP  : N/A
EXTBATT  : 0
BADBATT  : N/A
FIRMWARE : 60.11.I
APCMODEL : IWI
END APC  : Thu Dec 02 17:27:25 CET 1999
```

You should get pretty much the same output mixed in with html if you execute **upsfstats.cgi** directly from a Unix shell in the cgi subdirectory as explained above for **upsfstats.cgi** and **multimon.cgi**.

Working Example

To see a working example of the above programs, visit <http://www.sibbald.com/cgi-bin/multimon.cgi>.

Client Test Program

When your Network Information Server is up and running, you can test it using a simple program before attempting to access the server via your Web server. The test program is called **client.c** and can be found in the **examples** subdirectory of the source distribution. To build the program, when in the examples directory, use something like the following:

```
cc client.c ../lib/libapc.a -o client
```

Then execute it:

```
./client <host>[:<port>] [<command>]
```

Where **host** is the name of the host or the IP address of the host running the Network Information Server. The default is the local host. You may optionally specify a port address separated from the host name with a colon. You may also optionally specify a single command to be executed. If you specify a command, that command will be executed and the client program will exit. This is a very simple and useful way of pulling the **status** or **events** data into another program such as Perl.

If no error messages are printed, it has most likely established contact with your server. Anything that you type as standard input will be passed to the server, and anything the server sends back will be printed to standard output. There are currently two commands recognized by the server: **events** and **status**. Hence the following commands:

```
./client
status
events
xyz
^D
```

Should produce the status listing (the same as produced by **apcaccess status**), followed by the list of the last 10 events (in response to the **events** command), and finally **Invalid command** in response to the **xyz** input, which is not a valid command. The control-D terminates the **client** program.

A Tip from Carl Erhorn for Sun Systems

It is possible to run the CGI code to monitor your UPS using the answerbook http server that runs on Solaris. As long as your server has the Answerbook2 web server installed and running, you can insert the cgi scripts into the cgi directory of the web server, and access the cgi using something like:

```
http://hostname:8888/cgi/multimon.cgi
```

Credits

Many thanks go to Russell Kroll <rkroll@exploits.org> who wrote the CGI programs to work with his UPS Monitoring system named [Network UPS Tools \(NUT\)](#). Thanks also to Jonathan Benson <jbenson@technologist.com> for initially adapting the upsstatus.cgi program to work with **apcupsd**.

We have enhanced the bar graph program and hope that our changes can be useful to the original author in his

project.



Apcupsd DATA Logging

This feature is somewhat outdated and not often used.

DATA logging

Data logging consists of periodically logging important data concerning the operation of the UPS. For the definitive definition of the format, see `log_data()` in `apcreports.c`. The format varies according to the UPS model and the information available from the UPS.

For UPS models, NBKPRO, SMART, SHARESMART, and MATRIX, the output is written in a format very similar to what PowerChute writes. That is:

MinLineVoltage, MaxLineVoltage, OutputVoltage, BatteryVoltage, LineFrequency,
UPSTemperature,AmbientTemperature,Humidity,LineVoltage

Any value that is not supported by your UPS such as AmbientTemperature and Humidity will be blank; the commas before and after that field will still be output.

An actual example from the log file is:

```
Nov  2 12:43:05 matou apcupsd[23439]: 224.9,227.5,226.2,27.74,50.00,100.0,30.6,,,226.2
```



Configuring Your EEPROM

If you have a SmartUPS, there are depending on the UPS at least 12 different values stored in the EEPROM that determine how the UPS reacts to various conditions such as high line voltage, low line voltage, power down grace periods, etc.

In general, for the moment, we do not recommend that you change your EEPROM values unless absolutely necessary. There have been several reported cases of problems setting the Low Transfer Voltage. Consequently, if at all possible, do not attempt to change this value.

If despite these warnings, you must change your EEPROM, we recommend connecting your UPS to a Windows or NT machine running PowerChute and making the changes.

If you have no other choice, you may also change the EEPROM values with **apcupsd**. Before doing so, please make a printed copy of your UPS as it is before any EEPROM changes so that you can check the changes that you have made. Do so by printing a copy of the output from **apcaccess status** and also print a copy of the output from **apcaccess eprom**.

Once this is done, choose which values of the EEPROM you want to change. Choose the new values from the list provided by **apcaccess eprom**. For the battery date, and the UPS name, you can use any eight characters.

To make the EEPROM changes with **apcupsd** you must first stop the **apcupsd** daemon. See [Stopping Apcupsd](#) in the appropriate section of this manual. Then edit the appropriate configuration directive in **/etc/apcupsd/apcupsd.conf**.

NOTE: we recommend that you change one EEPROM value at a time, by defining only one configuration directive at a time. After each change, check that everything is OK before proceeding to the next value you wish to change.

To actually change the EEPROM, as **root** with the **apcupsd** daemon stopped, enter:

apcupsd -c

When it has completed the reprogramming of the EEPROM, it will print the new STATUS report. Check that you got the expected results before continuing.

The list of configuration directives and the permitted values for the SmartUPS follows:

Valid EEPROM values for the SMART-UPS 1000

Description	Config Directive	Current Value	Permitted Values
Upper transfer voltage	HITRANSFER	253	253 264 271 280
Lower transfer voltage	LOTRANSFER	204	196 188 208 204
Return threshold	RETURNCHARGE	15	00 15 50 90
Output voltage on batts	OUTPUTVOLTS	230	230 240 220 225
Sensitivity	SENSITIVITY	H	H M L L
Low battery warning	LOWBATT	2	02 05 07 10
Shutdown grace delay	SLEEP	180	020 180 300 600
Alarm delay	BEEPSTATE	T	0 T L N
Wakeup delay	WAKEUP	60	000 060 180 300
Self test interval	SELFTEST	336	336 168 ON OFF

apcaccess eeprom will produce the list that is appropriate for your UPS.



Apcupsd EVENTS

When **apcupsd** detects anomalies from your UPS device, it will make some decisions that usually result in one or more calls to the script located in **/etc/apcupsd/apccontrol**. The **apccontrol** file is a shell script that acts on the first argument that **apcupsd** passes to it. These actions are set up by default to sane behaviour for all possible situations **apcupsd** is likely to detect from the UPS. Nevertheless you can change the **apccontrol** behaviour for every single action. To do so create a file with the same name as the action, which is passed as a command line argument. Put your script in the **/etc/apcupsd** directory.

These events are sent to the system log, optionally sent to the temporary events file (**/etc/apcupsd/apcupsd.events**), and they also generate a call to **/etc/apcupsd/apccontrol** which in turn will call any scripts you have placed in the **/etc/apcupsd** directory.

Normally, **/etc/apcupsd/apccontrol** is called only by **apcupsd**. Consequently, you should not invoke it directly. However, it is important to understand how it functions, and in some cases, you may want to change the messages that it prints using **wall**. We recommend that you do so by writing your own script to be invoked by **apccontrol** rather than by modifying **apccontrol** directly. This makes it easier for you to upgrade to the next version of **apcupsd**.

In other case, you may want to write your own shell scripts that will be invoked by **apccontrol**. For example, when a power fail occurs, you may want to send an email message to root. At present the arguments that **apccontrol** recognizes are:

How apcupsd calls apccontrol

When **apcupsd** detects an event, it calls the **apccontrol** script with four arguments as:

```
apccontrol <event> <ups-name> <connected> <powered>
```

where:

event

is the event that occurred and it may be any one of the values described in the next section.

ups-name

is the name of the UPS as specified in the configuration file (not the name in the EEPROM). For version 3.8.2, this is always set to **Default**

connected

is 1 if **apcupsd** is connected to the UPS via a serial port (or a USB port). In most configurations, this will be the case. In the case of a Slave machine where **apcupsd** is not directly connected to the UPS, this value will be 0.

powered

is 1 if **apcupsd** is powered by the UPS and 0 if not. In version 3.8.2, this value is always 1.

apccontrol Command Line Options

apccontrol accepts the following command line options:

annoyme

	Does a printf "Power problems please logoff." wall then exits.
<i>changeme</i>	Does a printf "Emergency! UPS batteries have failed\nChange them NOW" wall then exits.
<i>commfailure</i>	Does a printf "Warning serial port communications with UPS lost." wall then exits.
<i>commok</i>	Does a printf "Serial communciations with UPS restored." wall then exits.
<i>doreboot</i>	Does a reboot of the system by calling shutdown -r now
<i>doshutdown</i>	Does a shutdown of the system by calling shutdown -h now
<i>emergency</i>	Does an emergency shutdown of the system by calling shutdown -h now
<i>failing</i>	Does a printf "UPS battery power exhausted. Doing shutdown.\n" wall then exits.
<i>loadlimit</i>	Does a printf "UPS battery discharge limit reached. Doing shutdown.\n" wall then exits. After completeting this event, apcupsd will immediately initiate a doshutdown event.
<i>mainsback</i>	Attempts to cancel the shutdown with a shutdown -c
<i>onbattery</i>	Does a printf "Power failure. Running on UPS batteries." wall then exits.
<i>powerout</i>	Does a printf "Warning power loss detected." wall then exits.
<i>remotedown</i>	Does a shutdown -h now
<i>restartme</i>	Terminates the currently running apcupsd and then restarts it.
<i>runlimit</i>	Does a printf "UPS battery runtime percent reached. Doing shutdown.\n" wall then exits. After completeting this event, apcupsd will immediately initiate a doshutdown event.
<i>timeout</i>	Does a printf "UPS battery runtime limit exceded. Doing shutdown.\n" wall then exits. After completeting this event, apcupsd will immediately initiate a doshutdown event.
<i>startselftest</i>	This is called when apcupsd detects that the UPS is doing a self test. No action is taken.
<i>endselftest</i>	This is called when apcupsd determines that a self test has been completed. No action is taken.

To write your own routine for the **powerout** action, you create shell script named **powerout** and put it in the lib directory (normally **/etc/apcupsd**). When the **powerout** action is invoked by **apcupsd**, **apccontrol** will first give control to your script. If you want **apccontrol** to continue with the default action, simply exit your script with an exit status of zero. If you do not want **apccontrol** to continue with the default action, your script should exit with the special exit code of **99**. However, in this case, please be aware that you must ensure proper shutdown of your machine if necessary.

Some sample scripts (onbattery and mainsback) that email power failure messages can be found in the **examples** directory of the source code.



Apcupsd Frequently Asked Questions

See [the bugs section](#) of this document for a list of known bugs and solutions.

*What UPS brands does **apcupsd** support?*

Currently **apcupsd** supports only APC UPSes. However, some companies such as Hewlett Packard put their own brand name on APC manufactured UPSes. Thus even if you do not have an APC branded UPS, it may work with **apcupsd**. You will need to know the corresponding APC model number. **apcupsd** supports all the popular APC models. See the installation and configurations sections of this document for more details.

*Does **apcupsd** support Windows?*

With release 3.8.0, **apcupsd** now runs on Win95/98, WinMe, WinNT, and Win2000 machines. All features of the Unix versions of **apcupsd** are implemented. The UPS EEPROM programming features of **apcupsd** have not been tested under Windows. Version 3.8.0 does not support simple signaling UPSes (BackUPS, etc). Version 3.8.1 does support most simple signaling UPSes, but not all cables (due to deficiencies in the Windows serial port API). Please note that we have had reports that **apcupsd** does not work properly on the WinXP system. If you have any information on this, please email us.

I don't have a cable, which one should I build?

First you must know if you have a Smart UPS or a simple signaling UPS — See the table of supported UPSes in the [Configuration Chapter](#) of this manual. If you have a Smart UPS (or a SubSmart), we recommend building a [Smart–Custom Cable](#). If you have a Simple Signaling UPS, we recommend that you build a [Simple–Custom Cable](#).

*How much CPU resources does **apcupsd** use?*

Depending on your CPU speed, you may see more or less of the CPU consumed by **apcupsd**. On a 400MHz Unix system, the CPU usage should fall well below 0.1%. On slower systems, the percentage will increase proportionally to the decrease in the CPU speed. On a 400Mhz Win98 machine, the CPU usage will be on the order of 0.5–1.0%. This is higher than for Unix systems. However, compared to the 30% CPU usage by APC's PowerChute (the version on the CDROM shipped with my UPS), **apcupsd's** 0.5–1.0% is very modest.

If you configure **apcupsd** to run with pthreads (--with-pthreads on the ./configure line), **apcupsd** will run considerably faster, otherwise said, it will consume less of your CPU, and it will use approximately one third of the memory. For example, Carl Erhorn reports that on his Solaris system, "With the old 3–process version, we averaged about 4.8MB of total memory used. With the new single process, we use only about 1.7MB! That's also a very good improvement."

*What language is **apcupsd** written in?*

It is written entirely in C.

We are using apcupsd–3.8.1–1 in RedHat 6.2. The slave, when shutting down, is reporting an error at line 436 of apcupsd.c. The error is initiated by apcupsd --killpower ! What can we do to fix this, and is it critical?

No, the error is not serious. Unfortunately, the documentation in the area of master/slaves is not very detailed, and for that reason, your slave setup is not totally correct as explained below.

On master machines, we modify `/etc/rc.d/init.d/halt` to re-invoke `apcupsd` with the `--killpower` option (actually the script `apccontrol` is called). This causes the UPS to send the codes to the UPS to make it power off.

On slave machines, these modifications should not be made to the `/etc/rc.d/init.d/halt` script since the slave has no connection to the UPS.

To eliminate the problem, on all your slave machines, either restore the original halt file, or simply delete all the lines containing `***apcupsd***`, which were inserted by the `apcupsd` installation process.

*To test **apcupsd**, I unplugged the UPS to simulate a power outage. After the machine went into the shutdown process I plugged the UPS back into the commercial power source. This caused the shutdown process to hang after the daemon tried to shut-off the ups. Have you run into this problem, and if so do you have a remedy?*

Normally, once the shutdown process has begun, we cannot stop it, though there IS some code that tries to do so, we don't consider it a very good idea – how do you stop a shutdown that has killed off half of the daemons running on your system? Most likely you will be left with an unusable system. In addition, when `apcupsd` is re-executed in the halt script after the disks are synced, it tries to shut off the UPS power, but the UPS will generally refuse to do so if the AC power is on. Since we cannot be 100% sure whether or not the UPS will shut off the power, we don't attempt to reboot the system if we detect that the power is back as it might then get caught by a delayed power off (at least for Smart UPSes).

*After running `apcupsd` for a while, I get the following error:
"Serial communications with UPS lost." What is the problem?*

We use standard Unix serial port `read()` and `write()` calls so once a connection is made, we generally have few problems. However, there have been reports that APC's SNMP Management Card can cause serial port problems. If you have such a card, we suggest that you remove it and see if the problem goes away. It is also possible that some other process such as a **getty** is reading the serial port.

When `apcupsd` starts, I get the following error: "attach_shmarea: cannot get shm area: Identifier removed." What is the problem?

This problem and the problem of **cannot create shm area** are due to the fact that the shared memory key that `apcupsd` wants to use is already in use. This happens most frequently when there is an old zombie `apcupsd` process still in the system.

The solution is to remove the old process.

You can often see what is going on by doing a: **ipcs** command as root when `apcupsd` is not running. If you see a segment with the key `0x10feed01`, you can be sure there is some old `apcupsd` process still using it. If you cannot kill the old process, you can try using **ipcrm** (see the man pages).

Recent versions of `apcupsd` starting with `apcupsd-3.8.2Beta6` should no longer have this problem as they will automatically try using a different key.

I get the following error: "Starting `apcupsd` power management. Mar 20 21:19:40 box `apcupsd`[297]: `apcupsd` FATAL ERROR in `apcserial.c` at line 83. Cannot open UPS tty `/dev/cua01`: No such file or directory." What is the problem?

The two most likely causes of your problem are:

1. You have the wrong serial port device name in the `apcupsd.conf` file.

2. The device name is not defined on your system.

Suggestions for proceeding:

For the first item, check what your serial port device should be named. You might be able to find the name with an:

```
ls /dev
```

normally there will be hundreds or even thousands of names that print.

If that doesn't produce anything useful, you can try step 2. Perhaps your device is not defined.

To get more information on your devices try:

```
man MAKEDEV
```

or

```
find / -name MAKEDEV
```

it is often located in /dev/MAKEDEV

Looking at the documentation may tell you what the correct name is, or at least allow you to create the device.

How do I ensure that the slaves shutdown before the master?

There are several strategies for getting the slaves properly shutdown before shutting down the master. The first is to make the master wait a period of time for the slaves to shutdown before doing its own shutdown. Currently, the master always waits 30 seconds before starting its own shutdown. If this is insufficient, you can add additional time by putting an appropriate **sleep** shell command in the /etc/apcupsd/apccontrol file just before the actual system shutdown command is executed (there are something like 3 places).

The second strategy is to put a TIMEOUT value in the apcupsd.conf file on the slave that is sufficiently short that you are sure that the slave will shutdown before the master. If the shutdown is done with a poweroff, this will also save power so that the master can stay up longer.

How do I ensure that my database server is correctly shutdown?

You simply add whatever commands are necessary in the appropriate case statements in **/etc/apcupsd/apccontrol**, which is a standard script file that is called to actually do the shutdown. Alternatively, you can add your own script file that will be called before doing the commands in **apccontrol**. Your script file must have the same name as the appropriate case statement in **apccontrol**; it must be executable; and it must be in the same directory as **apccontrol**.

I have Win2k Advanced server, and when starting the service, get: Could not start the Apcupsd Server service on Local Computer. Error 1067: The process terminated unexpectedly

The most common error causing your problem is an incorrect serial port specification on your DEVICE directive. It should be
DEVICE /dev/com2

On WinNT machines, and probably Win2000 machines you MUST use /dev/com2 unless you modify the behavior of the boot process to prevent Windows from probing the port. This is documented in our

manual for WinNT. Although I imagine it is the same for Win2000, I am not sure.

The second most common problem is bad placement of the files — i.e. you did not install them in `c:\apcupsd` Unfortunately for the current release, this path is "hard coded" into the binaries.

The third most common problem is that you did not run the `setup.bat` script after loading the files. This is necessary to install `apcupsd` as a service.

If all the above fails, try starting `apcupsd` by hand inside a CYGWIN `rxvt` window — if you use an `rxvt` window rather than a DOS window, you will see many more of the error messages.

In addition, most of the `apcupsd` startup errors are reported in:

`c:\apcupsd\etc\apcupsd\apcupsd.events`

many error messages associated with Windows services will be reported in the Windows System Log.

When using USB, I get the following log messages: `usb-uhci.c: interrupt, status 3, frame# 826`. What does it mean?

It means one transfer worked (bit 0 in status) and another one (after that) failed (bit 1) at timeframe 826. This kind of soft error is common on USB and if everything seems to be working, you can ignore it.

`apcnisd` doesn't work. It always gives: `FATAL ERROR in apcipc.c at line 497. attach_shmarea: shared memory version mismatch (or UPS not yet ready to report)`

Unfortunately `apcnisd` does not work with `pthreads` enabled. You have the following options:

1. If you build with `pthreads` enabled, `apcnisd` will not work no matter what you do.
2. If you build with `pthreads` enabled, and you want to have network information from `apcupsd`, you must set `NETSERVER ON`. This is the configuration we recommend (i.e. using `pthreads` and `NETSERVER ON`).
3. If you build with `pthreads` disabled, you have the choice of using `apcnisd` or the `NETSERVER` code. If you wish to use `apcnisd`, you must set `NETSERVER OFF`
4. If you build with `pthreads` disabled, and you do not use `apcnisd`, you must set `NETSERVER ON` if you wish to have network information from `apcupsd`.

Concerning the names one sees with `"ps"`.

1. With `pthreads` enabled, on Linux machines, you will see multiple copies of `apcupsd` running, but they will all be called `apcupsd` rather than `apcmain`, `apcser`, ... They will still run as LWP, but we are unable to set the names on threads (LWP). Note, though `ps` shows "multiple copies" of `apcupsd` running, it is really one memory image but with multiple threads.
2. With `pthreads` disabled, we are able to set the child process names (at least on Linux) so you will see `apcmain`, `apcser`, `apcnis`, ... in the `ps` output. In this case, they are really different processes each with its own memory image (the

code image is most likely shared).



Apcupsd System Logging

The **apcupsd** philosophy is that all logging should be done through the **syslog** facility (see: **man syslog**). This is now implemented with the exceptions that **STATUS** logging, for compatibility, with prior versions is still done to a file, and **EVENTS** logging can be directed to a "temporary" file so that it can be reported by the network information server.

Apcupsd logging into four separate types called:

1. DEBUG
2. DATA
3. STATUS
4. EVENTS

DEBUG Logging

Debug logging consists of debug messages. Normally these are turned on only by developers, and currently there exist very few of these debug messages.

DATA Logging

Data logging consists of periodically logging important data concerning the operation of the UPS. See the [DATA Format](#) section of this manual for more details.

STATUS Logging

Status logging consists of logging all available information known about your UPS as a series of ASCII records. This information is also made available by the **apcupsd** [network information server](#).

For more details on STATUS logging, see the [STATUS Format](#) section of this manual.

EVENTS Logging

Events logging consists of logging events as they happen. For example, successful startup, power fail, battery failure, system shutdown, ...

See the [EVENTS Format](#) section of this manual for more details.

Implementation Details

In order to ensure that the data logged to syslog() can be directed to different files, I have assigned syslog() levels to each of our four types of data as follows:

1. DEBUG logging has level LOG_DEBUG
2. DATA logging has level LOG_NOTICE
3. STATUS logging has level LOG_NOTICE
4. EVENTS logging has levels LOG_WARNING, LOG_ERR, LOG_CRIT, and LOG_ALERT

It should be noted that more work needs to be done on the precise definitions of each of the levels for EVENTS logging. Currently, it is roughly broken down as follows:

LOG_WARNING general information such as startup, etc.

LOG_ERR an error condition detected, e.g. communications problem with the UPS.

LOG_CRIT a serious problem has occurred such as power failure, running on UPS batteries, ...

LOG_ALERT a condition that needs immediate attention such as pending system shutdown, ...

More work needs to be done to the code to ensure that it corresponds to the above levels.

As a practical example of how to setup your syslog() to use the new logging feature, suppose you wish to direct all DATA logging to a file named /var/log/apcupsd.data, all EVENTS to the standard /var/log/messages file (to be mixed with other system messages), and at the same time send all EVENTS to /var/log/apcupsd.events, and finally, you want to send all STATUS logging to the named pipe /var/log/apcupsd.status

First as root, you create the named pipe:

```
mkfifo /var/log/apcupsd.status
```

change its permissions as necessary or use the -m option to set them when creating the pipe.

Then you modify your /etc/syslog.conf file to direct the appropriate levels of messages where you want them. To accomplish the above, my syslog.conf file looks like:

```
# exclude all apcupsd info by default
*.info;local0.none                /var/log/messages

# Everything for apcupsd goes here
local0.info;local0.!notice        /var/log/apcupsd.data
local0.notice;local0.!warn        | /var/log/apcupsd.status
local0.warn                       /var/log/apcupsd.events
local0.warn                       /var/log/messages
```

Developer's Notes

All logging functions and all error reporting are now done through the log_event() subroutine call. Exceptions to this are: initialization code where printf's are done, and writing to the status file. Once the initialization code has completed and the fork() to become a daemon is done, no printf's are used. log_event() has exactly the same format as syslog(). In fact, the subroutine consists of only a syslog() call. If anyone really wishes to log to a file, the code to do so can easily be done by adding code to log_event() in apclog.c.



Master/Slave Configuration

General

If you have two or more computers that are powered by the same UPS and they are connected by a network, you can configure **apcupsd** so that the computer that controls the UPS (connected by the serial port or USB port), which is called the master, can provide information to other machines powered by the UPS, called slaves. When the master detects a power failure, it will notify all the slaves (maximum of twenty). If the master detects that the battery is low, it will also notify the slave so that the slave may perform a shutdown.

In addition, in cases where you wish to keep the master up longer than the slave, you can configure the slave to shutdown in a predetermined time after the UPS goes on batteries.

If a picture is worth a thousand words for you, please see the [Three Major Configuration Possibilities for Apcupsd](#) section of the Configuration chapter of this manual.

Configuration Directives

The minimum set of configuration directive changes needed to create a proper master and slave configuration files is described in the [Configuration Examples](#) section of this manual.

The details of these directives is explained in the [UPS Sharing](#) section of the Configuration chapter of this document.

In addition, sample master and slave configuration files can be found in the <src>/examples directory (**apcupsd.master.conf** and **apcupsd.slave.conf**).

Master/Slave Problems

If you are having problems getting a master/slave configuration to work, or you are getting error messages, please see the [Trouble Shooting Apcupsd](#) Chapter of this manual for more details.

Master/Slave Shutdown

For additional details of shutting down a master/slave configuration, please see the [Master/Slave Shutdown](#) section of the Shutdown chapter of this manual.



Apcupsd Security Issues

- **apcupsd** runs as root.
- If you have **NETSERVER ON** in your [apcupsd.conf](#) file, be aware that anyone on the network can read the status of your UPS. This may or may not pose a problem. If you don't consider this information privileged, as is the case for me, there is little risk. In addition, if you have a firewall between your servers and the Internet, hackers will not have access to your UPS information. Additionally, you can restrict who can access your **apcupsd** server by using the [INETD](#) services and using access control lists with a TCP wrapper or by configuring TCP wrappers in **apcupsd** (see below for TCP Wrapper details).
- If you are running master/slave networking with a single UPS powering multiple machines, be aware that it is possible for someone to simulate the master and send a shutdown request to your slaves. The slaves do check that the network address of the machine claiming to be the master is that same as the address returned by DNS corresponding to the name of the master as specified in your configuration file.

TCP Wrappers

As of apcupsd version 3.8.2, TCP Wrappers are implemented if you turn them on when configuring (`./configure --with-libwrap`). With this code enabled, you may control who may access your **apcupsd** via TCP connections (the Network Information Server, and the Master/Slave code). This control is done by modifying the file: `/etc/hosts.allow`. This code is implemented but untested. If you use it, please send us some feedback.



Apcupsd STATUS Logging

There is a good deal of information available about the UPS and apcupsd's status. This document describes the format of that information.

STATUS format

The STATUS output is in ASCII format with a single data value or piece of information on each line output. Because not all UPSes supply the same information, the output varies based on the type of UPS that you are using. In general, if the information is not available for your UPS, the data portion of the output record will contain an N/A indicating that the information is not available.

Status logging consists of periodically logging ALL available information concerning the UPS. Since the volume of data is rather large (over 1000 bytes per status), the STATUS data is not automatically sent to the system log file, instead, it is written as a series of data records to a specific file (normally **/etc/apcupsd/apcupsd.status**).

After each write, the file is rewound so that the size of the file remains constant. At the current time, this file is 1135 bytes. The format of this file is very similar to the old apcupsd procfs file. The STATUS file is kept for backwards compatibility and will be eliminated in a future version of **apcupsd**. The preferred method for obtaining this information is from [apcaccess status](#) or by using the **apcupsd** [network information server](#).

SmartUps

From the following models:

UPSTYPE	Descriptive Name
=====	=====
newbackupsp	Smarter BackUPS Pro
backupspn	Smarter BackUPS Pro
smartups	SmartUPS
matrixups	MatrixUPS
sharesmart	ShareUPS Advanced Port

STATUS logging

To make reading the status data reliable via a named pipe, the first record written contains a version number, the number of records that follow the first record, and the total number of bytes in those subsequent records. An actual example of such a status file (**/etc/apcupsd/apcupsd.status**) is:

Consequently, the first record always consists of 24 bytes (23 characters followed by a newline). This record starts with APC and as indicated in the example above is followed by 28 records consisting of 675 bytes. The last record begins with END APC and contains the date and time matching the DATE record.

Documentation of each record needs to be written. In the coming weeks, I plan to add additional records and possibly change the names of some of the fields.

When this data is written to a file, it is written as two records, the first record, and all the other records together. In reading the file, it can be either be read a record at a time, or in one big read.

APC UPS management under Linux

When this data is written to syslog(), it is written a record at a time. The first record is the first 24 bytes. By having the number of records and the size in the first record, the complete status can be reliably reassembled.

An example of output from an international SmartUPS 1000 follows:

```
DATE       : Wed Sep 27 17:30:23 CEST 2000
HOSTNAME   : polymatou.sibbald.com
RELEASE    : 3.7.3-20000925
CABLE      : Custom Cable Smart
MODEL      : SMART-UPS 1000
UPSMODE    : Stand Alone
STARTTIME  : Wed Sep 27 10:39:23 CEST 2000
UPSNAME    : UPS_IDEN
STATUS     : ONLINE
LINEV      : 235.3 Volts
LOADPCT    : 9.3 Percent Load Capacity
BCHARGE    : 100.0 Percent
TIMELEFT   : 130.0 Minutes
MBATTCHG   : 5 Percent
MINTIMEL   : 3 Minutes
MAXTIME    : 0 Seconds
MAXLINEV   : 239.2 Volts
MINLINEV   : 234.0 Volts
OUTPUTV    : 236.6 Volts
SENSE      : High
DWAKE      : 000 Seconds
DSHUTD     : 020 Seconds
DLOWBATT   : 02 Minutes
LOTRANS    : 196.0 Volts
HITRANS    : 253.0 Volts
RETPCT     : 000.0 Percent
ITEMP      : 32.8 C Internal
ALARMDEL   : 5 seconds
BATTV      : 27.9 Volts
LINEFREQ   : 50.0 Hz
LASTXFER   : Line voltage notch or spike
NUMXFERS   : 0
XONBATT    : N/A
TONBATT    : 0 seconds
CUMONBATT  : 0 seconds
XOFFBATT   : N/A
SELFTTEST  : NO
STESTI     : 336
STATFLAG   : 0x08 Status Flag
DIPSW      : 0x00 Dip Switch
REG1       : 0x00 Register 1
REG2       : 0x00 Register 2
REG3       : 0x00 Register 3
MANDATE    : 07/31/99
SERIALNO   : QS9931125245
BATTDATE   : 07/31/99
NOMOUTV    : 230
NOMBATTV   : 24.0
HUMIDITY   : N/A
AMBTEMP    : N/A
EXTBATTS   : 0
BADBATTS   : N/A
FIRMWARE   : 60.11.I
APCMODEL   : IWI
END APC    : Wed Sep 27 17:30:31 CEST 2000
```

The meaning of the above variables are:

APC

is the header record indicating the STATUS format revision level, the number of records that follow the APC statement, and the number of bytes that follow the record.

DATE

is the date and time that the information was last obtained from the UPS.

HOSTNAME

is the name of the machine that collected the UPS data.

RELEASE

is the **apcupsd** release number.

CABLE

is the cable as specified in the configuration file.

MODEL

is the UPS model as derived from information from the UPS.

UPSMODE

is the mode in which **apcupsd** is operating.

STARTTIME

is the time/date that **apcupsd** was started.

UPSNAME

is the name of the UPS as stored in the EEPROM.

STATUS

is the current status of the UPS (ONLINE, CHARGING, ONBATT,...)

MASTERUPD

is the last time the master sent an update to the slave. This value is present only in slave configurations.

LINEV

is the current line voltage as returned by the UPS.

LOADPCT

is the percentage of load capacity as estimated by the UPS.

BCHARGE

is the percentage charge on the batteries.

TIMELEFT

is the remaining runtime left on batteries as estimated by the UPS.

MBATTCHG

if the battery charge percentage (BCHARGE) drops below this value, **apcupsd** will shutdown your system.

MINTIMEL

apcupsd will shutdown your system if the remaining runtime equals or is below this point.

MAXTIME

apcupsd will shutdown your system if the time on batteries exceeds this value. A value of zero disables the feature.

MAXLINEV

is the maximum line voltage since the last STATUS as returned by the UPS.

MINLINEV

is the minimum line voltage since the last STATUS as returned by the UPS.

OUTPUTV

is the voltage the UPS is supplying to your equipment.

SENSE

is the sensitivity level of the UPS to line voltage fluctuations.

DWAKE

is the amount of time the UPS will wait after a power off condition when the power is restored.

DSHUTD

is the grace delay that the UPS gives after receiving a power down command from **apcupsd** before it powers off your equipment.

DLOWBATT

is the remaining runtime below which the UPS sends the low battery signal. At this point **apcupsd** will force an immediate emergency shutdown.

LOTRANS

is the line voltage below which the UPS will switch to batteries.

HITRANS

is the line voltage above which the UPS will switch to batteries.

RETPCT

is the percentage charge that the batteries must have after a power off condition before the UPS will restore power to your equipment.

STATFLAG

is a status flag indicating the UPS status. See STATUS.

ITEMP

is the internal UPS temperature as supplied by the UPS.

ALARMDEL

is the delay period for the UPS alarm.

BATTV

is the battery voltage as supplied by the UPS.

LINEFREQ

is the line frequency in Hertz as given by the UPS.

LASTXFER

is the reason for the last transfer to batteries.

NUMXFERS

the number of transfers to batteries since **apcupsd** startup.

XONBATT

time and date of last transfer to batteries, or N/A.

TONBATT

time in seconds currently on batteries, or 0.

CUMONBATT

total (cumulative) time on batteries in seconds since **apcupsd** startup.

XOFFBATT

time and date of last transfer from batteries, or N/A.

SELFTEST

is the results of the last self test, and may have the following values:

OK – self test indicates good battery

BT – self test failed due to insufficient battery capacity

NG – self test failed due to overload

NO – No results (i.e. no self test performed in the last 5 minutes).

STESTI

is the interval in hours between automatic self tests.

STATFLAG

status flag. English version is given by STATUS.

DIPSW

is the dip switch settings.

REG1

is the value from the UPS fault register 1.

REG2

is the value from the UPS fault register 2.

REG3

is the value from the UPS fault register 3.

MANDATE

is the date the UPS was manufactured.

SERIALNO

is the UPS serial number.

BATTDATE

is the date that batteries were last replaced.

NOMOUTV

is the output voltage that the UPS will attempt to supply when on battery power.

NOMBATTV

is the nominal battery voltage.

HUMIDITY

is the humidity as measured by the UPS.

AMBTEMP

is the ambient temperature as measured by the UPS.

EXTBATTs

is the number of external batteries as defined by the user. A correct number here helps the UPS compute the remaining runtime more accurately.

BADBATTs

is the number of bad battery packs.

FIRMWARE

is the firmware revision number.

APCMODEL

is the old APC model identification code.

END APC

is the time and date that the STATUS record was written.

BackUPS Pro and SmartUPS v/s Smart Signals

```
LINEFAIL : OnlineStatus
BATTSTAT : BatteryStatus
LINEVOLT : LineVoltageState
LASTEVNT : LastEventObserved
```

BackUPS and NetUPS Simple Signals

```
LINEFAIL : OnlineStatus
BATTSTAT : BatteryStatus
```

BackUPS Pro and SmartUPS v/s Smart Signals

OnlineStatus BatteryStatus LineVoltageState LastEventObserved

BackUPS and NetUPS Simple Signals

OnlineStatus BatteryStatus

Logging the STATUS Information

If specified in the configuration file, the STATUS data will also be written to the system log file. Please note, that it would not normally be wise to write this data to a normal system log file as there is no mechanism in syslog() to rewind the file and hence the log file would quickly become enormous. However, in two cases, it can be very useful to use syslog() to write this information.

The first case is to setup your syslog.conf file so that the data is written to a named pipe. In this case, normally not more than about 8192 bytes of data will be kept before it is discarded by the system.

The second case is to setup your syslog.conf file so that the status data is sent to another machine, which presumably then writes it to a named pipe. Consequently, with this mechanism, provides a simple means of networking apcupsd STATUS information.

Although we mention system logging of STATUS information, we strongly recommend that you use the **apcupsd** [network information server](#) to network this information. Also, see the [system logging](#) section of this manual.



APC's smart protocol

Credits

The APC UPS protocol was originally analyzed by Pavel Korensky with additions from Andre H. Hendrick beginning in 1995, and we want to give credit for good, hard work, where credit is due. After having said that, you will see that Steven Freed built much of the original **apcupsd** information file. [Comment inserted by Riccardo Facchetti]

The start of this chapter of the **apcupsd** manual in HTML format was pulled from the [Network UPS Tools \(NUT\)](#) site. It has been an invaluable tool in improving **apcupsd**, and I consider it the **Bible** of APC UPS programming. In the course of using it, I have added information gleaned from **apcupsd** and information graciously supplied by APC. Hopefully, the additions made herein can benefit the original author and his [programming project](#), and maybe some day, the **Apcupsd** project and the **NUT** project can join forces.

Description

Here's the information on the elusive APC smart signaling protocol used by their higher end units (Back-UPS Pro, Smart-UPS, Matrix-UPS, etc). What you see here has been collected from a variety of sources. Some people analyzed the chatter between PowerChute and their hardware. Others sent various characters to the UPS and figured out what the results meant.

RS-232 differences

Normal 9 pin serial connections have TxD on 3 and RxD on 2. APC's smart serial ports put TxD on pin 1 and RxD on pin 2. This means you go nowhere if you use a normal straight through serial cable. In fact, you might even power down the load if you plug one of those cables in. This is due to the odd routing of pins – DTR and RTS from the PC usually wind up driving the on/off line. So, when you open the port, they go high and *poof* your computer dies.

Originally this evil hack was used to connect the UPS to the PC when this page was first being built. As you can see, I cheated and neglected the ground (only 2 wires!) and it still worked. This method can be used for playing around, but for professional systems this is obviously not a viable option.

That hack didn't work out so well (damned cats), so it was retired quite awhile back. The most practical solution was to go out and BUY the DOS/Win version of PowerChute just for the black (smart) cable. I recommend doing the same thing if you actually care about this thing working properly. Of course, if you have one of the newer packages that came with PowerChute, you already have the cable you need.

Diagram for cable hackers

If you are handy with cable creation tools, check out the [940-0024C clone diagram](#). That's the black "smart" cable normally provided with APC models sold after 1996. The loopback pins on that diagram are used to keep PowerChute happy by allowing cable detection. If you use the [NUT](#) apcsmart driver, those pins don't matter.

Many thanks to Steve Draper for providing this scan.

For additional information on cables, see the [cables section](#) of this manual.

The Smart Protocol

Despite the lack of official information from APC, this table has been constructed. It's standard RS-232 serial communications at 2400 bps/8N1. Don't rush the UPS while transmitting or it may stop talking to you. This isn't a problem with the normal single character queries, but it really does matter for multi-char things like "@000". Sprinkle a few calls to `usleep()` in your code and everything will work a lot better.

The following table describes the single character **Code** or command that you can send to the UPS, its meaning, and what sort of response the UPS will provide. Typically, the response shown below is followed by a newline (`\n` in C) and a carriage return (`\r` in C). If you send the UPS a command that it does not recognize or that is not available on your UPS, it will normally respond by "NA" for not available, otherwise the response is given in the "Typical results" column.

Code	Meaning	Typical results
^A	Model string	SMART-UPS 700
^N	Turn on UPS (send twice, with > 1.5s delay between chars) Only on 3rd gen SmartUPS and Black Back-UPS Pros	n/a
^Z	Permitted EEPROM Values	A large string (254 chars) that gives the EEPROM permitted values for your model. For details see below.
A	Front panel test	Light show + "OK" (and 2s beep)
B	Battery voltage	Ranges – typical "27.87"
C	Internal temperature (degrees C)	Ranges – typical "036.0"
D	Runtime calibration – runs until battery is below 25% (35% for Matrix) This updates the 'j' values – only works at 100% battery charge Can be aborted with a second "D"	! when on battery, \$ on line
E	Automatic self test intervals	Default = 336 (336 hours = 14 days) (336=14 days, 168=7 days, ON=power on, OFF=never)
F	Line frequency, Hz	60.00 (50.0 in Europe)
G	Cause of transfer	R = unacceptable utility voltage rate of change, H = high utility voltage, L = low utility voltage, T = line voltage notch or spike, O = no transfers yet (since turnon), S = transfer due to serial port U command or activation of UPS test from front panel, NA = transfer reason still not available (read again).
K--K	Shutdown with grace period (set with 'p') – need > 1.5s between	Matrix/3rd gen SmartUPS/Black Back-UPS Pros: "OK", all others: "*"

APC UPS management under Linux

	first and second K	
L	Input line voltage	Ranges – typical "118.3" or "228.8" in Europe
M	Maximum line voltage received since last M query	Ranges – typical "118.9" or "230.1" in Europe
N	Minimum line voltage received since last N query	Ranges – typical "118.9" or "226.2" in Europe
O	Output voltage	Ranges – typical "118.3" or "228.8" in Europe
P	Power load %	Ranges – typical "011.4" depends on what you have plugged in.
Q	Status flags	Bitmapped, see below
R	Turn dumb Only on 3rd gen SmartUPS, SmartUPS v/s, BackUPS Pro	"BYE"
S	Soft shutdown after 'p' delay, return online when power returns Only works when UPS is on battery	OK
U	Simulate power failure	!! when switching to battery, then \$ when back on line
V	Old firmware revision	"GWD" or "IWI" The last character indicates the locale (Domestic, International).
W	Self test (battery), results stored in "X"	"OK"
X	Results of last self test	"OK" – good battery, "BT" – failed due to insufficient capacity, "NG" – failed due to overload, "NO" – no results available (no test performed in last 5 minutes)
Y	Enter smart mode	"SM"
Z--Z	Shutdown immediately (no delay) – need > 1.5s between first and second Z	N/A
a	Show protocol version.alert messages.valid commands (delimited by periods)	"3.1.\$%+?=#.^A^N^Z+–789<@ABCDEFGHIJKLMNOPQRSTUVWXYZ'abcef– Link–Level.alert–messages.commands
b	Firmware revision	"50.9.D" – 50 = SKU (variable length), 9 = firmware revision, D = country code (D=USA, I=International, A=Asia, J=Japan, M=Canada)
c	UPS local id	UPS_IDEN (you can program any 8 characters here)
e	Return threshold	% battery charge threshold for return (00=00%, 01=15%, 02=25%, 03=90%)
f	Battery level %	Ranges – typical "100.0" when fully charged as should normally be the case
g	Nominal battery voltage (not actual voltage – see B)	"012" or "024" or "048".
h	Measure–UPS: ambient humidity (%)	"nnn.n" – percentage

APC UPS management under Linux

i	Measure–UPS: dry contacts	10 = contact 1, 20 = 2, 40 = 3, 80 = 4
j	Estimated runtime at current load (minutes)	"0112:" (note, it is terminated with a colon)
k	Alarm delay	0(zero) = 5 second delay after fail, T = 30 second delay, L = alarm at low battery only, N = no alarm
l	Low transfer voltage	Default "103" or "208" in Europe
m	Manufacturing date	Unique within groups of UPSes (production runs)
n	Serial number	Unique for each UPS
o	Nominal Output Voltage	The Nominal Output Voltage when running on batteries. Default "115" or "230" in Europe.
p	Shutdown grace delay, seconds	Default "020" (020/180/300/600)
q	Low battery warning, minutes	Default "02"
r	Wakeup delay (time) – seconds	Default "000" (000/060/180/300)
s	Sensitivity	"H" – highest, "M" – medium, "L" – lowest, "A" – autoadjust (Matrix only)
u	Upper transfer voltage	Default "132" or "253" in Europe
t	Measure–UPS: ambient temperature (degrees C)	"nn.nn"
x	Last battery change	Eight characters. Varies typically dd/mm/yy – 31/12/99
y	Copyright notice	"(C) APCC" – only works if firmware letter (from "V") is later than O
z	Reset the EEPROM to factory settings (but not ident or batt replacement date) Not on SmartUPS v/s or BackUPS Pro	"CLEAR"
+	Capability cycle	Cycle forward through possible values (" " from UPS afterward to confirm change). Do not use this unless you know how to program your UPS EEPROM or you may damage your UPS.
–	Capability cycle	Cycle backward through possible values (" " from UPS afterward to confirm change)Do not use this unless you know how to program your UPS EEPROM or you may damage your UPS.
@nnn	Shutdown (after delay 'p') with delayed wakeup of nnn tenths of an hour (after 'r' time)	Matrix/3rd gen UPS: "OK", others "*"
0x7f (DEL key)	Abort shutdown – use to abort @, S, K--K	"OK"
~	Register #1	See below
'	Register #2	See below
7	Dip switch positions (if applicable)	See below
8	Register #3	See below

9	Line quality	"FF" acceptable, "00" unacceptable
>	Number of external battery packs attached	SmartCell models: "nnn" where nnn is how many external packs are connected Non-SmartCell units: whatever has been set with >+ and >- by the user
Matrix UPS (and possibly Symmetra) specific commands		
^	Run in bypass mode	If online, "BYP" is received as bypass mode starts If already in bypass, "INV" is received and UPS goes online "ERR" received if UPS is unable to transfer
<	Number of bad battery packs	"nnn" – count of bad packs connected to the UPS
/	Load current	"nn.nn" – true RMS load current drawn by UPS
\	Apparent load power	"nnn.nn" – output load as percentage of full rated load in VA.
^V	Output voltage selection (editable)	"A" – automatic according to input tap, "M" – 208 VAC, "I" – 240 VAC
^L	Front panel language	"E" – English, "F" – French, "G" – German, "S" – Spanish, "1" "2" "3" "4" – ?
w	Run time conservation	"NO" (disabled) or "02" "05" "08" – minutes of runtime to leave in battery (UPS shuts down "early")

Dip switch info

Bit	Switch	Option when bit=1
0	4	Low battery alarm changed from 2 to 5 mins. Autostartup disabled on SU370ci and 400
1	3	Audible alarm delayed 30 seconds
2	2	Output transfer set to 115 VAC (from 120 VAC) or to 240 VAC (from 230 VAC)
3	1	UPS desensitized – input voltage range expanded
4–7	–	Unused at this time

Status bits

This is probably the most important register of the UPS, which indicates the overall UPS status. Some common things you'll see:

- 08 = On line, battery OK
- 10 = On battery, battery OK
- 50 = On battery, battery low
- SM = Status bit is still not available (retry reading)

Bit	Hex Bit	Meaning
0	0x01	1 = Runtime calibration occurring Not reported by Smart UPS v/s and BackUPS Pro
1	0x02	1 = SmartTrim Not reported by 1st and 2nd generation SmartUPS models
2	0x04	1 = SmartBoost
3	0x08	1 = On line (this is the normal condition)
4	0x10	1 = On battery
5	0x20	1 = Overloaded output
6	0x40	1 = Battery low
7	0x80	1 = Replace battery

Alert messages

These single character messages are sent by the UPS any time there is an Alert condition. All other responses indicated above are sent by the UPS only in response to a query or action command.

Character	Description
!	Line Fail – sent when the UPS goes on–battery, repeated every 30 seconds until low battery condition reached. Sometimes occurs more than once in the first 30 seconds.
\$	Return from line fail – UPS back on line power, only sent if a ! has been sent.
%	Low battery – Sent to indicate low battery, but not on SmartUPS v/s or BackUPS Pro models
+	Return from low battery – Sent when the battery has been recharged to some level only if a % has been sent previously
?	Abnormal condition – sent for conditions such as "shutdown due to overload" or "shutdown due to low battery capacity". Also occurs within 10 minutes of turnon.

=	Return from abnormal condition – Sent when the UPS returns from an abnormal condition where ? was sent, but not a turn-on. Not implemented on SmartUPS v/s or BackUPS Pro models.
*	About to turn off – Sent when the UPS is about to switch off the load. No commands are processed after this character is sent. Not implemented on SmartUPS v/s, BackUPS Pro, or 3rd generation SmartUPS models.
#	Replace battery – Sent when the UPS detects that the battery needs to be replaced. Sent every 5 hours until a new battery test is run or the UPS is shut off. Not implemented on SmartUPS v/s or BackUPS Pro models.
&	Check alarm register for fault (Measure-UPS) – sent to signal that temp or humidity out of set limits. Also sent when one of the contact closures changes states. Sent every 2 minutes, stops when the alarm conditions are reset. Only sent for alarms enabled with I. Cause of alarm may be determined with J. Not on SmartUPS v/s or BackUPS Pro.
	Variable change in EEPROM – Sent whenever any EEPROM variable is changed. Only supported on Matrix UPS and 3rd generation SmartUPS models.

Register 1

All bits are valid on the Matrix UPS. SmartUPS models only support bits 6 and 7. Other models do not respond.

Bit	Hex Bit	Meaning
0	0x01	In wakeup mode (typically lasts < 2s)
1	0x02	In bypass mode due to internal fault – see register 2 or 3
2	0x04	Going to bypass mode due to command
3	0x08	In bypass mode due to command
4	0x10	Returning from bypass mode
5	0x20	In bypass mode due to manual bypass control
6	0x40	Ready to power load on user command

7	0x80	Ready to power load on user command or return of line power
---	------	---

Register 2

Matrix UPS models report bits 0–5. SmartUPS models only support bits 4 and 6. SmartUPS v/s and BackUPS Pro report bits 4, 6, 7. Unused bits are set to 0. Other models do not respond.

Bit	Meaning
0	Fan failure in electronics, UPS in bypass
1	Fan failure in isolation unit
2	Bypass supply failure
3	Output voltage select failure, UPS in bypass
4	DC imbalance, UPS in bypass
5	Command sent to stop bypass with no battery connected – UPS still in bypass
6	Relay fault in SmartTrim or SmartBoost
7	Bad output voltage

Register 3

All bits are valid on the Matrix UPS and 3rd generation SmartUPS models. SmartUPS v/s and BackUPS Pro models report bits 0–5. All others report 0–4. State change of bits 1,2,5,6,7 are reported asynchronously with ? and = messages.

Bit	Meaning
0	Output unpowered due to shutdown by low battery
1	Unable to transfer to battery due to overload
2	Main relay malfunction – UPS turned off
3	In sleep mode from @ (maybe others)

4	In shutdown mode from S
5	Battery charger failure
6	Bypass relay malfunction
7	Normal operating temperature exceeded

Interpretation of the Old Firmware Revision

The Old Firmware Revision is obtained with the "V" command, which gives a typical response such as "GWD" or "IWI", and can be interpreted as follows:

Old Firmware revision and model ID String for SmartUPS MatrixUPS

This is a three character string XYZ

where X == Smart-UPS or Matrix-UPS ID Code.

range 0-9 and A-P

1 == unknown

0 == Matrix 3000

5 == Matrix 5000

the rest are Smart-UPS and Smart-UPS-XL

2 == 250 3 == 400 4 == 400

6 == 600 7 == 900 8 == 1250

9 == 2000 A == 1400 B == 1000

C == 650 D == 420 E == 280

F == 450 G == 700 H == 700XL

I == 1000 J == 1000XL K == 1400

L == 1400XL M == 2200 N == 2200XL

O == 3000 P == 5000

where Y == Possible Level of Smart Features, unknown???

G == Stand Alone

T == Stand Alone

V == ???

W == Rack Mount

where Z == National Model Use Only Codes

D == Domestic 115 Volts

I == International 230 Volts

A == Asia ?? 100 Volts

J == Japan ?? 100 Volts

Interpretation of the New Firmware Revision

The New Firmware Revision is obtained with the "b" command, which give a typical response such as "50.9.D" or "60.11.I", and can be interpreted as follows:

New Firmware revision and model ID String in NN.M.L is the format

where NN == UPS ID Code.

12 == Back-UPS Pro 650

```
13 == Back-UPS Pro 1000
52 == Smart-UPS 700
60 == SmartUPS 1000
72 == Smart-UPS 1400
```

where NN now Nn has possible meanings.

```
N == Class of UPS
1n == Back-UPS Pro
5n == Smart-UPS
7n == Smart-UPS NET
```

```
n == Level of intelligence
N1 == Simple Signal, if detectable WAG(*)
N2 == Full Set of Smart Signals
N3 == Micro Subset of Smart Signals
```

where M == Possible Level of Smart Features, unknown???

```
1 == Stand Alone
8 == Rack Mount
9 == Rack Mount
```

where L == National Model Use Only Codes

```
D == Domestic      115 Volts
I == International 230 Volts
A == Asia ??       100 Volts
J == Japan ??       100 Volts
M == North America 208 Volts (Servers)
```

EEPROM Values

Upon sending a ^Z, your UPS will probably spit back approximately 254 characters something like the following (truncated here for the example):

```
#uD43132135138129uM43229234239224uA43110112114108 ....
```

It looks bizarre and ugly, but is easily parsed. The # is some kind of marker/ident character. Skip it. The rest fits this form:

- Command character – use this to select the value
- Locale – use 'b' to find out what yours is (the last character), '4' applies to all
- Number of choices – '4' means there are 4 possibilities coming up
- Choice length – '3' means they are all 3 chars long

Then it's followed by the choices, and it starts over.

Matrix-UPS models have ## between each grouping for some reason.

Here is an example broken out to be more readable:

CMD	DFO	RSP	FSZ	FVL
u	D	4	3	127 130 133 136
u	M	4	3	229 234 239 224
u	A	4	3	108 110 112 114
u	I	4	3	253 257 261 265
l	D	4	3	106 103 100 097
l	M	4	3	177 172 168 182

```

l   A   4   3   092 090 088 086
l   I   4   3   208 204 200 196
e   4   4   2   00   15   50   90
o   D   1   3   115
o   J   1   3   100
o   I   1   3   230 240 220 225
o   M   1   3   208
s   4   4   1       H       M       L       L
q   4   4   2       02   05   07   10
p   4   4   3   020 180 300 600
k   4   4   1       0       T       L       N
r   4   4   3   000 060 180 300
E   4   4   3   336 168   ON OFF

```

```

CMD == UPSlink Command.
      u = upper transfer voltage
      l = lower transfer voltage
      e = return threshold
      o = output voltage
      s = sensitivity
      p = shutdown grace delay
      q = low battery warning
      k = alarm delay
      r = wakeup delay
      E = self test interval

```

```

DFO == (4)-all-countries (D)omestic (I)nternational (A)sia (J)apan
      (M) North America - servers.
RSP == Total number possible answers returned by a given CMD.
FSZ == Max. number of field positions to be filled.
FVL == Values that are returned and legal.

```

Programming the UPS EEPROM

There are at this time a maximum of 12 different values that can be programmed into the UPS EEPROM. They are:

Item	Command	Meaning
1.	c	The UPS Id or name
2.	x	The last date the batteries were replaced
3.	u	The Upper Transfer Voltage
4.	l	The Lower Transfer Voltage
5.	e	The Return Battery Charge Percentage
6.	o	The Output Voltage when on Batteries

7.	s	The Sensitivity to Line Quality
8.	p	The Shutdown Grace Delay
9.	q	The Low Battery Warning Delay
10.	k	The Alarm Delay
11.	r	The Wakeup Delay
12.	E	The Automatic Self Test Interval

The first two cases (Ident and Batt date) are somewhat special in that you tell the UPS you want to change the value, then you supply 8 characters that are saved in the EEPROM. The last ten items are programmed by telling the UPS that you want it to cycle to the next permitted value.

In each case, you indicate to the UPS that you want to change the EEPROM by first sending the appropriate query command (e.g. "c" for the UPS ID or "u" for the Upper Transfer voltage. This command is then immediately followed by the cycle EEPROM command or "-". In the case of the UPS Id or the battery date, you follow the cycle command by the eight characters that you want to put in the EEPROM. In the case of the other ten items, there is nothing more to enter.

The UPS will respond by "OK" and approximately 5 seconds later by a vertical bar (|) to indicate that the EEPROM was changed.

Acknowledgements

The apcupsd has a rather long and tormented history. Many thanks to the guys that, with time, contributed to the general public knowledge.

Pavel Korensky <pavelk@dator3.anet.cz>,
 Andre M. Hedrick <hedrick@suse.de>,
 Christopher J. Reimer <reimer@doe.carleton.ca>,
 Kevin D. Smolkowski <kevins@trigger.oslc.org>,
 Werner Panocha <wpanocha@t-online.de>,
 Steven Freed,
[Russell Kroll](#).

26 November 1999

additions by:

[Kern Sibbald <apcupsd-devel@apcupsd.org>](#) 21 December 1999

additional credits by:

Riccardo Facchetti

08 February 2000



Apcupsd's Support for USB UPSes

General

Apcupsd version 3.9.8 or later (development version to be released as 3.10.0) provides direct support for USB UPSes on **Linux systems only**. To run **apcupsd** with a USB UPS, you need the following things:

- A USB UPS (for example APC's BackUPS 350 CS) or an IOGear Serial to USB converter.
- Apcupsd version 3.9.8 or higher
- Version 2.4.5 or later of the Linux kernel
- A a pre-built kernel containing the USB patches such as is available as an update for RedHat 7.1, or standard in RedHat 7.2 and later. Other Linux vendors provide USB ready kernels as well.
- Or Alan Cox's patch to your kernel. If you have kernel 2.4.5, you must have patch **ac12** or later. For later versions of the kernel, any **ac** patch should do. Please note that USB enabled kernels are becoming more and more common so you may not need to build your own so this option is no longer recommended (Feb 2003).

At the current time (August 2002), **apcupsd** supports USB on Linux systems only. This is because there is no standard USB programming interface and USB on the majority of machines other than Windows and Linux is currently under development and not stable.

Indirect USB Support -- Connecting a Serial port UPS to a USB port

By using a special adaptor, you can connect your serial UPS to a USB port (note, this works only if you do NOT have a USB enabled UPS). If you would like to free up your serial port and connect your existing serial port UPS to a USB port, it is possible if you have one of the later kernels. You simply get a serial to USB adapter that is supported by the kernel, plug it in and make one minor change to your **apcupsd.conf** file and away you go. Thanks to Joe Acosta for this out to me.

The device that Joe and I are using is IOgear guc232a USB 2 serial adapter. There may be other adapters that work equally well. If you know of one, please let us know.

At my site, running RedHat 7.1 with kernel 2.4.9–12, I simply changed my **/etc/apcupsd/apcupsd.conf** configuration line to be:

DEVICE /dev/ttyUSB0

Depending on whether or not you have **hotplug** working, you may need to explicitly load the kernel modules **usbserial** and

pl2303. In my case, this was not necessary.

Direct support for USB UPSes

The rest of this chapter concerns making **apcupsd** work by connecting your USB enabled UPS directly to a USB port on your Linux machine.

Please note if you have cable number 940–0128A, your UPS will be connected to your serial port as a standard serial UPS and the rest of this chapter will not apply to your case.

Getting and Building a Kernel

Please note that a number of Linux packagers are including Alan Cox's patches in their standard releases. This is true for RedHat 7.1, 7.2, and 7.3 if you have the latest kernel updates. As a consequence before getting and building your own kernel, if you are already running a 2.4.5 kernel or later, please check whether or not it already has the necessary USB updates. This can be done by creating the device files and running the USB test program as described below.

For some very brief instructions on how to get and build your kernel, see the [Kernel Configuration](#) section of this manual. More information on configuring a kernel can be found in the kernel–HOWTO do.

Making the Device Files

Once you have your kernel installed and working, you need to define the hiddev device files. This can be done by invoking the script in `<apcupsd–src>/examples/make–hiddev`, which does the following:

```
#!/bin/sh
mkdir -p /dev/usb/hid
mknod /dev/usb/hid/hiddev0 c 180 96
mknod /dev/usb/hid/hiddev1 c 180 97
mknod /dev/usb/hid/hiddev2 c 180 98
mknod /dev/usb/hid/hiddev3 c 180 99
mknod /dev/usb/hid/hiddev4 c 180 100
mknod /dev/usb/hid/hiddev5 c 180 101
mknod /dev/usb/hid/hiddev6 c 180 102
mknod /dev/usb/hid/hiddev7 c 180 103
mknod /dev/usb/hid/hiddev8 c 180 104
mknod /dev/usb/hid/hiddev9 c 180 105
mknod /dev/usb/hid/hiddev10 c 180 106
mknod /dev/usb/hid/hiddev11 c 180 107
mknod /dev/usb/hid/hiddev12 c 180 108
mknod /dev/usb/hid/hiddev13 c 180 109
mknod /dev/usb/hid/hiddev14 c 180 110
mknod /dev/usb/hid/hiddev15 c 180 111
```

Note, as of RedHat 8.0, the hiddev devices are defined when the OS is installed, except they are defined as `/dev/usb/hiddev0 – 15`. Thus you will either need to run the above script and stick with our scheme, or you can choose to use the standard RedHat definitions, but you will need to modify all the apcupsd scripts appropriately. In a later release, we will adapt better to this situation.

Installing the HIDDEV Header File

If you have built the kernel, you must put a copy of `hiddev.h` into `/usr/include`. Use the following:

```
cd /usr/src<kernel-source-directory>/
cp include/linux/hiddev.h /usr/include/linux/
```

This step should not be necessary if you have a preconfigured kernel as long as you have loaded the **kernel–headers** rpm.

Building the Test Program

Next, we recommend that you build and run the hid-ups test program. To build it enter:

```
cd <apcupsd-src>/examples
make hid-ups
```

There should be no errors.

Now assuming that everything has gone well to this point and that you have connected your USB UPS, enter:

```
./hid-ups
```

It should print a sample report of the information that it has obtained from your UPS. CAUTION! Do not run two copies of this program at the same time, or your kernel will freeze.

The report that is printed should look very similar to the report in <src>/hid-ups.rpt.

If the program reports that the device was not found ensure that all the appropriate modules are loaded as described in the [Kernel Configuration](#) section of this manual, then unplug your UPS and plug it back in. This should permit the kernel to recognize the UPS.

If **./hid-ups** does not work, then it is very unlikely that apcupsd will work. If it complains that it could not open the device, then either you have not run the script "make-hiddev" or you are not running as root. In any case, you must correct the problem before continuing. If you choose to use the alternate RedHat device naming scheme, you will need to modify the source code of hid-dev.c to have the correct device address.

Building and Installing apcupsd

If you have gotten this far successfully, the last step should go fairly easily. You need a beta version 3.9.4 or later of apcupsd. We recommend version 3.9.8. Follow the instructions in the [Installation Chapter](#) of this manual, being sure to include the following options (in addition to any others you need) on the **./configure** line:

```
./configure \
--with-serial-dev=/dev/usb/hid/hiddev[0-9] \
--with-upstype=usb \
--with-upscable=usb \
--enable-pthreads \
--enable-usb
```

Please note, it is IMPORTANT to include the **--with-serial-dev=/dev/usb/hid/hiddev[0-9]** \ line. This will cause the **apcupsd.conf** file to contain:

```
DEVICE /dev/usb/hid/hiddev[0-9]
```

The **[0-9]** is not a typo, but should be entered exactly as shown. This is because the UPS can change device numbers while it is running. Every time there is a blip or slowdown on the USB line, the kernel will invalidate the UPS connection, then a few moments later, it will reconnect but with a different device number. Not very Unix like, but that is what happens. This bizarre syntax allows **apcupsd** to try a range of devices until it finds or re-finds the UPS device.

USB Specific Information

The UPS has an internal set of timers and remaining capacity counters, which it uses to determine when to shutdown. These are in addition to the **apcupsd** counters BATTERYLEVEL and MINUTES. As a consequence, **apcupsd** will shutdown on the first limit that triggers (either an **apcupsd** limit, or a UPS limit).

The UPS internal counter equivalent to BATTERYLEVEL can be found in the **hid-ups** report as RemainingCapacityLimit, which is typically factory set to 10 percent. In addition, the Low Battery signal is normally given by the UPS when less than 2 minutes of run time remain.

If you are technically inclined, you may want to look at the /proc file system to see what devices are attached to your USB ports. The most interesting information will be found by listing the contents of **/proc/bus/usb/devices**. This information is updated by the kernel whenever a device is plugged in or unplugged, irrespective of whether **apcupsd** is running or not. To interpret the codes in this file, please see http://www.linuxhq.com/kernel/v2.4/doc/usb/proc_usb_info.txt.html

As a reference, on my system, I have the following entry for my Back-UPS 350 direct connected USB device:

```
T: Bus=01 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#= 2 Spd=1.5 MxCh= 0
D: Ver= 1.10 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=051d ProdID=0002 Rev= 1.00
S: Manufacturer=American Power Conversion
S: Product=Back-UPS 350 FW: 5.2.I USB FW: c1
S: SerialNumber=BB0115017954
C:* #Ifs= 1 Cfg#= 1 Atr=a0 MxPwr= 30mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=03(HID ) Sub=00 Prot=00 Driver=hid
E: Ad=81(I) Atr=03(Int.) MxPS= 8 Iv1= 10ms
```

And for my IOgear that runs my serial SmartUPS 1000 (plugged into a USB port):

```
T: Bus=01 Lev=01 Prnt=01 Port=01 Cnt=02 Dev#= 4 Spd=12 MxCh= 0
D: Ver= 1.10 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=0557 ProdID=2008 Rev= 0.01
C:* #Ifs= 1 Cfg#= 1 Atr=a0 MxPwr=100mA
I: If#= 0 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=serial
E: Ad=81(I) Atr=03(Int.) MxPS= 10 Iv1= 1ms
E: Ad=02(O) Atr=02(Bulk) MxPS= 64 Iv1= 0ms
E: Ad=83(I) Atr=02(Bulk) MxPS= 64 Iv1= 0ms
```

Note that the IOgear device is using the **serial** driver (the I: line) while the Back-UPS 350 is using the **hid** driver.

Here is an example of a **cat /proc/modules** on my machine (RedHat 7.1 – kernel 2.4.9–12). Note, I am running both an IOGear serial USB device and a standard USB device.

```
nfs                77312    9 (autoclean)
es1371             26816    0 (autoclean)
ac97_codec         9376     0 (autoclean) [es1371]
gameport           1856     0 (autoclean) [es1371]
soundcore          4144     4 (autoclean) [es1371]
nfsd                69920    4 (autoclean)
lockd              51664    1 (autoclean) [nfs nfsd]
sunrpc             62832    1 (autoclean) [nfs nfsd lockd]
parport_pc         14736    1 (autoclean)
```

APC UPS management under Linux

lp	6176	0 (autoclean)
parport	24832	1 (autoclean) [parport_pc lp]
autofs	10784	1 (autoclean)
nls_iso8859-1	2880	1 (autoclean)
smbfs	35344	1 (autoclean)
3c59x	26336	1 (autoclean)
ipchains	36704	0
pl2303	7648	1
hid	18576	1
usbserial	18288	1 [pl2303]
input	3648	0 [hid]
usb-uhci	21568	0 (unused)
usbcore	50784	1 [pl2303 hid usbserial usb-uhci]

I am not a kernel expert, but for a standard USB connection, I believe that you need the following modules loaded:

```
usbcore
input
hid
```

For the IOGear serial USB connection, you need:

```
usbcore
usbserial
pl2303
```

Known Problems

Lock File not Released if UPS Disconnected

If either you disconnect the UPS or it disconnects because of some electrical problem, it will most certainly reconnect with a different device number. `apcupsd` will detect this and reconnect properly. However, **apcupsd** does not release the old device (serial port) lock file and create a new one. This is not too serious.

Reinitialization If You Connect a Different UPS

If you disconnect the UPS and plug in a different one or a different model, it will continue to function properly, but in `apcupsd` version 3.9.4 the static data such as the UPS name, model, serial number, and firmware will not be updated. Versions 3.9.6 and greater detect the change and do a complete reinitialization of the UPS and so do not have this problem.

Power Off (killpower) of UPS Does Not Work

Currently (as of 3.10.5) the code to power off the UPS does not function properly. It does not look like a solution to this problem will be available until 2.5 of the kernel is released. In the mean time, the UPS will normally power itself down one to two minutes after the machine is shutdown.

Apcupsd Cannot Reconnect After a Reboot

If `apcupsd` does not connect to the USB port when you reboot, it is probably the appropriate kernel modules are not getting loaded correctly.

You can check this by bringing up your system, fiddling around until you get `apcupsd` to work with the UPS, then do:

```
cat /proc/modules
```

and save the output some place. Then reboot your computer and before you do anything else, do the `cat /proc/modules` again. Most likely you will find some of the usb modules are missing in the second listing.

There are two solutions:

- Ensure that you have the `hotplug` program loaded. It should fix the problem. This is a bit of magic, so we are not exactly sure how it works. The rpm I (Kern) have loaded is:

hotplug-2001_02_14-15

You might want to read the man page on **hotplug**, and it might be necessary to:

```
cp /etc/hotplug/usb.rc /etc/init.d/hotplug
```

to get it fully working.

- You can explicitly force the appropriate usb modules to be loaded by adding:

```
/sbin/modprobe <missing-module-name>
```

in the `/etc/rc.d/init.d/apcupsd` script just after the **start** case (at about line 17). This will force the modules to be loaded before `apcupsd` is invoked.

Normally, the modules you will need loaded are the following:

usbcore

hid

input

Discliamer

First, please rememeber this is beta software. It is not yet complete and there are sure to be some problems. We would appreciate hearing about your experiences.



Apcupsd's Support for SNMP UPSes

General

Apcupsd version 3.10.0 or later provides direct support for SNMP UPSes. To run **apcupsd** with an SNMP UPS, you need the following things:

- An SNMP UPS, for example a Web/SNMP card installed into the SmartSlot.
- Apcupsd version 3.10.0 or higher
- [NETSNMP](#) library (previously known as ucd-snmp) installed

Connecting an SNMP UPS

The Simple Network Management Protocol provides an interface to connect to remote devices through the network. **apcupsd** is now capable of using the SNMP interface of an SNMP-enabled UPS to communicate with an UPS. Currently **apcupsd** supports only APC's PowerNet MIB. To enable the SNMP support it is enough to configure the correct device in your apcupsd.conf configuration file. The directive needed for this configuration is:

```
DEVICE 192.168.100.2:161:APC:private
```

where the directive is made by four parts:

- IP address of the remote UPS
- Remote SNMP port
- Kind of remote SNMP agent, currently can only be "APC" for APC's powernet MIB
- The read-write community string, usually it is "private" for read-write access.

Building and Installing apcupsd

Follow the instructions in the [Installation Chapter](#) of this manual, being sure to include the following options (in addition to any others you need) on the **./configure** line:

```
./configure \
--with-serial-dev= \
--with-upstype=snmp \
--with-upscable=smart \
--enable-pthreads \
--enable-snmp
```

SNMP Specific Information

The SNMP connection gives less information compared to a serial smart cable. This is not a problem as the most useful information is given, together with a number of secondary parameters that are informative enough to run safely your UPS.

Known Problems

Power Off (killpower) of SNMP UPS needs special handling

Currently (as of 3.10.0) the code to power off the UPS needs special configuration. The killpower command for SNMP UPSes can not be issued during shutdown as typically at some time during shutdown operations the network stack is stopped. To overcome this problem it is needed to modify the /etc/rc.d/apcupsd system control script to tell **apcupsd** to issue the power down command (killpower) to the UPS immediately before apcupsd initiates the system shutdown. For this reason it is paramount to set your UPS grace time to a value greater than 120 seconds to allow for clean shutdown operations before the UPS removes the power from its plugs. To enable correct shutdown operation during powerdown do the following:

- Connect to your Web/SNMP card using your favorite web browser, go to the UPS configuration menu and change the "Shutdown Delay" parameter to 180 seconds or more, depending on how much time your system shutdown requires to umount all the filesystems.
- Change /etc/rc.d/apcupsd script adding the '--kill-on-powerfail' to the apcupsd invocation.
- Restart your **apcupsd**

With this setup your UPS operations should be safe.



Configuring a Kernel for USB Support

Note, this chapter is somewhat out of date as the kernel has evolved quite a bit in the last year and a half. In fact, you should not have to venture into building your own kernel unless your OS provider is seriously behind the times. With that said, there is still much valuable information here that you can use if you must build your own system.

General

Apcupsd version 3.10.5 provides support for USB UPSes on Linux systems. However at this time (February 2003) Linux kernels do not yet support the HIDDEV device that is used by **apcupsd**. If you are running a release such as RedHat 7.3 or 8.0, the kernels generally come preconfigured with all the necessary patches. If you are not so fortunate but you have a kernel version 2.4.5 or later and you apply the appropriate Alan Cox patch, you will be able to enable USB support in **apcupsd**.

For kernel 2.4.5, you need patch **ac12** or later. For later versions of the kernel, any **ac** patch should work.

Downloading

New kernel versions are released and an amazing speed, so by the time you read this, the current stable kernel will no longer be 2.4.5. You can obtain the 2.4 kernels from: <http://www.kernel.org/pub/linux/kernel/v2.4/>.

You can obtain the Allen Cox patches from: <http://www.kernel.org/pub/linux/kernel/people/alan/2.4/>

Alternatively, you can obtain the latest kernel source from your favorite vendor. For example, for RedHat, you would obtain: **kernel-source-2.4.x.rpm** and install it with **rpm**, then skip the first two steps in the next section.

Building the Kernel

I provide here only a very brief explanation of the steps necessary to build your kernel. If you have a later kernel such as RedHat 7.3 (kernel 2.4.18–5), you don't need to rebuild it. If you want to anyway, you can install the **kernel-source** rpm, cd to the appropriate directory (usually **/usr/src/linux-2.4**), then skip immediately to step 11 below.

1. Download kernel from:
`http://www.kernel.org/pub/linux/kernel/v2.4/`
 (I assume you get **linux-2.4.5.tar.gz** and that you put it into **/usr/src**)
2. Download Alan Cox patch from:
`http://www.kernel.org/pub/linux/kernel/people/alan/2.4/`
 (I assume you get **patch-2.4.5-ac12.gz** and that you put it into **/usr/src**)
3. `su root`
4. `cd /usr/src`
5. Ensure that the directory **linux** does not exist, or if it is linked, remove the link or change the

name.

6. Unpack the kernel with:

```
tar xvfz linux-2.4.5.tar.gz
```

7. Unpack the patch with:

```
gunzip patch-2.4.5-ac12.gz
```

8. Move the kernel source into a different directory:

```
mv linux linux-2.4.5
```

or

```
mv linux linux-2.4.5-ac12
```

9. cd linux-2.4.5

10. Apply the patch with:

```
patch -p1 cat /proc/modules
```

If not, load them by hand.

```
modprobe uhci
```

```
modprobe hid
```

Build Problems

If you start with the RedHat **kernel-source** package as I did and use one of their config files in **configs** as I did, you are very likely to get a number of undefined symbols when making or installing the modules. In this case, I found that I can simply edit **.config** with my favorite editor, comment out modules that don't build (knowing which ones is not always obvious from the names), then simply **make modules** and **make modules_install** until all the problems go away. Once the module build and install is correct, I recommend doing **make bzImage** and **make install** again.

Disclaimer

I'm not at all a kernel expert so you are pretty much on your own here. Any corrections to these instructions would be welcome.



The Windows Version of Apcupsd

General

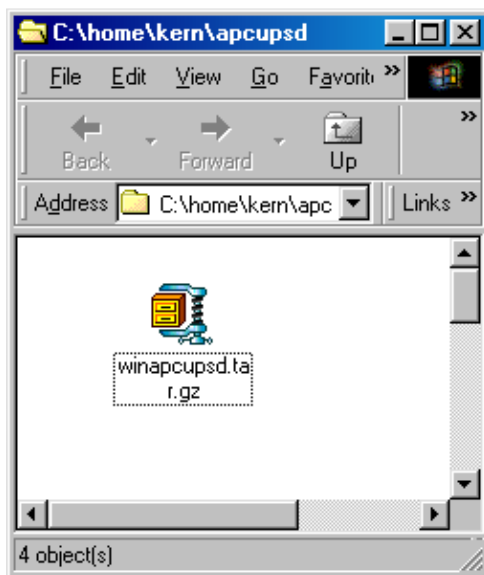
The Windows version of **apcupsd** has been tested on Win95, Win98, WinMe, WinNT, WinXP, and Win2000 systems. This version of **apcupsd** has been built to run under the CYGWIN environment, which provides many of the features of Unix on Windows systems. It also permitted a rapid port with very few source code changes, which means that the Windows version is for the most part running code that has long proved stable on Unix systems. Even though the Win32 version of **apcupsd** is a port that relies on many Unix features, it is just the same a true Windows program. When running, it is perfectly integrated with Windows and displays its icon in the system icon tray, and provides a system tray menu to obtain additional information on how **apcupsd** is running (status and events dialogue boxes). If so desired, it can also be stopped by using the system tray menu, though this should normally never be necessary.

Once installed **apcupsd** normally runs as a system service. This means that it is immediately started by the operating system when the system is booted, and runs in the background even if there is no user logged into the system.

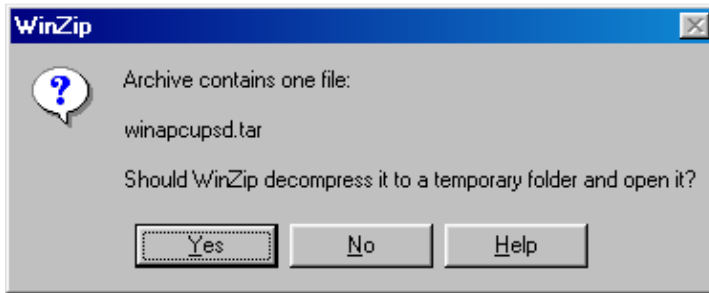
Installation

Normally, you will install the Windows version of Apcupsd from the binaries. This install is somewhat Unix like since you do many parts of the installation by hand. To install the binaries, you need **WinZip**.

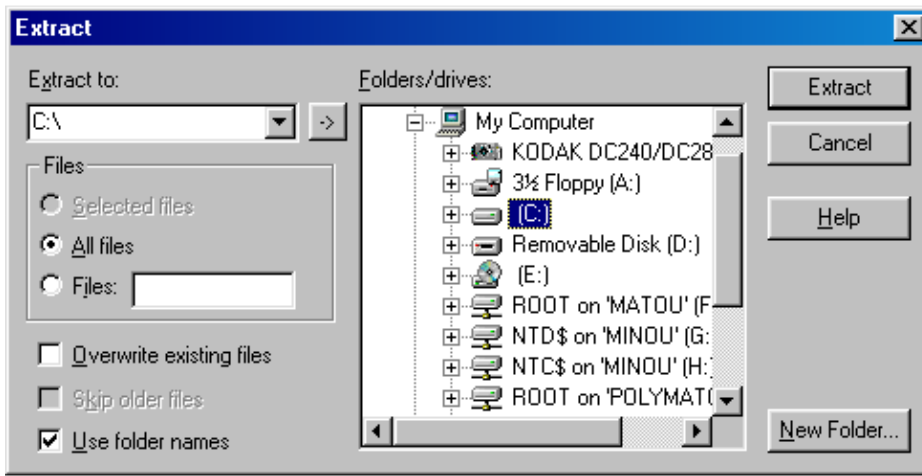
- Simply double click on the **winapcupsd-3.8.5.tar.gz** icon. The actual name of the icon will vary from one release version to another.



- When Zip says that it has one file and asks if it should unpack it into a temporary file, respond with **Yes**.



- Ensure that you extract all files and that the extraction will go into C:\



If you wish to install the package elsewhere, please note that you will need to proceed with a manual installation, which is not particularly easy as you must rebuild the source and change the configuration file as well.

This installation assumes that you do **not** have CYGWIN installed on your computer. If you do, and you use mount points, you may need to do a special manual installation.

Once you have unzipped the binaries, open a window pointing to the binary installation folder (normally **c:\apcupsd**). This folder should contain folders with the name bin, etc, examples, and manual. If and when you no longer need them, the examples and manual sub-folders of the c:\apcupsd directory may be removed.

Continuing the installation process:

- Open the directory **c:\apcupsd\etc\apcupsd** in the Windows Explorer by Clicking on the **apcupsd** folder then on the **etc** folder, then on the **apcupsd** folder. Finally double click on the file **apcupsd.conf** and edit it to contain the values appropriate for your site. In most cases, no changes will be needed, but if you are not using COM1 for your serial port, you will need to set the **DEVICE** configuration directive to the correct serial port. Note, if you are using WinNT or Win2000, the operating system may probe the port attempting to attach a serial mouse. This will cause **apcupsd** to be unable to communicate with the serial port. If this happens, or out of precaution, you can edit the **c:\boot.ini** file. Find the line that looks something like the following:

```
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Workstation Version 4.00"
```

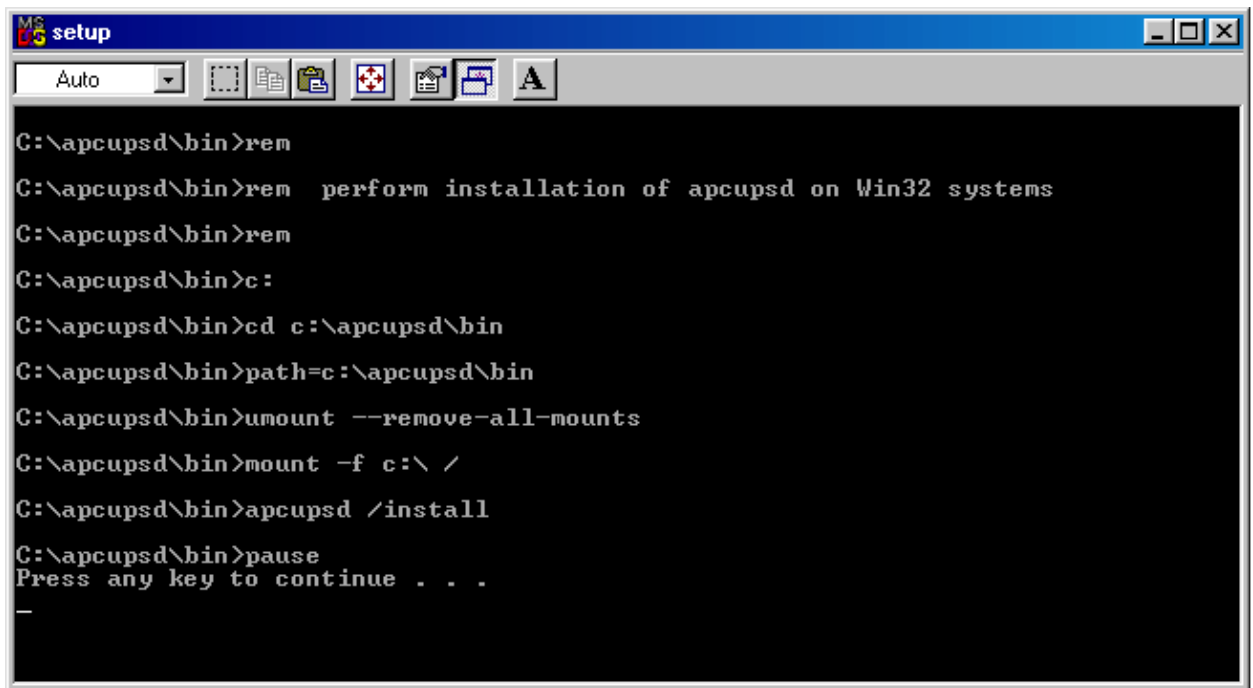
and add the following to the end of the line: /NoSerialMice:COM1 (or COM2 depending on what you want to use). The new line should look similar to:

```
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Workstation Version 4.00"  
/NoSerialMice:COM1
```

where the only thing you have changed is to append to the end of the line. This addition will prevent the operating system from interfering with **apcupsd**

- Then return to c:\apcupsd and open on the **bin** folder so that you see its contents.
- To do the final step of installation, double click on the **setup.bat** program. This script will setup the appropriate mount points for the directories that **apcupsd** uses, it will install **apcupsd** in the system registry, and on Windows 98, it will start **apcupsd** running.

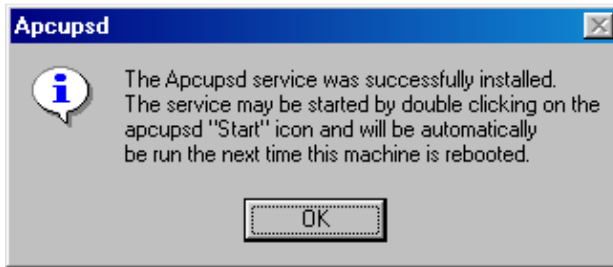
If everything went well, you will get something similar to the following output in a DOS shell window:



```
MS-DOS setup
Auto
C:\apcupsd\bin>rem
C:\apcupsd\bin>rem perform installation of apcupsd on Win32 systems
C:\apcupsd\bin>rem
C:\apcupsd\bin>c:
C:\apcupsd\bin>cd c:\apcupsd\bin
C:\apcupsd\bin>path=c:\apcupsd\bin
C:\apcupsd\bin>umount --remove-all-mounts
C:\apcupsd\bin>mount -f c:\ /
C:\apcupsd\bin>apcupsd /install
C:\apcupsd\bin>pause
Press any key to continue . . .
-
```

What is important to verify in the DOS window is that the root directory \ is mounted on device c:\.

The DOS window will be followed immediately by a Windows dialogue box as follows:

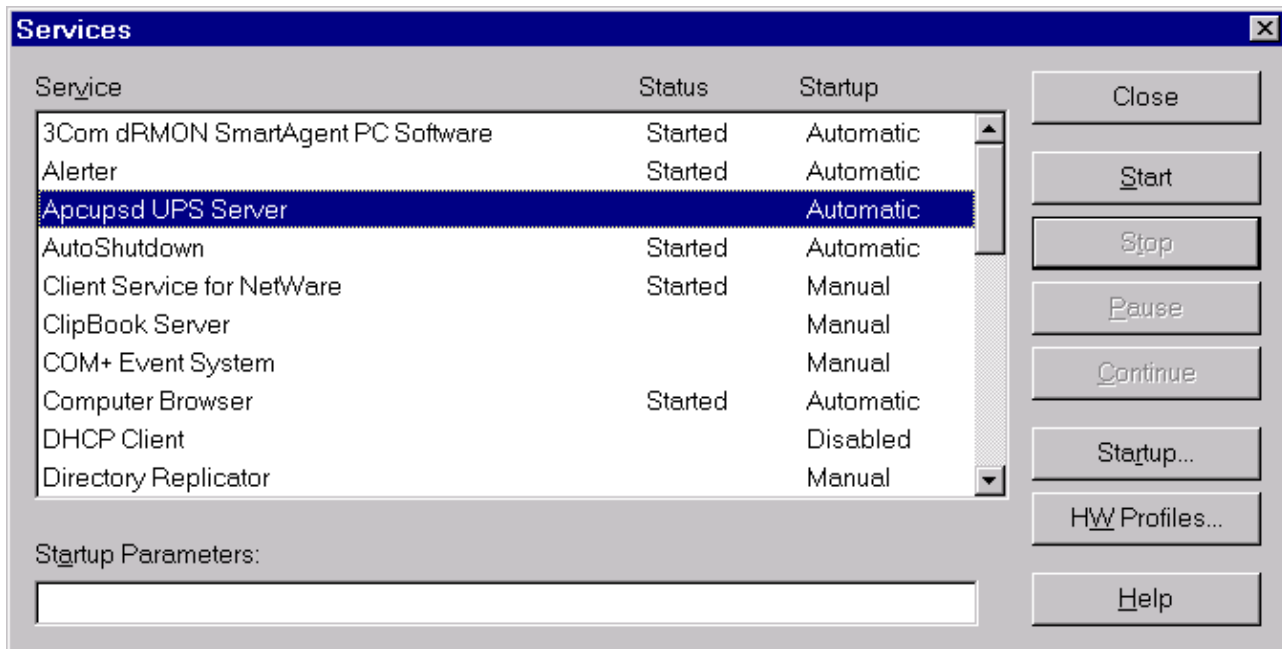




- On Windows 98, to actually start the service, either reboot the machine, which is not necessary, or open a DOS shell window, and type the following commands:

```
cd c:\apcupsd\bin
apcupsd /service
```


Alternatively, you can go to the c:\apcupsd\bin folder with the Explorer and double click on the **Start** icon.

- On Windows NT, to start the service, either reboot the machine, which is not necessary, or go to the Control Panel, open the Services folder and start the Apcupsd daemon program by selecting the Apcupsd UPS Server and then clicking on the Start button as shown below:



If the Services dialog reports a problem, it is normally because your DEVICE statement does not contain the correct serial port name. That should complete the installation process. When the system tray icon turns from a battery  into a plug , right click on it and a menu will appear. Select the **Events** item, and the Events dialogue box should appear. There should be no error messages. By right clicking again on the system tray plug and selecting the **Status** item, you can verify that all the values for your UPS are correct.

When the UPS switches to the battery, the battery icon will reappear in the system tray. While the UPS is online, if the battery is not at least 99% charged, the plug icon will become a plug with a lightning bolt in the

middle  to indicate that the battery is charging.

Installation Directory

The Win32 version of **apcupsd** must reside in the **c:\apcupsd** directory, and there must be a **c:\tmp** directory on your machine. The installation will do this automatically, and we recommend that you do not attempt to place **apcupsd** in another directory. If you do so, you are on your own, and you will need to do a rebuild of the source.

Testing

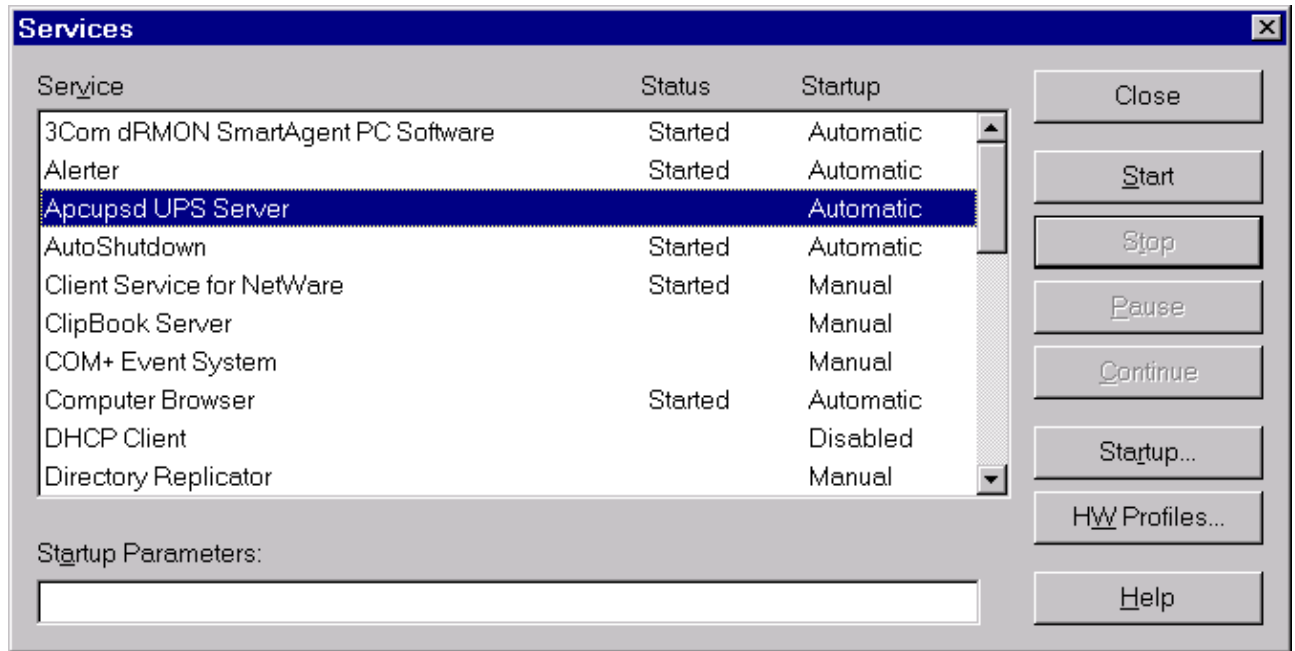
It is hard to over emphasize the need to do a full testing of your installation of **apcupsd** as there are a number of reasons why it may not behave properly in a real power failure situation.

Please read the [Testing chapter](#) of this document for general instructions on testing the Win32 version. However, on Win32 systems, there is no Unix system log file, so if something goes wrong, look in the file **c:\apcupsd\etc\apcupsd\apcupsd.events** where **apcupsd** normally logs its events, and you will generally find more detailed information on why the program is not working. The most common cause of problems is either improper configuration of the cable type, or an incorrect address for the serial port.

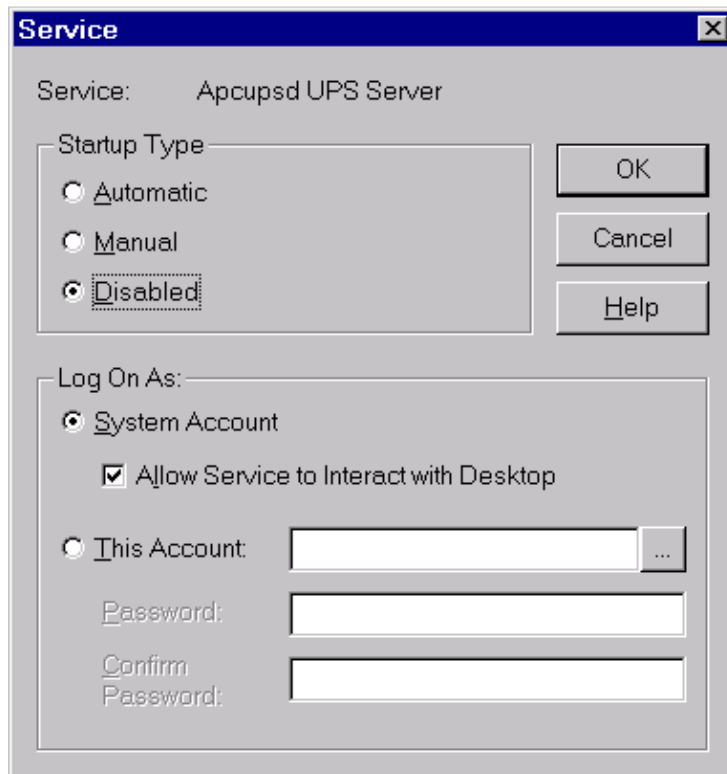
Upgrading

On Win98 and Win95 systems, to upgrade to a new release, simply stop **apcupsd** by using the tray icon and selecting the **Close Apcupsd** menu item, or by double clicking on the **Stop** icon located in the **c:\apcupsd\bin** directory, then apply the upgrade and restart **apcupsd**.

On WinNT systems (and Win2000 systems), you may stop **apcupsd** as indicated abover or alternatively you may stop **apcupsd** by using the **Services** item in the **Control Panel**. In addition, at least on my system, there seems to be a WinNT bug that causes the system to prevent **apcupsd.exe** from being overwritten even though the file is no longer being used. This is manifested by an error message when attempting load a new version and overwrite the old **apcupsd.exe** (the extract part of WinZip as described above). To circumvent this problem (if it happens to you), after shutting down the running version of **apcupsd**, through the **Services** dialogue in the **Control Panel**, first click on the **Stop button**:



then click on the **Startup ...** button, and in the Startup dialogue select the **Disabled** button to disable **apcupsd**:



After closing the dialogues, reboot the system, typical of Microsoft :-(. When the system comes back up, **apcupsd** will not be automatically launched as a service, and you can install the new version. To reinstate **apcupsd** as an automatic service, using the **Control Panel**: reset **apcupsd** to **Automatic** startup in the Startup dialogue, then restart **apcupsd** in the **Services** dialogue as shown above in the installation instructions. Frequently after an upgrade, you will click on the **Start** button and after a few seconds, the system reports that it failed to start. The cause of this problem is unknown, but the solution is simply to click again on the **Start**

button.

Post Installation

After installing **apcupsd** and before running it, you should check the contents of two files to ensure that it is configured properly for your system. The first is **c:\apcupsd\etc\apcupsd\apcupsd.conf**. You will probably need to change your UPSCABLE directive, your UPSTYPE and possibly your DEVICE directives. Please refer to the configuration section of this manual for more details.

The second file that you should examine is **c:\apcupsd\etc\apcupsd\apccontrol**. This file is called by **apcupsd** when events (power loss, etc) are generated. It permits the user to program handling the event. In particular, it permits the user to be notified of the events. For the Win32 version, each event is programmed to display a Windows popup dialogue box. If your machine is mostly unattended, you may want to comment out some of these popup dialogue boxes by putting a pound sign (#) in column one of the appropriate line.

Problem Areas

In addition to possible problems of reinstallation or upgrade on WinNT systems, as noted above, we have discovered the following problem: On some Windows systems, the domain resolution does not seem to work if you have not configured a DNS server in the Network section of the Control Panel. This problem should be apparent only when running a master or a slave configuration. In this case, when you specify the name of the master or the slave machine(s) in your **apcupsd.conf** file, **apcupsd** will be unable to resolve the name to a valid IP address. To circumvent this problem, simply enter all machine addresses as an IP address rather than a domain name, or alternatively, ensure that you have a valid DNS server configured on your system (often not the case on Win32 systems). For example, instead of using the directive:

MASTER my.master.com use something like:

MASTER 192.168.1.54 where you replace the IP address with your actual IP address.

Also, on WinNT systems, the PIF files in **/apcupsd/bin** used for starting and stopping **apcupsd** do not work. Use the services control panel instead.

On Win95 systems, there are reports that the PIF files do not work. If you find that to be the case, the simplest solution is to use the batch files that we have supplied in the **c:/apcupsd/bin** directory. Also, on Win95 systems, we have an unconfirmed report that indicates that **apcupsd** does not start automatically as a service even though the Registry has been properly updated. If you experience this problem, a work around is to put a shortcut to **apcupsd** in the StartUp folder.

As noted above, after an upgrade, you may need to start **apcupsd** several times before it will actually run.

Utility Functions

The directory **c:\apcupsd\bin** contains six utility routines (actually .pif files) that you may find useful. They are:

```
Start
Stop
Install
Uninstall
ups-events
```

`ups-status`

Any of these utilities may be used on any system, with the exception of the **Start** utility, which cannot be used on WinNT and Win2000 systems. On those systems, the **apcupsd** service must always be started through the **Services** sub-dialogue of the **Control Panel**.

The **Install** and **Uninstall** utilities install and uninstall **apcupsd** from the system registry only. All other pieces (files) of **apcupsd** remain intact. It is not absolutely necessary for **apcupsd** to be installed in the registry as it can run as a regular program. However, if it is not installed in the registry, it cannot be run as a service.

The functions of **Stop**, **ups-events**, and **ups-status** can be more easily invoked by right clicking on the **apcupsd** icon in the system tray and selecting the desired function from the popup menu.

Disclaimer

Some of the features such as EEPROM programming have not been exhaustively tested on Win32 systems. If at all possible, we recommend not to use it as a network master on Win95, Win98, and WinMe due to the instability of those operating systems.

Some items to note:

- This version of **apcupsd** will not attempt to shut off the UPS power when the battery is exhausted. Thus if the power returns before the UPS completely shuts down, your computer may not reboot automatically. This is because we do not know how to regain control after the disks have been synced in order to shut off the UPS power.

Nevertheless, it is possible to use the **--kill-on-powerfail** option on the **apcupsd** command line, but the use of this option could cause the power to be cut off while your machine is still running. See the section on [Shutdown Chapter](#) of this document for a more complete discussion of this subject. If you are still interested in trying to get this to work, please look at the code that is commented out in **c:\apcupsd\etc\apcupsd\apccontrol** under the **doshutdown** case.

An alternative to the **--kill-on-powerfail** option is to use the **KILLDELAY** configuration directive as explained in the [Configuration Section](#) of this document.

This configuration directive is appropriate on Windows machines where **apcupsd** continues to run even when the machine is halted (as is the case on most NT machines).

- When **apcupsd** detects important events, it calls **c:\apcupsd\etc\apcupsd\apccontrol**, which is a Unix shell script. You may modify this script to suit your particular needs. Currently, it puts a Windows dialogue on the screen with a brief explanation of the event. If these dialogues annoy you, you can remove or comment out the calls to **popup** from this file.

Email Notification of Events

On Win95/98 systems, it is possible to receive notification of **apcupsd** events that are passed to **apccontrol**.

This is possible using a simple email program that unfortunately is not functioning 100% correctly. In addition, I (Kern) was not able to make this program work on WinNT while **apcupsd** is running as a service under the system account (it works fine with any user account).

If you wish to try this program on Win95/98 systems, look at the files named **changeme**, **commfailure**, **commok**, **onbattery**, and **mainsback** in the directory **c:\apcupsd\examples**. To use them, you must modify the SYSADMIN variable to have a valid email address, then copy the files into the directory **c:\apcupsd\etc\apcupsd**.

Killpower under Windows

If your batteries become exhausted during a power failure and you want your machine to automatically reboot when the power comes back, it is useful to implement the **killpower** feature of the UPS where apcupsd sends the UPS the command to shut off the power. In doing so, the power will be cut to your PC and if your BIOS is properly setup, the machine will automatically reboot when the power comes back. This is important for servers.

This feature is implemented on Unix systems by first requesting a system shutdown. As a part of the shutdown, apcupsd is terminated by the system, but the shutdown process executes a script where apcupsd is recalled after the disks are synced and the machine is idle. Bacula then requests the UPS to shut off the power (killpower).

Unfortunately on Windows, there is no such shutdown script that we are aware of and no way for apcupsd to get control after the machine is idled. If this feature is important to you, it is possible to do it by telling apcupsd to immediately issue the killpower command after issuing the shutdown request. The danger in doing so is that if the machine is not sufficiently idled when the killpower takes place, the disks will need to be rescanned (and there is a possibility of lost data however small). Generally, UPSes have a shutdown grace period which gives sufficient time for the OS to shutdown before the power is cut.

To implement this feature, you need to add the **-p** option to the apcupsd command line that is executed by the system. Currently the procedure is manual. You do so by editing the registry and changing the line:

```
c:\apcupsd\apcupsd.exe /service
```

found under the key:

```
HKEY_LOCAL_MACHINE Software\Microsoft\Windows\CurrentVersion\RunServices
```

to

```
c:\apcupsd\apcupsd.exe /service -p
```

If you have a Smart UPS, you can configure the kill power grace period, and you might want to set it to 3 minutes. If you have a dumb UPS, there is no grace period and you should not use this procedure. If you have a Back-UPS CS or ES, these UPSes generally have a fixed grace period of 2 minutes, which is probably sufficient.

Power Down During Shutdown

Our philosophy is to shutdown a computer but not to power it down itself (as opposed to having the UPS cut the power as described above). That is we prefer to idle a computer but leave it running. This has the

advantage that in a power fail situation, if the killpower function described above does not work, the computer will continue to draw down the batteries and the UPS will hopefully shutoff before the power is restore thus permitting an automatic reboot.

Nevertheless some people prefer to do a full power down. To do so, you might want to get a copy of PsShutdown, which does have a power down option. You can find it and a lot more useful software at: <http://www.sysinternals.com/ntw2k/freeware/pstools.shtml>. to use their shutdown program rather than the apcupsd supplied version, you simply edit:

```
c:\apcupsd\etc\apcupsd\apccontrol
```

with any text editor and change our calls to shutdown to pssshutdown.

Command Line Options Specific to the Windows Version

These options are not normally seen or used by the user, and are documented here only for information purposes. At the current time, to change the default options, you must either manually run **apcupsd** or you must manually edit the system registry and modify the appropriate entries.

In order to avoid option clashes between the options necessary for **apcupsd** to run on Windows and the standard **apcupsd** options, all Windows specific options are signaled with a forward slash character (/), while as usual, the standard **apcupsd** options are signaled with a minus (-), or a minus minus (--). All the standard **apcupsd** options can be used on the Windows version. In addition, the following Windows only options are implemented:

<i>/servicehelper</i>	Run the service helper application
<i>/service</i>	Start apcupsd as a service
<i>/run</i>	Run the apcupsd application
<i>/install</i>	Install apcupsd as a service in the system registry
<i>/remove</i>	Uninstall apcupsd from the system registry
<i>/about</i>	Show the apcupsd about dialogue box
<i>/status</i>	Show the apcupsd status dialogue box
<i>/events</i>	Show the apcupsd events dialogue box
<i>/kill</i>	Stop any running apcupsd
<i>/help</i>	Show the apcupsd help dialogue box

It is important to note that under normal circumstances the user should never need to use these options as they are normally handled by the system automatically once **apcupsd** is installed. However, you may note these options in some of the .pif files that have been created for your use.

Building the Win32 Version from the Source

If you have the source code, follow the standard procedures for building **apcupsd** on Unix in the [Installation Section](#) of this manual. Please don't forget to look at the [Win32 specific instructions](#).

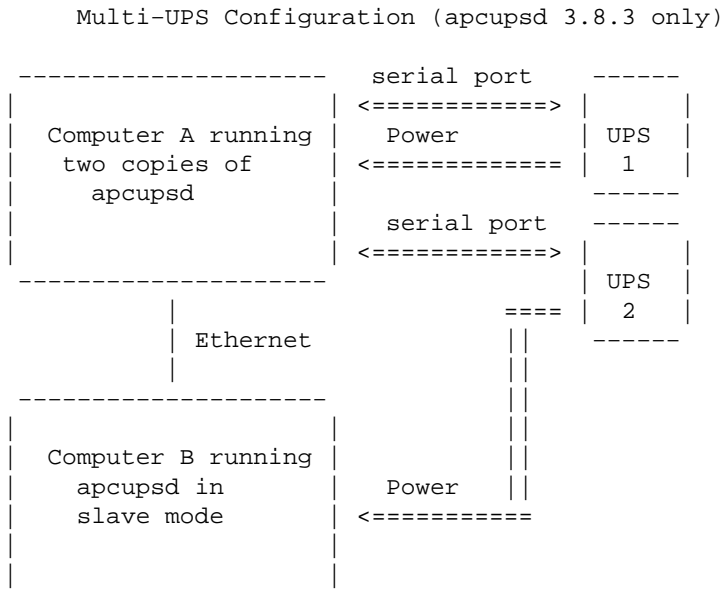


Controlling Multiple UPSes on one Machine

General

You may want to use your server to control multiple UPSes. With **apcupsd** version 3.8.3 or later, this is possible by proper configuration and by running one copy of **apcupsd** for each UPS to be controlled.

A multiple UPS configuration might be configured like the following diagram:



Configuration

The way to accomplish the above is to ensure that none of the critical files used by each of the two copies of **apcupsd** are the same. By using suitable configuration options, this is possible.

The First Copy of Apcupsd

For example, assuming you have SmartUPSes in both cases, to configure and install the first copy of **apcupsd**, which controls a UPS and Computer A, one could use the following configuration:

```

./configure \
  --prefix=/usr \
  --sbindir=/sbin \
  --with-cgi-bin=/home/http/cgi-bin \
  --enable-cgi \
  --with-css-dir=/home/http/css \
  --with-log-dir=/etc/apcupsd \
  --with-serial-dev=/dev/ttyS0 \
  --enable-pthreads \
  --with-nis-port=7000 \
  --enable-powerflute
  
```

This is pretty much a "normal" installation using many of the defaults. Once built and installed, this would control the first UPS and cause a shutdown of the system when the batteries are low. This copy of **apcupsd** will be started and stopped automatically when the system is booted and halted.

The Second Copy of Apcupsd

To configure and install the second copy of **apcupsd**, which controls the second UPS and Computer B, you could use the following configuration:

```
./configure \
--prefix=$HOME/apcupsd/bin \
--sbindir=$HOME/apcupsd/bin \
--enable-cgi \
--with-cgi-bin=$HOME/apcupsd/bin \
--with-log-dir=$HOME/apcupsd/bin \
--with-pid-dir=$HOME/apcupsd/bin \
--sysconfdir=$HOME/apcupsd/bin \
--with-lock-dir=$HOME/apcupsd/bin \
--with-pwrfail-dir=$HOME/apcupsd/bin \
--with-serial-dev=/dev/ttyS1 \
--enable-pthreads \
--with-nis-port=7001 \
--disable-install-distdir
```

Note, in this case, we use considerably more configuration options to ensure that the system files are placed in a different directory (\$HOME/apcupsd/bin). We have also selected a different serial port and a different NIS (Network Information Server) port. And finally, we have used the **--disable-install-distdir** option, which prevents **make install** from doing the final system installation (i.e. the modification of the halt script) since this was previously done.

Important Steps after Installation of the Second Copy

After the **make install** of the second copy of **apcupsd** there are a number important steps to complete. You must either remove or modify the file **\$HOME/apcupsd/bin/apccontrol**, so that it will not shutdown Computer A when the battery of UPS 2 is low. One suggestion is to copy **examples/safe.apccontrol** into **\$HOME/apcupsd/bin/apccontrol**. Alternatively, you could edit the **\$HOME/apcupsd/bin/apccontrol** and delete all statements that attempt to shutdown the machine.

Another important step is to find a way to shutdown Computer B when UPS 2's battery is low. Probably the simplest way to do this is to edit **\$HOME/apcupsd/bin/apcupsd.conf** on Computer A so that this second copy of **apcupsd** becomes a network master. Then install a standard slave configuration on Computer B.

Please remember that if UPS 1's batteries are exhausted before UPS 2's batteries, Computer B may not be properly shutdown. And at the current time, there is no simple means to make the two copies of **apcupsd** running on Computer A communicate. Thus there are certain risks in such a configuration. However, these configurations can be very useful for powering electronic equipment and such.

If Computer B is vitally important, it would probably be better to purchase a serial port card for it, or perhaps use a USB UPS. To ensure that it is properly shutdown if Computer A goes down, you could run a second copy of **apcupsd** on Computer B as a slave connected to the main copy of **apcupsd** on Computer A. Thus Computer B would be running two slaves, one driven by the master controlling UPS 1 and the other by the master controlling UPS 2, and Computer B could be shutdown by the first master that signaled it to do so.



Batteries

Battery Problems

If you have your UPS long enough, you will probably have battery problems. Below, you will find some suggestions for replacing batteries. One IMPORTANT note of caution: at least one user purchased one of the non-APC batteries noted below and found out that they would not fit into his unit. This required cutting and soldering and other very undesirable things, so be extremely careful in measuring the batteries including every millimeter of the terminal connections which can cause problems.

Although you can do a hot swap of your batteries while the computer is running (it works, I tried it), it may not be very satisfactory because the unit will not know that the batteries have been swapped and `apcupsd` will continue to show Low Battery. To correct this situation, you must do a discharge and recharge of the battery. At that point the battery should be calibrated better. As noted below, Carl has found that it takes several discharge/charges before the runtime calibration is accurate. Take care not to discharge your battery too much as it tends to shorten the battery life.

What Various People Have to Say about Batteries

Here is what John Walker has to say about APC UPS batteries:

I thought I'd pass on some information I've obtained which you'll probably eventually need. Besides, by writing it down I'll be able to find it the next time.

I started installing mine in 1995-1996. Lead-acid batteries have a finite life even if not subjected to deep discharge cycles. For the batteries used by APC, this is typically four to six years. As part of the self-test cycle, the UPS measures the voltage of the battery at full charge (which falls as the battery ages), and if it's below about 90% of the value for a new battery, it sets off the "Replace battery" alarm, which it repeats every day. [on `apcupsd` versions prior to 3.8.0, this message is sent once, on version 3.8.0, it is sent every 9 hours - KES].

You will occasionally get a false alarm. It's a good idea if you get an alarm to repeat the self-test the next day and see if the alarm goes away. If the alarm is persistent, you need to replace the batteries, which can be done without powering down the UPS or load—you just open up the battery door, take out the old batteries, and hook up the new ones.

APC makes "Replacement Battery Units" for each of the SmartUPS models, but they sell them directly only in the U.S.

It's best to wait until the low battery alarm before ordering a replacement—keeping batteries on the shelf reduces their life unless you keep them fully charged.

And André Hendrick says:

[For replacement batteries] You need to goto you your local Yamaha SeaDoo shop. There are 35 AMP Hour deep cycle marine batteries that are direct replacements. These are gel-cel and will double the runtime and/or cut your recharge time in half.

Jet Works

APC UPS management under Linux

1587 Monrovia Ave.
Newport Beach CA 9266?
Tel: +1 714 548-5259

J-W Batteries, Inc.
Tel: +1 714 548-4017

WPS 49-1200
GEL-CELL KB-35 BATTERY

For those that do not know what this means.....

I found the best battery for APCC UPS products that use In the two systems below:

SMART-UPS 3000 10.9% is running at 327W runs for 47.0 min.

Smart-UPS 1250 22.3% is running at 279W runs for 54.0 min.

APCUPSD UPS Network Monitor

Thu Jan 18 21:55:36 PST 2001

System Model Status Battery Chg Utility UPS Load UPS Temp Batt. Run Time Data

Linux ATA Development SMART-UPS 3000 ONLINE

100.0 % 120.2 VAC 10.9 % 36.9 C 47.0 min. All data

Linux ATA Development II APC Smart-UPS 1250 ONLINE

100.0 % 119.6 VAC 22.3 % 45.9 C 54.0 min. All data

Look at the numbers and see that these batteries are better and have more total running energy than standard ones.

SMART-UPS 3000 10.9% is running at 327W runs for 47.0 min.

Smart-UPS 1250 22.3% is running at 279W runs for 54.0 min.

APCUPSD UPS Network Monitor

Thu Jan 18 22:00:45 PST 2001

System Model Status Battery Chg Utility UPS Load UPS Temp Batt. Run Time Data

Linux ATA Development SMART-UPS 3000 ONLINE

100.0 % 120.2 VAC 19.2 % 36.9 C 27.0 min. All data

Linux ATA Development II APC Smart-UPS 1250 ONLINE

100.0 % 119.6 VAC 21.8 % 45.9 C 55.0 min. All data

SMART-UPS 3000 19.2% is running at 576W runs for 27.0 min.

Smart-UPS 1250 21.8% is running at 273W runs for 55.0 min.

Smart-UPS 1250 46.1% is running at 576W runs for 26.0 min.

Kind of cool.

The 1250 can outrun the 3000 by a factor of two under identical percentages, or run head to head for the same time.

SMART-UPS 3000 is a 48V based or 4 batteries
Smart-UPS 1250 is a 24V based or 2 batteries
Cheers,

Andre Hedrick
Linux ATA Development

Finally, here is what Carl Erhorn has to say about batteries:

Hi, Folks.

Well, Kern was absolutely right. The problem with my UPS was batteries. It was unexpected though, because there was no indication of a bad battery right up until the UPS failed entirely.

For those who might encounter the same thing, and don't know what's happening (I didn't either), here's what happened.

A week or so ago, I turned on one of my SmartUPS 700-NET models. The load is a small dual P-III unix server (Solaris 8, X86) and a 4MM tape drive. During the normal selftest that runs when you first turn on any APC UPS, the UPS 'freaked out'. The alarm stuttered at about 4 or 5 beeps per second, and all the panel lights flashed spasmodically, as if something was loose inside the UPS.

I turned off the UPS and it's load, then turned the UPS on again. This time, everything seemed fine. I booted the system that was attached, and there were no problems. The status monitor showed 9 minutes runtime (which indicates fairly low capacity), but the batteries showed fully charged. I began to suspect a bad inverter in the UPS.

However, Kern told me that he suspected the batteries. So I took the UPS offline, put an old SU-600 in it's place (just barely big enough to handle the startup peaks – I get an 'overload' lamp lit for about 2 seconds during boot), and checked out the batteries. They did indicate that they were near the end of life, so I ordered a replacement set. Those came in on Friday, and after the initial charge, a complete charge/discharge cycle to recalibrate the UPS, and some testing, I put it back in service.

Surprise! (Or maybe not?) Kern was right – there is nothing wrong with the inverter or the charging circuit, and the new cells fixed everything.

What confused me is that there was no 'replace battery' indication from the UPS, even when it failed, plus a fair amount of runtime indicated with a full charge. So if you see such behavior on one of your UPS models, it makes sense to replace the batteries, even if there is no indication that the batteries have failed yet.

One of the things I learned during this process is that the UPS internal calibration will lose accuracy over the life of the battery. I always do a recalibrate when I install new cells, but rarely do it after that, as it's time-consuming, and you really can't use the system attached to the UPS while doing it. Since my systems are almost constantly in use, it's a pain to schedule a recal, and I tend to put it off. This time it bit me. I'd suggest that folks do a recal at least once every six months. It will make your runtime estimates much more accurate, and also allows you to keep track of the state of your batteries.

For those who don't know how to do this, here's what you do. This procedure should not be confused with the 'Recalibrate' feature in the APC PowerchutePlus software. They do not do the same thing.

>From APC's web site:

Perform a Runtime Calibration. This is a manual procedure and should not be confused with the runtime calibration performed through PowerChute plus. The batteries inside of the Smart-UPS are controlled by a microprocessor within the UPS. Sometimes it is necessary to reset this microprocessor, especially after the installation of new batteries. Stop the PowerChute plus software from running and disconnect the serial cable. There must be at least a 30% load attached to the UPS during this procedure, but the process will cause the UPS to shut off and cut power to its outlets. Therefore, attach a non-critical load to the UPS and then force the UPS on battery by disconnecting it from utility power. Allow the unit to run on battery until it turns off completely. Make sure a 30% load is present! Plug the UPS back into the wall outlet and allow it to recharge (it will recharge more quickly turned off and with no load present). Once the unit has recharged, the "runtime remaining" calculation should be more accurate. Remember that if the unit is an older model, then the runtime will not improve significantly.

Background:

An APC Smart-UPS has a microprocessor which calculates runtime primarily based on the load attached to the UPS and on its battery capacity. On the right side of the front display panel there is a vertical graph of five LEDs. Each LED is an indication of battery charge in increments of twenty percent: 20, 40, 60, 80, 100% (bottom to top). For example, if the battery charge is 99%, then only four of the five LEDs are illuminated.

To ensure that an operating system receives a graceful shutdown when using PowerChute plus or a SmartSlot accessory, an alert is generated by the Smart-UPS indicating that the UPS has reached a low battery condition. The alert is audible (rapid beeping), visual (flashing battery LED or LEDs), and readable through the graphical interface of PowerChute plus software (or a native UPS shutdown program within a particular operating system.) In order to calculate this "low battery condition," all Smart-UPS products have a preconfigured low battery signal warning time of two minutes (this is the factory default setting). There are a total of four user-changeable settings: 2, 5, 7, or 10 minutes. If the low battery signal warning time is set for 2 minutes, then the alerts will activate simultaneously two minutes prior to shutdown. Similarly, if the total runtime for a particular UPS is 30 minutes with a low battery signal warning time set at 10 minutes, then the UPS will run on battery for 20 minutes before the low battery alert begins.

Total runtime is primarily based on two factors, battery capacity and UPS load. UPS load and runtime on battery are inversely proportional: as load increases, battery runtime decreases and vice versa. When utility power is lost, the UPS begins discharging the battery in order to support the attached load. Once power returns, the Smart-UPS will automatically begin to recharge its battery.

My comments on this procedure:

I believe this procedure works for all APC models that calculate runtime, not just the SmartUPS. It's important that you load the UPS to 30% of the UPS capacity, as reported by apcupsd or another UPS monitor program. I've found that normal house lamps of different

wattages allow me to adjust the load to almost exactly what I want, which is between 30% and 35% of the UPS capacity. This is critical to getting an accurate reading (according to the APC web documents). Always bring the UPS to 100% charge first, as indicated by the front panel lamps, or your UPS monitoring software.

Set the UPS shutdown time to 2 minutes, all other settings to nominal, and disconnect the serial port cable from the UPS before running the recalibration. If you leave a monitoring program running through the serial port, it will turn the UPS off early, and you don't want to do that during a recalibration run. When the run is complete, and the UPS turns off, you can reattach the serial cable, and the normal loads, and recharge the batteries normally. If you think you might have a power outage during the recharge time, allow the UPS to recharge to 20% or so (indicated by the panel lamps) before trying to use the computer system. This will allow the UPS to handle short dropouts while it recharges. Of course, if you can leave the computer off during the recharge time, the UPS will recharge much faster.

As an aside, when the batteries failed, my total runtime at 100% charge and an idle state was 9 minutes, which is pretty bad. I replaced the batteries with extended capacity cells, which add about 15% to the stock capacity. Now, after two complete charge/discharge cycles, 100% charge shows the available runtime to be 42 minutes on the system when it's idle, and 33 minutes when the system is very busy. The differences are due to the load of the computer, when the disks are busy, and the cpus are not in a halted state (my system halts the cpus when they are idle, to save power and lower heat, as do other OS like Linux), when compared to an idle state. Apcupsd indicates the load is about 27% when idle, and as much as 37% when heavily loaded.

I've found that two charge/discharge cycles result in a more accurate recalibration when installing new cells. It appears that some batteries need to be put through a couple of complete cycles before they reach their full capacity. I've also noticed that the full-charge voltage is different for each battery until they have been through two cycles. On the initial charge of my new batteries, the 100% charge voltage on the two cells was almost .5 VDC apart. After two complete cycles, the batteries measure within .01 VDC of each other!

I hope this information helps anyone who might encounter the problem I saw, and also shows folks how to recal their batteries. If you haven't done a complete recalibration in a year or two, I'd recommend it, so that you have warning of a low battery instead of what happened to me.

Regards,
—Carl

Where Carl Suggests to get Batteries

Hi, Folks.

I'm just replacing the batteries in one of my SmartUPS models, and it occurs to me that some of you may not know about the place I get them from. I have no relationship with this company, other than as a customer, but I feel they know what they are doing, their prices are fair, and they have some interesting batteries available that you can't obtain from APC.

These are the reasons I use them, and I thought this information might be useful to the US list

members. They will ship outside of the US. If you have questions, you can contact them through the email address listed on their web pages. They have always responded pretty quickly to my questions.

The company is called Battery Wholesale Distributors, and they are located in Georgetown, Texas. If you have questions, you can reach them by phone at (800) 365-8444, 9:00AM to 5:00PM (their local time), Monday through Friday. I've gotten email from them on the weekends, although the office is not open then.

I won't post prices, as you can get current pricing from their web site. They have an entire section dedicated to APC replacement batteries, and it's easy to find what you need. You can order over the web, or by phone. They accept all the usual credit cards.

The web site (as you might guess) is: www.batterywholesale.com

The thing I really like is that they have found manufacturers who make batteries in the standard case sizes, but have additional capacity over the original batteries shipped with the APC UPS models. Often, the difference is as much as 15% or so, and this can result in additional runtime. It's a nice upgrade for a minor increase in price.

They are also 'green-aware', in that they encourage you to recycle your old batteries, and will accept the old batteries back from you if you cannot find a local place that recycles them. You pay the shipping, but I think other than that, there is no charge. I've never done this, as I have a battery retailer just down the street who will accept my old batteries.

Anyway, if you didn't know about these folks, put the info aside where you can find it when you need replacement batteries. I won't make any guarantees, but I've been very pleased with their products, service, and pricing. I hope you find them as helpful to you as I do. I've been dealing with them since about 1994, and have never been disappointed. The owner of the place also is very good on technical issues, so if you have questions on their products, he can get as technical as you need to go.

Regards,
--Carl



APC UPS management under Linux

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

APC UPS management under Linux

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program" below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based

APC UPS management under Linux

on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not

signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

APC UPS management under Linux

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) 19yy <name of author>
```

```
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
```

APC UPS management under Linux

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
"Gnomovision" (which makes passes at compilers) written by James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.





APCUPSD success is due to the many people that helped in development, testing and in many other ways.

Thank all the developers that worked hard to make APCUPSD one of the best piece of software for UPS management.

Project Starter and Former Code Maintainer:

Andre Hedrick (andre@linux-ide.org)

Current Code Maintainer and Project Manager:

Riccardo Facchetti (riccardo@master.oasi.gpa.it)

Serial Communications:

Andre Hedrick (andre@linux-ide.org)

Alpha Port:

Kern Sibbald (kern at sibbald dot com)

João Rochate (jrochate@ualg.pt) testing and machine loan

Caldera:

John Pinner (john@clocksoft.com)

HP-UX Port:

Carl Erhorn (Carl_Erhorn@hyperion.com)

Robert K Nelson (rnelson@airflowsciences.com)

SOLARIS Port:

Carl Erhorn (Carl_Erhorn@hyperion.com)

OpenBSD Port:

Devin Reade (gdr@gno.org)

NetBSD Port:

Neil Darlow (neil@darlow.co.uk)

Win32 Port:

Kern Sibbald (kern at sibbald dot com)

Paul Z. Stagner (paul.stagner@charterco.com) testing

WEB Interfaces:

Kern Sibbald (kern at sibbald dot com)

Joseph Acosta (joeja@mindspring.com)

Apcupsd Support and Knowledge Base:

Brian Schau (Brian.Schau@compaq.com)

Hard Core Coders:

Riccardo Facchetti (riccardo@master.oasi.gpa.it)

Kern Sibbald (kern at sibbald dot com)

Part Time Coders:

Jonathan H N Chin (jc254@newton.cam.ac.uk)

Andre Hedrick (andre@linux-ide.org)

Brian Schau (Brian.Schau@compaq.com)

APC UPS management under Linux

Carl Erhorn (Carl_Erhorn@hyperion.com)

Distributions Mantainers:

Alpha	Kern Sibbald (kern at sibbald dot com) temp
Debian	Leon Breedt (ljb@debian.org)
FreeBSD/BSDi	Jeff Palmer (scorpio@drkshdw.org)
NetBSD	Neil Darlow (neil@darlow.co.uk)
HP-UX	Carl Erhorn (Carl_Erhorn@hyperion.com), Robert K Nelson (rnelson@airflowsciences.com)
OpenBSD	Devin Reade (gdr@gno.org)
RedHat	Kern Sibbald (kern at sibbald dot com)
Slackware	Devin Reade (gdr@gno.org)
Sparc Solaris	Carl Erhorn (Carl_Erhorn@hyperion.com)
SuSE	Riccardo Facchetti (riccardo@master.oasi.gpa.it)
Win32	Kern Sibbald (kern at sibbald dot com)

Project Discussions:

APCUPSD Mailing List: apcupsd-devel@apcupsd.org

Thanks to American Power Conversion (APC) who helped in giving technical information on their UPSes.

A special thanks to APC who gave me (Riccardo) a Smart UPS 1400 INET when my old Back UPS v/s 650's battery died. Thank you guys, your help has been invaluable.

Thanks to all the users that send bug reports and suggestions: we need your help.

Thanks to every one I forgot here. If you feel I have forgot your name, please don't hesitate to tell me.

Other Credits

Miquel van Smoorenburg

The Doctor What

Pavel Korensky

Russell Kroll <rkroll@exploits.org> for the CGI programs

Jonathan Benson <jbenson@technologist.com> for adapting the upsstatus.cgi program to work with **apcupsd**

The gd 1.2 Image Library used in our CGI programs is copyright 1994, 1995, Quest Protein Database Center, Cold Spring Harbor Labs. Permission granted to copy and distribute this work provided that this notice remains intact. Credit for the library must be given to the Quest Protein Database Center, Cold Spring Harbor Labs, in all derived works. This does not affect your ownership of the derived work itself, and the intent is to assure proper credit for Quest, not to interfere with your use of gd.

gd 1.2 was written by Thomas Boutell and is currently distributed by boutell.com, Inc.

Parts of the VNC project by ATT (cool code) were used as templates for our Win32 code, see:

<http://www.uk.research.att.com/vnc>

License

Apcupsd is generally licensed under the GNU GPL License version 2. Certain files may carry different licenses (e.g. LGPL). Please see the [License Chapter](#) of this manual for additional details.

AUTHOR

André M. Hedrick

Retired Co–Author

Christopher J. Reimer

Contributors and Testers for Version 3.7.0

André M. Hedrick <andre@suse.com>
Brian Schau <Brian.Schau@Digital.com>
Riccardo Facchetti <riccardo@master.oasi.gpa.it>
Carl Erhorn <Carl_Erhorn@hyperion.com> (Solaris Port)
Petr Soucek <petr@ryston.cz>
Jonathan H N Chin <jc254@newton.cam.ac.uk>
Neil Darlow <neil@darlow.co.uk>
Michael Perscheid <michael@perscheid.de>
Kaspar Klingholz <kp@balu.klingholz.de>
Chris Adams <cmadams@hiwaay.net>
Vladimir Ivaschenko <hazard.bsn@hazard.maks.net>
Thorsten Ziegler <ziegler@schlund.de>
Matt Mozur <matt.mozur@flashcom.net>
Thomas Porter <txporter@mindspring.com>
David W. Wormuth <dwormuth@post.harvard.edu>
Robert K. Nelson <rnelson@airflowsciences.com>
Tom Schroll <storm@liststorm.com>
John McSwain <jmcswain@InfoAve.Net>
Brian Daniels <briandaniels@mindspring.com>
Marcus Redivo <mredivo@binarytool.com>
Al Tuttle (Binkster) <altuttle@home.com>
Christian Moeller <moeller@curbysoft.dk>
Kern Sibbald <kern at sibbald dot com>

The Brave Unnamed PATCH–WORKS and TESTERS on Previous versions

Daniel Quinlan
Tom Kunicki
Karsten Wiborg <4wiborg@informatik.uni-hamburg.de>
Jean–Michel Rouet
Chris Adams
Jason Orendorf
Neil McAllister
Werner Panocha
Lee Maisel
Riccardo Facchetti
Brian Schau

The Information HELPERS and TESTERS.

Eric S. Raymond
Chris Hanson
Pavel Alex
Theo Van Dinter
Thomas Porte
Alan Davis
Oliver Hvrmann
Scott Horton
Matt Hyne
Chen Shiyuan

Other Credits

Miquel van Smoorenburg
The Doctor What
Pavel Korensky
Russell Kroll <rkroll@exploits.org> for the CGI programs
Jonathan Benson <jbenson@technologist.com> for adapting the upsstatus.cgi program to work with **apcupsd**

gd 1.2 Image Library used in our CGI programs

gd 1.2 is copyright 1994, 1995, Quest Protein Database Center, Cold Spring Harbor Labs. Permission granted to copy and distribute this work provided that this notice remains intact. Credit for the library must be given to the Quest Protein Database Center, Cold Spring Harbor Labs, in all derived works. This does not affect your ownership of the derived work itself, and the intent is to assure proper credit for Quest, not to interfere with your use of gd.

gd 1.2 was written by Thomas Boutell and is currently distributed by boutell.com, Inc.

Parts of the VNC project by ATT (cool code) were used as templates for our Win32 code, see:
<http://www.uk.research.att.com/vnc>

Bugs and Limitations

All the network modes are not supported yet. There are possible bugs in all ShareUPS mode types. If anyone has had success at all with any ShareUPS models, please report it.

EtherUPS/NetUPS

This is fully functional as of version 3.4.0.

Disclaimer NO WARRANTY

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND

PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Riccardo



Features in Previous Versions of Apcupsd

- [New features in Apcupsd 3.9.8](#)
 - [New features in Apcupsd 3.8.5](#)
 - [New features in Apcupsd 3.8.4](#)
 - [New features in Apcupsd 3.8.3](#)
 - [New features in Apcupsd 3.8.2](#)
 - [New features in Apcupsd 3.8.1](#)
 - [New features in Apcupsd 3.8.0](#)
 - [New features in Apcupsd 3.7.2](#)
 - [Upgrading to Apcupsd 3.7.2](#)
-



New Features in Apcupsd 3.9.8

- ```

----> Release apcupsd-3.9.8 (27 January 2002)
- New drivers layer adds new keywords to the UPSTYPE configuration
 variable (dumb, apcsmart, usb, snmp, net, and test). NOTE! only the
 first three are currently functional. All the old UPSTYPE keywords
 are translated into dumb or apcsmart for backward compatibility.
 NOTE! snmp, net, and test drivers are experimental and thus
 largely non-functional.
- Reorganized developers documentation inside an html manual
 (Riccardo).
- Support SCO OpenServer.
- Includes all enhancements to the officially released versions through
 3.8.5.
- Make drivers block a reasonable amount of
 time (maximum 1 minute, minimum 1 second). Blocking is determined
 by the state and functions requested.
- Many enhancements to the USB driver to support SmartUPSes (mostly).
- New code by Kamal to handle and truncate the events file.
- New smtp program to permit email messages from WinNT/Win2000 machines.
- Updates to apcaccess.
- New Dmsg() debug message code implemented.
- Include apctest Runtime Calibration feature.

----> Release apcupsd-3.9.4 Aug 13 2001
- Improvements in master/slave timeout detection and error messages.
- New master connect/disconnect events (accontrol).
- Additional master/slave documentation.
- Document USB.
- Updated manual to include ideas/suggestions received
 from users installing 3.8.2.
- Fixed safe.apccontrol to be a bit more system independent
 (thanks to Steven Orr for the idea).
- Added safe.apccontrol.in so that it is configured properly.
- Added a big warning message to the end of the ./configure
 output if /usr/uch is on your path.
- Fixed reference to lock file in Solaris apcupsd.in so it is
 properly configured.
- Removed unused variable from Solaris apcupsd.in
- Added Makefile to examples directory
- Modified main Makefile.in to do clean and distclean of examples
- Additional documentation.
- Removed the sleep 2d from the halt script.
- Install apcupsd.conf with 644 permissions
- Add --disable-install-distdir ./configure option.
 This allows easier installation of slaves or
 multiple copies of apcupsd.
- Used makedepend if not gcc (Solaris fix).
- Set SO_REUSEADDR in slave machines.
- Add 940-0020C cable support (same as 20B).
- UPSNAME now sets upsname if given. Otherwise,
 apcupsd attempts to get name from UPS, if not found,
 uses hostname, finally "default".
- Implement CommLost NIS status.
- Implement Shutdown NIS status.
- Implement Slave NIS status.
- Correct SmartTrim and SmartBoost detection code.
- ./configure prints name of shutdown program.
- Add port to hosts.conf.

```

## APC UPS management under Linux

- Add new apccontrol arguments to script file.
- Add UPSNAME to RedHat apccontrol.
- Manual updates.
- Eliminated all N/A fields in STATUS report.
- Always construct STATFLAG in STATUS report.
- Added generic symlink installation in distributions/
- Cleanups of main Makefile.in

**apcupsd** is mainly developed under Linux and will compile cleanly and work under most flavors of Unix as well as many other operating systems including Windows.

What to do if you find bugs :

send an email to [apcupsd-devel@apcupsd.org](mailto:apcupsd-devel@apcupsd.org) (Developers mailing list) or go to one of the following sites:

<http://www.apcupsd.org>

<http://www.sibbald.com/apcupsd>

Please be sure to include the version of **apcupsd** you are running, your operating system, and a detailed description of your problem.

Change Log

----> Release apcupsd-3.9.8 (xx January 2002)



## New Features in Apcupsd 3.8.5

This version of **apcupsd** corrects a bug that could cause a master/slave process to die due to a TCP/IP disconnect error.

This version also includes the ability with the **apctest** program to perform a Battery Runtime Calibration. For more information on using **apctest**, please see the [apctest Chapter](#) of this manual.

**apcupsd** is mainly developed under Linux and will compile cleanly and work under most flavors of Unix as well as many other operating systems including Windows.

What to do if you find bugs :

send an email to [apcupsd-devel@apcupsd.org](mailto:apcupsd-devel@apcupsd.org) (Developers mailing list) or go to one of the following sites:

<http://www.apcupsd.org>

<http://www.sibbald.com/apcupsd>

Please be sure to include the version of **apcupsd** you are running, your operating system, and a detailed description of your problem.

### Change Log

```
----> Release apcupsd-3.8.5 (4 January 2002)
- Add Battery Runtime Calibration to apctest.
- apctest is now built and installed by default.
- Fixed crash if TCP/IP connection broken.
- Quote DISTVER in case it contains spaces.
- Fix segmentation fault if kill power invoked by hand in
 a non-powerfail condition.
```



## New Features in Apcupsd 3.8.4

There are no new features other than two additional cables that are supported. This is a bug fix release for 3.8.3.

The software is completely developed under Linux and will compile cleanly and will work under Linux as well as many other operating systems and flavors of Linux.

What to do if you find bugs :

send an email to [apcupsd-devel@apcupsd.org](mailto:apcupsd-devel@apcupsd.org) (Developers mailing list) or go to one of the following sites:

<http://www.apcupsd.org>

<http://www.sibbald.com/apcupsd>

### Change Log

```
----> Release apcupsd-3.8.4 (14 Dec 2001)
- Fix multimoncss.c to use the Normal class in Temp and
 Humidity columns.
- Add support for 0127A and 0128A cables.
- Fix bad placement of subsys lock file on
 RedHat, HP, SuSE, Caldera, Slackware, and unknown
 systems.
- Added #include to multimon to prevent compiler
 warnings.
```

---



## New Features in Apcupsd 3.8.3

### · Highlights of this release:

- . The ./configure program prints a warning message if /usr/ucb is on your path. This for Solaris systems where the wrong shutdown program could be used.
- . Corrected serial port lock file location on Solaris.
- . Added an optional port specification to the hosts.conf file for configuring multimon.cgi
- . Added the --disable-install-distdir configure option to allow clean installation of two or more apcupsds on the same system.

The software is completely developed under Linux and will compile cleanly and will work under Linux as well as many other operating systems and flavors of Linux.

What to do if you find bugs :

send an email to [apcupsd-devel@apcupsd.org](mailto:apcupsd-devel@apcupsd.org) (Developers mailing list) or go to one of the following sites:

<http://www.apcupsd.org>

<http://www.sibbald.com/apcupsd>

### Change Log

```

----> Release apcupsd-3.8.3 (26 Nov 2001)
- Updated manual to include ideas/suggestions received
 from users installing 3.8.2.
- Fixed safe.apccontrol to be a bit more system independent
 (thanks to Steven Orr for the idea).
- Added safe.apccontrol.in so that it is configured properly.
- Fixed all apccontrol.sh.in so that it restart the computer if shutdown
 was cancelled. Fixed also the lock directory location in rc scripts.
- Removed infinite sleep from shutdown sequences.
- Added a big warning message to the end of the ./configure
 output if /usr/ucb is on your path (for Solaris shutdown).
- Updated Solaris configuration with specific config values.
- Fixed reference to lock file in Solaris apcupsd.in so it is
 properly configured.
- Removed unused variable from Solaris apcupsd.in
- Install apcupsd.conf with 644 permissions
- Add --disable-install-distdir ./configure option.
 This allows easier installation of slaves or
 multiple copies of apcupsd.
- Used makedepend if not gcc (Solaris fix).
- Set SO_REUSEADDR in slave machines.
- Use select() to timeout master if slave does not respond.
- Recognize APC Smart-UPS 370ci.
- Add 940-0020C cable support (same as 20B).
- UPSNAME now sets upsname if given. Otherwise, apcupsd
 attempts to get name from UPS, if not found, uses
 hostname, finally "default".
- Implement CommLost NIS status.
- Implement Shutdown NIS status.
- Implement Slave NIS status.
- Correct SmartTrim and SmartBoost detection code.

```

## APC UPS management under Linux

- ./configure prints name of shutdown program.
  - Add port to hosts.conf.
  - Add new apccontrol arguments to script file.
  - Eliminated all N/A fields in STATUS report.
  - Always construct STATFLAG in STATUS report.
  - Added Ambient Temperature and Humidity to multimon (Carl Erhorn)
- 





## New Features in Apcupsd 3.8.2

### · Highlights of this release:

- . Detects Self Test and reports it as such rather than a Power Failure.
- . True pthreads implementation (uses less CPU and one third the memory).
- . For SmartUPSes, apcupsd does a much better job of adapting to the actual features of the UPS and is more efficient.
- . Many new ./configure options permitting much more complete pre-configuration. Nearly every possible file and feature is configurable.
- . More complete and updated support for more systems (Caldera, Debian, FreeBSD, NetBSD, OpenBSD)
- . Correction of all outstanding 3.8.0 and 3.8.1 bugs.
- . Cascading Style Sheet version of multimon for user customization.
- . TCP Wrapper support (untested).
- . KILLDELAY configuration directive requested by users (untested).
- . Reap zombies on BSD systems.
- . Win32 shutdown program significantly enhanced, now shuts down in all known configurations giving users time to save their changes.
- . Win32 version uses threads rather than Unix processes.

The software is completely developed under Linux and will compile cleanly and will work under Linux as well as many other operating systems and flavors of Linux.

What to do if you find bugs :

send an email to [apcupsd-devel@apcupsd.org](mailto:apcupsd-devel@apcupsd.org) (Developers mailing list) or go to one of the following sites:

<http://www.apcupsd.org>

<http://www.sibbald.com/apcupsd>

### Change Log

- ```

----> Release apcupsd-3.8.2 (3 July 2001)
    - Additional documentation.
    - Bug fix provided by Riccardo.
    - Update script for making RPMs.
    - Please remember that apcnetd has been renamed to
      apcnisd
    - Tweaked the pthreads flags for FreeBSD
    - Added Linux From Scratch hint file to unknown distribution.
    - Fixed Makefile in lib to use CFLAGS.

----> Released apcupsd-3.8.2Beta14 (24 Jun 2001)
    - Fixed stall in Network Information Server
    - Fixed possible race condition in pthreads
      Network Information Server.
    - Fixed a serious shutdown bug for BackUPS UPSes.
    - Tightened security for some scripts (thanks Neil).
    - More cleanups for --with-pwrfail-dir so it
      is properly configured.

```

APC UPS management under Linux

See techdocs/kes27Jun01 for more details.

- > Released apcupsd-3.8.2Beta13 (21 Jun 2001)
- Fixed bug in creation of PWRFAIL file. Thanks to Jose for his excellent testing efforts.
 - Cleaned up configuration of PWRFAILDIR in distribution files.
 - Fixed installation of multimon.css
 - Implemented localtime_r() for Win32 systems.
- > Released apcupsd-3.8.2Beta12 (20 Jun 2001)
- Fixed CGI Makefile problems pointed out by Carl.
 - Fixed SuSe install problem pointed out by user (and fixed provided by Neil).
 - Fixed powerflute, which has been broken for a long time.
 - Upgraded powerflute to work with pthreaded apcupsd (via TCP/IP).
- > Released apcupsd-3.8.2Beta11 (12 Jun 2001)
- Mostly documentation and small bug fixes.
 - Reworked installation of CSS files in CGI directory.
 - use localtime_r() to avoid potential problems in threaded version.
 - Cleaned up a few serious shutdown problems with new apcaction code pointed out by a user (thanks!).
 - Fix bug in libwrap code.
- > Released apcupsd-3.8.2Beta9 (28 Apr 2001)
- As promised with release apcupsd-3.8.1-5, I have now added pthreads support -- well, at least it is a first cut. The reentrant library calls must still be added where appropriate.
 - Configuration with ./configure is now much more complete. See kes09Apr01 and kes10Apr01 in techlogs.
 - See kes28Apr01 in techlogs for more details of the pthreads implementation.
 - apcaccess and powerflute need NIS running to work with a threaded apcupsd.
 - Set SO_REUSEADDR and SO_KEEPALIVE on sockets.
- [ChangeLog] I
- > Released apcupsd-3.8.2Beta6 (10 Apr 2001)
- Much enhanced ./configure. Lots of new options for easier system configuration.
 - New Debian code.
 - New release numbering scheme to eliminate -nn at end for packagers.
 - Shared memory initialization improvements.
 - A single apcupsd.conf (built from apcupsd.conf.in) in the /etc directory.
- See techlogs/kes9Apr01 for more details.
See techlogs/kes10Apr01 for more details.
- > Released apcupsd-3.8.1-5 (6 Apr 2001)
- This version is a pre-release of version 3.8.2. It contains fixes for most known problems.
 - Major work has been done to include in this release much of the code that will go into version 4.0.
 - Addition merging of 3.8 and 4.0 will occur prior to the final release of 3.8.2

APC UPS management under Linux

- The only major addition that I am planning to 3.8.2 for the next prerelease (3.8.1-6) will be pthread support (at least the first cut).
- The name of apcnetd has been changed to apcnisd
- OpenBSD fixes from Devin Reade
- Fixed Zombies that were created on BSD systems.

New Features:

- New indenting standard (see code). Not yet uniformly applied -- work in progress.
- New NetBSD support.
- Updated Debian support.
- apccontrol now called with additional arguments.
- GNOME realtime monitoring program added in gupsc directory (not supported).
- Improved shutdown on WinNT.
- Self test detection -- no more false power fails.
- TCP Wrapper support (untested).
- New KILLDELAY configuration directive (untested) that causes apcupsd to wait after issuing a shutdown, and after the delay seconds have expired, it will issue a kill power to the UPS. Potentially useful on systems where apcupsd cannot regain control after completing a system shutdown (i.e. WinNT).
- Multimon can now use css for more control by the user.

See techlogs/kes06Apr01 for details.

See techlogs/gdr01Mar01 for more details

----> Release apcupsd-3.8.1-3 (5 Mar 01)

- Added Caldera code.
- NetBSD patches.
- Modified code for handling 940-0095B cable.
- Integrated 4.0 code

See techlogs/kes05Mar01 for more details.

----> Release of 25 Feb 2001

- New NetBSD code (thanks to a number of contributors).
- Added O_NDELAY to prevent blocking on serial port opens on BSD systems.
- Fixed a bug in the status display on BackUPS Pros that cut the output short.
- Fixed potential DoS problem with email scripts.
- Much improved shutdown program for Win32 systems.

See techlogs/kes27Feb01 for additional details.



New Features in Apcupsd 3.8.1

· Highlights of this release.

The software is completely developed under Linux and will compile cleanly and will work under Linux. Porting is being addressed for Solaris and HP-UX, where the actual code should compile cleanly. If you own an APC UPS attached to a Linux machine or you want to realize a setup like this, or you are already an user of older version of **apcupsd**, please upgrade to the latest version. We need testers and any bug reports will be more than welcome. ·

What to do if you find bugs :

send an email to apcupsd-devel@apcupsd.org (Developers mailing list) or go to one of the following sites:

<http://www.apcupsd.org>

<http://www.sibbald.com/apcupsd>

Change Log

```

/*****
/*          ChangeLog of apcupsd                      */
/*          Riccardo Facchetti                        */
/*          http://www.apcupsd.org                    */
/*          http://www.sibbald.com/apcupsd            */
/*          ftp://ftp.apcupsd.org                    */
*****/

```

----> Released apcupsd-3.8.1

```

    See techlogs/kes08Dec00
    Corrected depend creation in config.status.
    See techlogs/kes05Dec00 for details
    Fixed Win32 to work with simple signalling UPSes.
    Upgraded Win32 to CGYWIN 1.1.5.
    Fixed an Alpha Tru64 64/32 bit problem in the networking code.
    Fixed the random incorrect on battery timers (I think).
    Upgraded to latest version of gcc
    Do much better argument checking in the .cgi programs. This
        improves security (at least psychologically).

```

----> Released apcupsd-3.8.0

```

    Additional updates to apctest.c, it now tests
        Smart UPSes as well as Dumb UPSes
    Increase getline buffer size
    Quick documentation of apctest.c

```

----> Released apcupsd-3.8.0-pre7

```

    See techlogs/kes27Oct00
    Win32 make binary_tar_file
    Rewrote the 940-0119A cable code based on Jason A. Smith's testing.
    Improved apcupsd version print out.
    Win32 and Signaling UPS documentation.

```

----> Released apcupsd-3.8.0-pre6

```

    See techlogs/kes21Oct00
    A number of Win32 fixes including an email program for Win95/98.

```

APC UPS management under Linux

Put Win32 system binaries in source code.
Fixed WinNT problem when running as a master or slave.
Win32 make install.
Minor corrections to scripts.
Fixed a Solaris xlib link problem in the cgi directory.

----> Released apcupsd-3.8.0-pre5 for testing
See techlogs/kes17Oct00
Applied André's fixes to support the 0119A cable used by the
BackUPS Office 500.
Applied Riccardo's fix for Solaris xlib.

----> Released apcupsd-3.8.0-pre4
Applied Kern's diff, see techlogs/kes06Oct00
Key points: apcaction.c fix for timer starting during power failure,
install key event handlers by default that email the
following events: changeme, commfailure, commok, mainsback,
and onbattery.

----> Released apcupsd-3.8.0-pre3
Applied Kern's diff, see techlogs/kes05Oct00
Key points: fixes to Win32 binary release, manual updates,
TIMEOUT bug fix, handle pre-1994 UPSes correctly,
portability issues.

----> Released winapcupsd-3.8.0-pre2 Win32 binarys only
Applied Kern's corrections to the Win32 binary files.
Fix to /win32/apccontrol.in
Add sh.exe to \apcupsd\bin directory.

----> Released apcupsd-3.8.0-pre-1.
Applied Kern diffs, see techlogs/kes27Sep00
Key points: Alpha Tru64 port, documentation updates,
additional retries in networking, new STATUS variables,
tighten permission of files, RedHat RPM spec, corrections
to event scripts.

----> Snapshot released 20000911.
Applied Kern diffs, see techlogs/kes10Sep00
Key points: Win32 enhancements, buffer overrun
fix, tightning of file permissions, manual updates.

----> Snapshot released 20000910.
WIN32 (cygwin) enhancements (KES): win'ization and use of system tray.
Documentation update.
autoconf reorganization.

----> apcupsd-3.7.2 released.
true and false programs are no more hardcoded into the configure script.
Reorganized some of the distributions/ Makefiles.
Added FreeBSD installation support.

----> Snapshot released 20000528
Now apcaccess.c::stat_print() output status to stdout instead of stderr.
Reorganized the kill_* code in apcupsd.c::main()
Removed the kill_ups_power argument from apcnet.c::kill_net()
Moved all save_dumb_status() calls to apcserial.c::setup_serial()
Applied kern's patch for kill_net in apcupsd.c and some other
beautifications of the code.
Changed scripts/autoregen.sh to compile with all the options switched on.
Added a web page of thanks to the html manual.

APC UPS management under Linux

Moved apcproctitle.c to lib/proctitle.c. Use that source only if there is not a system setproctitle(3).
Don't use 's' subflag to ar(1). It's unnecessary since ranlib is being called, and more importantly not all ar implementations have it.
Install distribution-specific apcupsd.conf files if they exist.
Fixed some calls to open(2) that didn't specify file create modes.
Added OpenBSD distribution and minor source patches.
Modified Slackware distribution: Update README, patch instead of replacing rc files, delete obsolete distribution files.
Updated "makediff" developers' tool.
Minor manual updates.
Updated again Developers file: we gained more *BSD help.
Added the '-p' option to cmdline: was missing.
Abort on killpower for slaves instead of simply exit, in apcupsd.c
Removed nologin information from apcnet.c because we don't need. All the nologin file is managed by apcaction.c

----> Snapshot released 20000511
New maintainer for Slackware and OpenBSD ports: updated Developers file.
Applied David patch for slackware
Applied kern diffs, see techlogs/kes26Jan00
po/POTFILES file is now removed by main Makefile since is always built.
Corrected Slackware distribution Makefile.in: it was tied to /usr/src. Thanks to John McSwain.
Corrected a bug in configure.in and include/apc_defines.h in which the nologin file was created in /etc/apcupsd/nologin instead of /etc/nologin.
Moved README.solaris to doc/.
Moved the patch "./1" to techlogs/kern-patch-17-Feb-2000

----> apcupsd-3.7.1 released 17 February 2000
Bumped version to 3.7.1 and released by Kern since Riccardo is on vacation.
Update to upsible.html to include additional credits.
Fixed dumb UPSes, which did not work in version 3.7.0, see Kern's fixes techlogs/kes12Feb00

----> apcupsd-3.7.0 released.
Bumped version to 3.7.0 and released.
Powerflute: last 8 events are loaded from events file (if present).
Revamped powerflute: now curses work correctly.
Small fix to include/apc_config.h to clean Solaris compile.
Clean intl/ a bit better.
Fixed apcnetlib.c and apcnetd.c: now apcupsd compile under FreeBSD too.
Applied kern diffs, see techlogs/kes26Jan00
Fixed compile for Solaris 2.5.1.
Added check for snprintf when CGI is configured in configure.in.
Fixed internationalization inclusion when msgfmt is not found.
Fixed warning in lib/getopt.c.
Fixed AC_PATH_PROGS in configure.in.
Verified clean compile under Linux and HP-UX.
Added libintl and nls in information at the end of configure.
Corrected stpcpy warning in intl/dcgettext.c.
Changed all `make -C', this fixes HP-UX compilation of additional modules like internationalization and cgi support.
Fixed detection of UP-UX version.
In apcnetlib.c use memset instead of bzero.
In apcupsd.c don't use TIOCNOTTY if not defined.
Corrected a macro bug in arguments of setpgrp() call in apcupsd.

----> apcupsd-3.7.0-rc1 released.

APC UPS management under Linux

Updated my e-mail and (C) signatures: if you want to make it simple, use vim and the scripts you can find in scripts/.
Restructured include/apc.h inclusion order: now make more sense.
Corrected setpgrp() call in apcupsd.c for BSD/non-BSD.
Added FreeBSD to configure autodetection and placeholder in distributions/.
Other minor fixes to Makefile.in and cgi/Makefile.in
Fixed make clean for cgi.
Applied kern diffs, see techlogs/kes17Jan00
Added slackware scripts from John McSwain.
Added Kern's fixes to distributions/redhat/awkhaltprog.in
Applied kern diffs, see techlogs/kes16Jan00
Added handling of `exit 0' at the end of halt.local as in suse 5.x.
Added autodetection for SuSE 5.x.
SuSE installation: written a shell script for installing apcupsd directives in /etc/rc.config.

----> apcupsd-3.7.0-beta4 released.
Bumped version to beta4 and released.
Fixed a minor po/ problem.
Applied kern diffs, see techlogs/kes13Jan00
Applied kern diffs, see techlogs/kes12Jan00
Fixed CYGWIN detection in configure.in.
Added clean and distclean for distributions/.
Added slackware detection but still manual installation (scripts generated but no actual installation performed: too dangerous doing the code without being able to test).

----> apcupsd-3.7.0-beta3 released.
Bumped version to beta3 and released.
Applied kern diffs, see techlogs/kes09Jan00
Changed clean_threads() in apcexec.c not hang on waitpid, and make sure we do all we can to kill the childs.
Updated INSTALL file to make clear that prior to install a new version of apcupsd over an old version, is advisable to make uninstall.
Corrected a syntax error in distributions/*/apccontrol.sh.in.
Changed suse halt.local script to:
 . do something only if a powerdown is detected.
 . kill the processes before remounting read only the filesystems.
Corrected a minor cosmetic (-Wall) in apcnetd.c
Added -Wall to default CFLAGS.
Applied kern diffs, see techlogs/kes06Jan00
Kern minor changes to configure.in.
Thanks to Tom Schroll, corrected an execv nasty error in apcexec.c.
Applied kern diffs, see techlogs/kes30Dec99

----> apcupsd-3.7.0-beta2 released.
Bumped version to beta2 and released.
Changed install-apcupsd target in main Makefile.in not to overwrite old apcupsd.conf file and instead create an apcupsd.conf.new file.
Applied Kern diffs, see techlogs/kes19Dec99
Changed the install to warn the user that if an old apcupsd.conf is in place, it is saved to apcupsd.conf.old.
Applied Kern diffs, see techlogs/kes18Dec99.
Reorganized the path construction in include/apc_defines.h so that now file paths are built with autoconf variables and not hardcoded.
Applied Carl Erhorn patches for Solaris, see techlogs/cpe16Dec99.
Now SuSE 5.2 is correctly detected (don't know previous versions).
Cleanups of configure.in.
Corrected a problem with SuSE and halt scripts where killpower where not issued correctly.
Applied Helmut Messerer corrections.

APC UPS management under Linux

Applied Kern diffs, see techlogs/kes09Dec99
Applied Kern diffs, see techlogs/kes08Dec99
Added a new contrib/ directory where user contributed files are put.
New user contribution for sending sms messages on UPS troubles.

----> apcupsd-3.7.0-beta1 released.

Applied Kern diffs, see techlogs/kes30Nov99
Applied Kern diffs, see techlogs/kes28Nov99
Applied Kern diffs, see techlogs/kes20Nov99
Applied Kern diffs, see techlogs/kes18Nov99
Applied Kern diffs, see techlogs/kes15Nov99
More internationalization.
Applied Kern diffs, see techlogs/kes13Nov99
Moved TIOCM_LE HP-UX define and friends to include/apc_config.h
Internationalization: internationalized and translated in Italian all the messages that are printed with printf and fprintf: more need to be done for error_* functions.
Removed #ifdef wrapping around debug code and substituted with debug_level checks.
Cleaned a remaining spit of 'killpower' in apcupsd.c
Help screen output to stdout.
Now (C) and Brian's Support Center are visible in help screen output.
Updates to the cgi files from Kern.
Added a vimrc for TABs.
--debug argument changed meaning. Now is meant to be set to a number that range from 0 to N where increasing numbers means increasing debug output.
Now we use getopt_long [see getopt(3)] for parsing command line.
Documentation updates and reorganization: now 'developers' documentation is in doc/developers/.
Added support for cable 940-0095B.
Applied Kern patches: syslog, getline, code cleanups, new cgi interface. Read his technical log in techlogs/kes03Nov99
Added doc/CodingStyle file.
Added Developers file.
Some minor code documentation.
Removed the "failed to reacquire the lockfile" problem. Now we release the lockfile just before fork() and reacquire it just after.
More duplicated code cleanups and nasty bugs fixed in apcaction.c.
SuSE's apcupsd start script now return green "done" and red "failed" (SuSE 6.2)
apcccontrol is installed in /etc/apcupsd/ because we need it when filesystems may be umounted.
Removed powersc script: now it is all done in apcccontrol and /etc/rc.d/apcupsd scripts.
Moved _all_ the scripts into scripts/ directory (where they belong).
apcccontrol script installation dir is now sysconfdir
\${prefix}/etc/apcupsd
Simplified apcaction.c (removed duplicate code).
Attempt to document some features of apcupsd.conf.
Deleted two alarm()s related to apcreports.c from apcaction.c: this was a bug.
Added setproctitle for setting forked procs's argv[0]. Now we have
apcmain -> waiting for other tasks to exit, generic watchdog.
apcser -> serial task
apcact -> actions task
apcnet -> network task
apcslv -> netslave task
so that now we can tell with a 'ps' which task is doing what.
apcserial+apcreports are now one single thread.
Better locking scheme.
Removed another nasty bug in Old* variables into apcaction.c. Now

APC UPS management under Linux

there is a local structure for these values.
Removed a nasty bug in apcsetup.c.
Updated man pages and apcupsd.conf.
RedHat installation scripts.
Removed 10 seconds sleep() from terminate().
SHM ID for sanity checks on SHM accesses.
Obsolete config options now generate only warnings.
New error handling routines.
doc/README.developers updated.
Syslogging functions are bracketed with HAVE_GCC so that with gcc we can use macros and with other compilers we use functions for compatibility.
Syslog functions are now real functions. There's no point in having incompatible preprocessor macros when a set of little functions can do the job.
Fixed a next_slave label in apcnet.c with a semicolon for HPUX compiler.
OS detection in configure: now Makefiles and sources know about which OS they are supposed to compile for.
Better integration of lib/ sources into the configure mechanism.
configure cleanups.
Added cflags and ldflags selection in Makefile.in and configure.
Fixed getopt_long detection in configure. Now if not found it compile the lib/ version.
Now configure.in have a hardcoded PATH in which search the system programs needed.
Lot of cleanups in global variables and code partitioning and duplication.
More configure cleanups.
Updated RH 6.0 halt script.
Removed apchttp.
Cleaned THREADS. Now forked processes are the only option.
Various cleanups.
SuSE-specific install/uninstall.
All distribution specific directories are now in distributions/.
All scripts are now in scripts/.
Moved default apcupsd.conf in etc/.
Added distribution-specific scripts installation.
Install /etc/apcupsd.conf if not already present.
Removed apc(un)install.sh: no more needed.
Rewritten external scripts: no more system(), better customization support.
Removed Makefile.in.in from po/: it's not needed.
Fixed bogus --enable and --disable behavior of configure.
Cleaned up po/ and intl/ autoconf.
Only powerflute is linked with ncurses libraries.
The program makedepend is not any more vital for compilation.

----> apcupsd-3.6.2

- Fixed the "apcupsd -c" configure command.
- Fixed two thread bugs in alarm handling.
- Fixed two potential security exploits.
- More cleaned up autoconf.
- Random documentation cleanups.

----> apcupsd-3.6.1

- Cleaned up new autoconf stuff from version 3.6.0
- Undefined a test flag for testing networks wierdness.
#undef WACKY_NETWORK_ATTEMPS
- Network is fully functional under non-threaded compile.
- Possible fix for "pipe_master_status" calls on slaves.

APC UPS management under Linux

Added 940-1524C smart signal cable support.

```
----> apcupsd-3.6.0
    Added autoconf.
    Added internationalization support. There is _only_ the support but
    no current code is written for the intl package. It can be compiled
    in, but intl strings have still to be translated (to be done in the
    future).
    Reorganized documentation.
    Reorganized support for distributions. Now we have a directory for
    every possible distribution (suse, unix, debian etc etc) so that the
    job for the package-men can be easier.

----> apcupsd-3.5.9
    Added new configuration options to reduce init time of daemon.
    powersc CONFIG
    powersc NAME
    powersc BATTERY

----> apcupsd-3.5.8.patch

    Fixes a FIFO error that I forgot to include in the rush
    to release the code

----> apcupsd-3.5.8

    GPL2 source code status finally.....April 7, 1999

    Threaded code is stable but requires glibc2 or libc6
    Finished SELFTEST setup
    Fixed naming of UPS if allowed.
    Redesigned "apcsetup.c" to allow for ease of parameter setup.

----> apcupsd-3.5.7

    Added 940-0024G cable.
    "newbackupspro" to be replaced by "backupspropnp"

    If you have a BackUPS Pro UPS that is identified as
    BP(SIZE)(EXTRAS) examples BP420SC/BP650SC/BP650PRO-PnP,
    then you have a PnP BackUPS Pro that is very near a SmartUPS.
    Else anything with the identifier BK(SIZE)PRO or BKPRO(SIZE)
    is the very early version of the "dumbed down smartups".
    Unfortunately, it has very little to say. It is superior to the
    classic BackUPS (simple signals) in that it can at least tell event
    histories and if the batteries are faulted.

    now powerflute is compiling and working, needs THREADS

----> apcupsd-3.5.6

    See Makefile for enabling flags.

    # New multi-threading code BETA
    #
    # THREADS    = 1

    # New multi-threading code with http BETA functional needs THREADS
    #
    # HTTP      = 1
```

APC UPS management under Linux

```
# New/Old powerflute tool
#
# MUSIC      = 1

dual building model possible complete.
intergrated beta http data
breakages in logs and procfs are still present in shm.
code sorting and corrections.
return of ncurses powerflute tool.
fixed Makefile to find two possible locations of
ncurses package; however, there are three methods for this animal.

apccconfig.c: removed check for existance of lock directory. It's not
needed and dangerous when -killpower !
apccconfig.c: new configuration checks
added semun definition to apc_struct.h since it is needed for glibc6

----> apcupsd-3.5.5

dual building model begun.
intergrated beta shared memory mapping.
breakages in logs and procfs are present.

----> apcupsd-3.5.4

Preparation for GPL status and transfer to GNU.
Fixed Network bug.
Fixed IPC PIPE bug.

----> apcupsd-3.5.3

Fixed UPS killpower bug.

----> apcupsd-3.5.2

Source wide reformat to conform to standard C programing format

----> apcupsd-3.5.1

----> apcupsd-3.5.0

fixes "NO" reports in setup loops for UPSes that do not allow for
changing parameters at initialization. Previously it was assumed that
UPSes did not report anything if the feature was not pollable and/or
changeble.

----> apcupsd-3.4.9

fixes a long missed error that was incorrectly fixed in the past.

----> apcupsd-3.4.8

solved more mystery functions and the UPSlink language may
be completely decoded.

----> apcupsd-3.4.7

now auto-learns features based on UPS's answer to questions.
initial porting to Solaris for i386 is complete.
```

APC UPS management under Linux

```
----> apcupsd-3.4.6 45493 Jul 15 13:14 apcupsd.c
                22326 Jul  9 12:50 apcnet.c
                12129 Jul  9 12:50 apcpipes.c
                25634 Jul  9 12:50 apcconfig.c
                8565 Jul 15 13:07 apcsetup.c
                11039 Jul 15 13:09 apcreports.c
                18137 Jul 15 13:19 apcsmart.c

all logging functions will be moved to "apcreports.c"
all smart mode calls will be moved to "apcsmart.c"
fix ups model reporting in "apcsetup.c"

----> apcupsd-3.4.5 77096 Jul  7 17:18 apcupsd.c
                22326 Jul  7 03:12 apcnet.c
                12129 Jun 22 15:12 apcpipes.c
                25634 Jul  7 16:39 apcconfig.c

                18477 Jul  7 02:08 apcaccess.c

Added NOLOGON delay for systems with large Matrix UPSes.
Short information polling, with constant values being set
in the extended setup functions.

----> apcupsd-3.4.4
Fixed a missing and needed wait delay for netmaster systems.
This was discovered when mixing flavors of Linux.
Since SuSE, RHS, and Debain require extra time for shutdowns.
man-page is closer to date.....

----> apcupsd-3.4.3
Added new limit features by polling a new command for
internal calculated remaining time on line.
Fixed naming UPS if allowed.

----> apcupsd-3.4.2
Fixed an unknown error for the forced ups-kill for the
backupspro models. This was discovered with a new
BackUPS Pro 1000.

----> apcupsd-3.4.1
Changed ./includes to have an "apc_" prefix.
This is for initial port requirements to FreeBSD.

----> apcupsd-3.4.0 72305 May 19 14:18 apcupsd.c
                21597 Apr 16 17:59 apcnet.c
                12117 Apr 16 15:44 apcpipes.c
                22832 Apr 16 15:42 apcconfig.c

                17687 Apr 16 13:47 apcaccess.c

Slave management disconnect/reconnect added to apcaccess.
Fixed excessive loss of UPS communications loggings of ::
    UPSlink Comm. Error, SM != SM
    UPSlink Comm. reestablished, SM == SM

Fixed Signal/Cable Combination errors.
    A "SmartUPS" with a "Simple Cable" with report as a "BackUPS".

----> apcupsd-3.3.0 72450 Mar 20 01:09 apcupsd.c
                15321 Mar 20 01:08 apcnet.c
                11093 Mar 20 01:09 apcpipes.c
```

APC UPS management under Linux

22465 Mar 20 01:09 config.c

17300 Mar 20 01:08 apcaccess.c

Network death solved (hopefully)
PowerFlute removed due to unexplainable daemon kills under 2.0.X. and
some cases of 2.1.X. (TCPIP)

This is replaced by the original tool that now works "apcaccess".
"apcaccess" is functional under both 2.0.X and 2.1.X kernels without
killing the daemon "apcupsd".

EPROM programming of many UPS models is now functional.

Example::

apcaccess : polling apcupsd for status.

```
APC      : Mar 20 01:24:56
CABLE    : APC Cable 940-0024B
UPSMODEL : SmartUPS
UPSMODE   : Stand Alone
ULINE    : 124.0 Volts
MLINE    : 125.2 Volts
NLINE    : 124.0 Volts
FLINE    : 60.0 Hz
VOUTP    : 124.0 Volts
LOUTP    : 028.0
BOUTP    : 27.3 Volts
BCHAR    : 100.0
BFAIL    : 0x08
SENSE    : HIGH
WAKEUP   : 060 Cycles
SLEEP    : 020 Cycles
LOTRANS  : 103.0 Volts
HITRANS  : 129.0 Volts
CHARGE   : 025.0 Percent
UTEMP    : 49.5 C Internal
ALARM    : Low Batt + 30
DIPSW    : 0x0000
```

```
root@Orion% cat apcupsd.status
APC      : Mar 20 01:27:31
CABLE    : APC Cable 940-0024B
UPSMODEL : SmartUPS
UPSMODE   : Stand Alone
UPSNAME   : UPS_IDEN
ULINE    : 124.0 Volts
MLINE    : 124.6 Volts
NLINE    : 124.0 Volts
FLINE    : 60.0 Hz
VOUTP    : 124.0 Volts
LOUTP    : 028.0 Load Capacity
BOUTP    : 27.3 Volts
BCHAR    : 100.0 Batt. Charge
BFAIL    : 0x08 Status Flag
SENSE    : HIGH
WAKEUP   : 060 Cycles
SLEEP    : 020 Cycles
LOTRANS  : 103.0 Volts
HITRANS  : 129.0 Volts
```

APC UPS management under Linux

```
CHARGE    : 025.0 Percent
UTEMP     : 49.5 C Internal
ALARM     : Low Batt
DIPSW     : 0x0000
```

----> apcupsd-3.2.2 66556 Jan 27 15:24 apcupsd.c

Fixed a reporting error in /etc/apcupsd.status for CUSTOM SIMPLE cable.
Fixed a reporting error in /var/log/apcupsd.log for CUSTOM SIMPLE cable.

----> apcupsd-3.2.1 67450 Jan 8 13:41 apcupsd.c

Fixed INSTALL from "readhat" to "redhat"
Fixed INSTALL errors of "slackware"

Bug search that causes network death, lost slaves under 2.0.X only??
Bug search that causes text-powerflute to kill daemon only under 2.0.X.

----> apcupsd-3.2.0 66096 Dec 18 16:11 apcupsd.c

apcflute.c all code for reconfiguration complete.

----> apcupsd-3.1.0 65925 Dec 8 21:18 apcupsd.c

----> apcupsd-3.0.0

Now binaries only.....

----> apcupsd-2.9.9 63431 Dec 3 13:05 apcupsd.c

Fixed "for (killcount=0;killcount>3;killcount++)" bug
^ killcount.

----> apcupsd-2.9.8 62158 Nov 25 15:13 apcupsd.c

Added user defined magic and security timeouts.
Started powerflute update running master or standalone daemon.
Added another field for SmartUPS procfs file.

----> apcupsd-2.9.7 60945 Nov 18 19:19 apcupsd.c

Added second alternate method for killing power for non SU SmartUPS.
I have one of these older models, and it needs a bigger kick.

Disable lockfiles if slave is a netslave with (ups->cable == APC_NET),
or ethernet.

----> apcupsd-2.9.6 60156 Nov 10 22:34 apcupsd.c

split source files.....

```
10885 Nov 10 20:40 apcnet.c      :: NetUPS
18805 Nov 10 22:37 config.c     :: configuration
```

```
5924 Nov 10 19:54 apcflute.c.src  :: powerflute interface to apcupsd
5264 Nov 10 12:43 powerflute.c.src :: powerflute tcp management of apcupsd
```

Brian Schau added lockfiles for serial ports.

NetUPS or EtherUPS now works.
Needs more security likely, you can set your own TCP port number.

APC UPS management under Linux

Will add user MAGIC at compile in the future.

----> apcupsd-2.9.5 72431 Oct 28 00:27 apcupsd.c

Listing my hack of "powerflute". To enable it edit your /etc/apcupsd.conf and set NETTIME to any value seconds is something to play around.

Reformat subroutines to get apcupsd to handle networks.

----> apcupsd-2.9.4 70707 Oct 25 13:51 apcupsd.c

Smart V/S is in the same class as a BackUPS Pro.

Riccardo Facchetti has been volunteered to solve the network part for the project. YEAH!!!!!!!!!!!!!!

Hello Andre,

I own a SmartUPS v/s and I have played a bit with your apcupsd. I have found that the SmartUPS v/s command set is similar (if not equal) to the one of BackUPS PRO, so I have changed the apcupsd.c code to behave this way and seems to work well.

Another thing: I would like to have an application like powerchute. Of course I am willing to write it, so I will call it powerflute.

My idea is:

```
apcupsd -> daemon
- controls power status
- listen on a TCP socket for incoming connections
  - get a command from TCP line
  - send the response to the command
```

```
powerflute -> application
- connect to apcupsd
  - sends a command to apcupsd
  - get the response from apcupsd
```

apcupsd is listening to the socket with a select so it don't block. After one or more connections are established, it read non-blocking from the TCP file descriptor(s) and when a char is available, send it to UPS and readline(), then send the buffer (terminated with '\n' to the TCP client over the connection file descriptor.

You can note that I have protected apcupsd UPS monitoring from TCP servicing by doing only non-blocking calls to the TCP layer, so there should be no problem for it in detecting power problems without delay caused by TCP connections.

A patch is enclosed for v/s == BKpro and for TCP servicing. Of course this patch is still very preliminar, but I think it is a good idea.

You can try the idea running the apcupsd with this patch applied, telnetting to the port 6669 from localhost (no external connections allowed now for security) and sending to the daemon some commands for the UPS. The daemon will redirect them to the UPS and send the answer to you. You can connect more than one client to the daemon (max 64 connections).

Please send me comments about it.

APC UPS management under Linux

Ciao,

Riccardo.

--

Riccardo Facchetti | e-mail: riccardo@master.oasi.gpa.it

The TCP patch is not there but my hack job is present.
The package will have an addition called "powerflute".

----> apcupsd-2.9.3 67655 Oct 23 14:57 apcupsd.c

Fixed spelling of "dissable" to "disable"
Forgot to add this function for ShareUPS.

```
int fillShareUPS(int sharefd, UPSINFO *ups);
```

Brian Schau with needed install changes in Makefile
for UNIFIX.

----> apcupsd-2.9.2 67410 Oct 22 19:34 apcupsd.c

Changed "case" names in /etc/apcupsd.conf, again.....
This should be the last time.....

ShareUPS project maybe nearing an end.....

make install is almost done.

----> apcupsd-2.9.1 65886 Oct 19 00:30 apcupsd.c

Changed "case" names in /etc/apcupsd.conf to lower case.
UPSTYPE SMARTUPS is now UPSTYPE smartups
Changed and add two new UPSMODE "cases" for ShareUPSES
Added Two new external call command for TIMEOUT and LOADLIMIT.
Shutdown types are now called by event not by a general shutdown.
Added CHANGEEME to /sbin/powersc for SmartUPSES to announce that
its needs a batteries changed.

Tests are now run on (2) BackUPS 600's, SmartUPS Net 1400RM 700RM,
and SmartUPS 1250.

SmartUPSES now can call for shutdown based on percent of remaining
battery life or capacity.

Cleaned up manpage.....

----> apcupsd-2.9 58603 Oct 16 22:17 apcupsd.c

Changed to POSIX Style of control.
Now uses an external CONFIG file -> /etc/apcupsd.conf
All events are handled in the /sbin/powersc file, no more
sysvinit flavor hassels.....
Added ManPage, well an attempt.
Other things begun.....

----> apcupsd-2.8 42800 Oct 6 14:36 apcupsd.c

Finally solved the PLUG-N-PLAY Cable.....
Tested on a BackUPS 600 and SmartUPS 700RM

Someone needs to test a BackUPS Pro.....

APC UPS management under Linux

If main power returns during a shutdown series, "apcupsd" will now reboot.

---->

Better fix and support for #940-0023A cable, but still broke, drat.

---->

Adds new code for UNIFIX Linux

----> apcupsd-2.7.1 35902 Sep 3 21:38 apcupsd.c

By Chris:

More changes and fixes with Pro Series tested.

Better method of "walling" the console.

Minor debugging added.

ShareUPS Project Started, BETA.

----> apcupsd-2.7 35413 Aug 14 15:50 apcupsd.c

TESTED on the following setups:

SLACKWARE SMP/NON-SMP (2.1.49) SmartUPS 700 BackUPS 600

REDHAT 4.1 SMP/NON-SMP (2.0.30) BackUPS 600

SUSE 5.0 SMP/NON-SMP (2.0.30) BackUPS 600

NOTE BackUPS Pro Series Untested at this time of release.

All UPSes can be time limited shutdown.

Fixed all possible killpower problems.....

Changed rc.power to include a NOW command.

Changed inittab file and added

What to do when battery power fails.

pn::powerfailnow:/etc/rc.d/rc.power now

Made changes for SuSE distribution.

Change /sbin/init.d/powerfail file

Chanages to CONFIG File

USE_SLACKWARE=yes

USE_REDHAT=no

USE_SUSE=no

Auto Updater

USE_MAKE_UPDATES=yes

USE_MAKE_INITTAB=yes

RedHat version for "powerstatus" file

REDHAT_VERSION=42

----> apcupsd-2.6 35718 Jul 10 08:10 apcupsd.c

Added more info to apcupsd.maunal about special external controls.

SEE "6) OTHER" in apcupsd.maunal

APC UPS management under Linux

----> apcupsd-2.6.pre6 35718 Jul 10 08:10 apcupsd.c

Made changes in rc.power to become more universal.
Made N-th attempt to make use of APC cable #940-0095A "DRAT"
in "dumb mode". Will now try "smart mode".
"DOUBLE DRAT" APC has a new cable #940-0095C.
They just want to make things hard for us.....
More Pro Fixes

S.u.S.e work around. This is a major DANGER!!!!
If you call /usr/sbin/apcupsd /dev/apcupsd killpower,
in a non-power problem situation.
"YOU WILL SCREW YOURSELF" this safe guard is a direct override
of the daemon.

----> apcupsd-2.6.pre5a 36279 Jun 23 16:39 apcupsd.c

OOPS.....forgot an "endif" in the distribution Makefile

----> apcupsd-2.6.pre5 36279 Jun 23 16:39 apcupsd.c

Added Linux Flavor Dependencies in CONFIG file.
Fixed "nologin", so only root can login during a powerfailure.
All other attempts are defeated at the login prompt.

----> apcupsd-2.6.pre4 35616 Jun 19 15:40 apcupsd.c

Possible RedHat 4.2 fix.

----> apcupsd-2.6.pre3 35282 Jun 17 01:41 apcupsd.c

Fixes RedHat install and Makefile to make backup of files
to be changed during installation. Sorry about the mistake
RedHat users.

----> apcupsd-2.6.pre2 35282 Jun 17 01:41 apcupsd.c

More Pro Fixes and stuff for RedHat install and Makefile

----> apcupsd-2.6.pre1 35279 Jun 16 23:04 apcupsd.c

More Pro Fixes and stuff for RedHat install

----> apcupsd-2.5 36558 Jun 11 19:41 apcupsd.c

More Pro Fixes and stuff for RedHat install

----> apcupsd-2.5.pre4 35240 May 23 11:10 apcupsd.c

Can now re-init logfile at boot time with:
/{path}/apcupsd /dev/apcupsd newlog
This will help keep the log file from eating your root partition

940-0020B retested, fine
940-0095A untested -----!!BETA CODE!!-----

----> apcupsd-2.5.pre3 34829 May 22 17:56 apcupsd.c

major bug check....sorry
tested on smart and back ups, not pro yet
940-0020B not retested yet

APC UPS management under Linux

----> apcupsd-2.5.pre2 28712 May 21 20:39 apcupsd.c

New logging features.

----> apcupsd-2.5.pre

RedHat Support and Back UPS Pro Support via third party.

Christopher J. Reimer

----> apcupsd-2.4 24409 May 19 14:17 apcupsd.c

Back UPS Pro BETA Support

Karsten Wiborg

Christopher J. Reimer

----> apcupsd-2.3 21278 May 13 10:11 apcupsd.c

I made a patch to your apcupsd that allows it to be configured for external commands to run (instead of the built-in wall). This allows me to set it up to send out a call to my pager when the power fails, among other things.

Anyway, here it is.

--

Chris Adams - cadams@ro.com

not public release, yet.

----> apcupsd-2.2 21275 May 8 12:53 apcupsd.c

Fixed psuedo procfs type status file in the /etc directory.

Now works for both SMART and DUMB MODE.

Eric Raymond

Redit of apcupsd.manual.

----> apcupsd-2.1 20926 May 6 23:08 apcupsd.c

Jason Orendorf

"After building and installing the daemon init files I found out why you #define the term _ANNOY_ME...it is rather annoying to get 'wall'ed every 3 seconds - though I do think the feature is important. I made a trivial modification to a few files to make the time delay ('wall' frequency) configurable at build time. If you haven't made this change yourself, feel free to incorporate these changes into your development source if you think it would be useful to others."

Eric Raymond

Edited APC_UPSD.README.1ST to apcupsd.manual.

Many thanks.

Finally added psuedo procfs type status file in the /etc directory.

Broken for DUMB MODE

APC UPS management under Linux

----> apcupsd-2.0 20119 May 2 17:36 apcupsd.c

Name change to help sunsite.unc.edu "keeper" daemon.

Black Cables #940-0024B and #940-0024C are now supported for APC SmartUPS. They have been tested on an APC SmartUPS SU700/1400RM. I don't know if it will work on APC BackUPS Pro Series.....

Tested SmartUPS SU700RM with SNMP PowerNet(tm) card and SmartMode i/o through DB-9 port with 940-0024B and #940-0024C. This means that you can SNMP manage your SmartUPS with SNMP PowerNet(tm) card. You must have your own SNMP software.....

Added /proc/apcupsd file for pseudo procfs info.

Deleted announce aka -D_ANNOY_ME, for pseudo procfs info file.

Borrowed a bunch of code for Smart-UPS from Pavel Korensky's apcd.c apcd.h.

----> Enhanced-APC-UPS v1.9 April 17, 1997

Name change to help sunsite.unc.edu "keeper" daemon.

----> Enhanced-APC-UPS v1.9 Apr 14 15:02 apc_upsd.c

bug found by Tom Kunicki

"I still can cause the UPS to shutoff with a power disturbance by typing 'apc_upsd /dev/apc_ups killpower' before the disturbance." ... "Maybe you could disable the 'killpower' switch except for when the UPS is in the condition where it is supposed to be used..."

I have tested my BackUPS cable on the following and it works as designed:

APC BackUPS 400, APC BackUPS 600, and APC SmartUPS SU1400RM.

I have tested APC's cable #940-0020B on the following and it works as designed:

APC BackUPS 400, APC BackUPS 600, and APC SmartUPS SU1400RM.

I have NOT tested a BackUPS Pro, since I don't have one yet.

NOTE: My SmartUPS CABLE has broken code and more likely a broken cable design.

Black Cables #940-0024B and #940-0024C are still in the works but not high priority, yet.

----> Enhanced-APC-UPS v1.8 Apr 8 00:06 apc_upsd.c

Corrected defines in code for 940-0020B, 940-0023A, and 940-0095A.

-D_SMARTUPS can no longer be define if you select USE_APC=yes

There is no SmartUPS "Smart Mode" support, it has never been available.

I am waiting to hear from APC about pinouts for the following cables

#940-0023A their Unix OS cable and

#940-0095A their Win95OS cable.

If you want to use and APC cable, you must have cable #940-0020B ONLY....

Deleted Enhanced-APC-BackUPS.tar.gz from distribution.

Deleted rc.apc_power, all cables use rc.power.

Not publically released at sunsite.unc.edu

----> Enhanced-APC-UPS v1.7g-fix Apr 3 17:42 Makefile

Error in Makefile, rc.apc_power is for testing APC SmartUPS "black" cable.

All installs should use rc.power only, I forgot to make this change in the distribution Makefile from the Master Makefile.

----> Enhanced-APC-UPS v1.7g Apr 1 15:31 apc_upsd.c

APC UPS management under Linux

Possible FULL support for Grey cable APC# 940-0020B for BackUPS only
There still may be a bug like the original Enhanced_APC_BackUPS v1.0,
but has not shown it face yet after FIVE (5) forced power outages.

NO SUPPORT YET: Black cable APC# 940-0024B for SmartUPS BackUPS PRO only
This is take MUCH longer.....I don't have either SmartUPS BackUPS PRO to
test with, but can simulate the effect of PIN#8 on a BackUPS 400 and 600.

----> Enhanced_APC_UPS v1.6b Mar 31 22:03 apc_upsd.c

Added USE_APC in CONFIG for using APC cables
Added for using APC cables rc.apc_power
Basic support like original three-wire daemon.

Black cable APC# 940-0024B for SmartUPS and BackUPS PRO only
Grey cable APC# 940-0020B for BackUPS only

VERY BETA.....REPEAT, VERY BETA!!!!, ALMOST ALPHA.....

Not publically released at sunsite.unc.edu

----> Enhanced_APC_UPS v1.5 Mar 31 12:07 apc_upsd.c

Fixed a possible bug on the SmartUPS side of the daemon.
Retested cable design for "Andreas vasquez Mack"
Soon to test "black APC cable" for "Kofi N. Agyemang" and
"Pete Rylko"
APC Black Cable Support soon v1.6?

----> Enhanced_APC_UPS v1.4 Mar 1 17:18 apc_upsd.c

Added ANNOUNCE defeat in CONFIG

----> Enhanced_APC_UPS v1.3 Mar 1 16:41 apc_upsd.c

Fixed bug with momentary power outage less than 10 seconds with announce.
That darn announce.....
Added original Enhanced_APC_BackUPS package to distribution.
Never released.....

----> Enhanced_APC_UPS v1.2 Mar 1 15:46 apc_upsd.c

Fixed a loop order bug.
It now release the power problems message after power returns.

----> Enhanced_APC_UPS v1.1 Feb 28 16:25 apc_upsd.c

Copied new dowall.c from Sysvinit-2.69.
Added support for SmartUPS (non-Smart Accessories Port).
Fixed the above problem of repeated power failures.
Added cable design for SmartUPS (non-Smart Accessories Port).

----> Enhanced_APC_BackUPS v1.0 Sep 20 01:52 backupsd.c

Known Bug in special power situations. IFF (if and only if) the following
happens will this daemon fail to do its job.

If your power fails long enough to cause "backupsd" to activate and cleanly
shutdown your Linux-Box, and returns later only briefly.

APC UPS management under Linux

```
( That is long enough for the UPS to clear the KILL_POWER_BIT and causes )  
( your Linux-Box to reboot. But not long enough to recharge you UPS's    )  
( battery above the low-battery latch state, and power fails again. The  )  
( daemon cannot currently catch the status change.                        )
```

I have never had this happen to me ever, but I can force this to happen under laboratory test conditions. You can too, but it is not advised. Your filesystem will say " !@#\$%#\$*and much more to you.

Thanks for the support and use of this daemon.

hedrick@astro.dyer.vanderbilt.edu
Andre Hedrick

<http://www.brisse.dk/site/apcupsd/>



New Features in Apcupsd 3.8.0

· Highlights of this release.

- . In master/slave configurations, all masters and slaves must be updated at the same time. Version 3.8.0 is not compatible with versions prior to 3.8.0-pre4.
- . An implementation of apcupsd under Win32 has been completed.
- . Alpha Tru64 port.
- . Support for the Back-UPS Office series of UPSes.
- . Fixed **dumb** UPSes working with the Custom Simple cable.
- . Master/slave code works with 64 bit machines.
- . Correction of a buffer overflow on certain newer UPSes.
- . More fault tolerant on startup in master/slave configuration.
- . Additional STATUS variables (number of times on battery, amount of time on batteries, time/date of last transfer to batteries, apcupsd start time/date).
- . Proper support for older (pre 1994) UPSes in slave configurations.
- . Tighten file permissions for improved security.
- . apctest program for testing serial ports.
- . Automatic notification of certain events by email.
- . Strip executables during make install.
- . A lot of additions to the online manual.
- . New --with and --enable options in configuration. See `./configure --help`

The software is completely developed under Linux and will compile cleanly and will work under Linux. Porting is being addressed for Solaris and HP-UX, where the actual code should compile cleanly. If you own an APC UPS attached to a Linux machine or you want to realize a setup like this, or you are already an user of older version of **apcupsd**, please upgrade to the latest version. We need testers and any bug reports will be more than welcome. ·

What to do if you find bugs :

send an email to apcupsd-devel@apcupsd.org (Developers mailing list) or go to one of the following sites:

<http://www.apcupsd.org>

<http://www.sibbald.com/apcupsd>

Change Log

```

/*****
/*                      ChangeLog of apcupsd                      */
/*      Riccardo Facchetti <riccardo@master.oasi.gpa.it>          */
/*      http://www.apcupsd.org                                     */
/*      http://www.sibbald.com/apcupsd                           */
/*      ftp://ftp.apcupsd.org/pub/apcupsd                         */
/*****

```

----> Released apcupsd-3.8.0-pre4

Applied Kern's diff, see techlogs/kes06Oct00

Key points: apcaction.c fix for timer starting during power failure,
install key event handlers by default that email the
following events: changeme, commfailure, commok, mainsback,
and onbattery.

----> Released apcupsd-3.8.0-pre3

APC UPS management under Linux

Applied Kern's diff, see techlogs/kes05Oct00
Key points: fixes to Win32 binary release, manual updates,
TIMEOUT bug fix, handle pre-1994 UPSes correctly,
portability issues.

----> Released winapcupsd-3.8.0-pre2 Win32 binarys only
Applied Kern's corrections to the Win32 binary files.
Fix to /win32/apccontrol.in
Add sh.exe to \apcupsd\bin directory.

----> Released apcupsd-3.8.0-pre-1.
Applied Kern diffs, see techlogs/kes27Sep00
Key points: Alpha Tru64 port, documentation updates,
additional retries in networking, new STATUS variables,
tighten permission of files, RedHat RPM spec, corrections
to event scripts.

----> Snapshot released 20000911.
Applied Kern diffs, see techlogs/kes10Sep00
Key points: Win32 enhancements, buffer overrun
fix, tightning of file permissions, manual updates.

----> Snapshot released 20000911.
Applied Kern diffs, see techlog/kes10Sep00

----> Snapshot released 20000910.
WIN32 (cygwin) enhancements (KES): win'ization and use of system tray.
Documentation update.
autoconf reorganization.

----> apcupsd-3.7.2 released.
true and false programs are no more hardcoded into the configure script
Reorganized some of the distributions/ Makefiles.
Added FreeBSD installation support.

----> Snapshot released 20000528
Now apcaccess.c::stat_print() output status to stdout instead of stderr
Reorganized the kill_* code in apcupsd.c::main()
Removed the kill_ups_power argument from apcnet.c::kill_net()
Moved all save_dumb_status() calls to apcserial.c::setup_serial()
Applied kern's patch for kill_net in apcupsd.c and some other
beautifications of the code.
Changed scripts/autoregen.sh to compile with all the options switched o
Added a web page of thanks to the html manual.
Moved apcproctitle.c to lib/proctitle.c. Use that source only if
there is not a system setproctitle(3).
Don't use 's' subflag to ar(1). It's unnecessary since ranlib is
being called, and more importantly not all ar implementations have it.
Install distribution-specific apcupsd.conf files if they exist.
Fixed some calls to open(2) that didn't specify file create modes.
Added OpenBSD distribution and minor source patches.
Modified Slackware distribution: Update README, patch instead of
replacing rc files, delete obsolete distribution files.
Updated "makediff" developers' tool.
Minor manual updates.
Updated again Developers file: we gained more *BSD help.
Added the '-p' option to cmdline: was missing.
Abort on killpower for slaves instead of simply exit, in apcupsd.c
Removed nologin information from apcnet.c because we don't need. All th
nologin file is managed by apcaction.c

APC UPS management under Linux

----> Snapshot released 20000511
New maintainer for Slackware and OpenBSD ports: updated Developers file.
Applied David patch for slackware
Applied kern diffs, see techlogs/kes26Jan00
po/POTFILES file is now removed by main Makefile since is always built.
Corrected Slackware distribution Makefile.in: it was tied to
/usr/src. Thanks to John McSwain.
Corrected a bug in configure.in and include/apc_defines.h in which
the nologin file was created in /etc/apcupsd/nologin instead of
/etc/nologin.
Moved README.solaris to doc/.
Moved the patch "./1" to techlogs/kern-patch-17-Feb-2000

----> apcupsd-3.7.1 released 17 February 2000
Bumped version to 3.7.1 and released by Kern since Riccardo
is on vacation.
Update to upsbible.html to include additional credits.
Fixed dumb UPSes, which did not work in version 3.7.0,
see Kern's fixes techlogs/kes12Feb00

----> apcupsd-3.7.0 released.
Bumped version to 3.7.0 and released.
Powerflute: last 8 events are loaded from events file (if present).
Revamped powerflute: now curses work correctly.
Small fix to include/apc_config.h to clean Solaris compile.
Clean intl/ a bit better.
Fixed apcnetlib.c and apcnetd.c: now apcupsd compile under FreeBSD too.
Applied kern diffs, see techlogs/kes26Jan00
Fixed compile for Solaris 2.5.1.
Added check for snprintf when CGI is configured in configure.in.
Fixed internationalization inclusion when msgfmt is not found.
Fixed warning in lib/getopt.c.
Fixed AC_PATH_PROGS in configure.in.
Verified clean compile under Linux and HP-UX.
Added libintl and nls in information at the end of configure.
Corrected stpcpy warning in intl/dcgettext.c.
Changed all `make -C', this fixes HP-UX compilation of additional modules
like internationalization and cgi support.
Fixed detection of HP-UX version.
In apcnetlib.c use memset instead of bzero.
In apcupsd.c don't use TIOCNOTTY if not defined.
Corrected a macro bug in arguments of setpgrp() call in apcupsd.

----> apcupsd-3.7.0-rc1 released.
Updated my e-mail and (C) signatures: if you want to make it simple,
use vim and the scripts you can find in scripts/.
Restructured include/apc.h inclusion order: now make more sense.
Corrected setpgrp() call in apcupsd.c for BSD/non-BSD.
Added FreeBSD to configure autodetection and placeholder in distributions/.
Other minor fixes to Makefile.in and cgi/Makefile.in
Fixed make clean for cgi.
Applied kern diffs, see techlogs/kes17Jan00
Added slackware scripts from John McSwain.
Added Kern's fixes to distributions/redhat/awkhaltprog.in
Applied kern diffs, see techlogs/kes16Jan00
Added handling of `exit 0' at the end of halt.local as in suse 5.x.
Added autodetection for SuSE 5.x.
SuSE installation: written a shell script for installing apcupsd
directives in /etc/rc.config.

----> apcupsd-3.7.0-beta4 released.

APC UPS management under Linux

Bumped version to beta4 and released.
Fixed a minor po/ problem.
Applied kern diffs, see techlogs/kes13Jan00
Applied kern diffs, see techlogs/kes12Jan00
Fixed CYGWIN detection in configure.in.
Added clean and distclean for distributions/.
Added slackware detection but still manual installation (scripts generated but no actual installation performed: too dangerous doing the code without being able to test).

----> apcupsd-3.7.0-beta3 released.
Bumped version to beta3 and released.
Applied kern diffs, see techlogs/kes09Jan00
Changed clean_threads() in apcexec.c not hang on waitpid, and make sure we do all we can to kill the childs.
Updated INSTALL file to make clear that prior to install a new version of apcupsd over an old version, is advisable to make uninstall.
Corrected a syntax error in distributions/*/apccontrol.sh.in.
Changed suse halt.local script to:
 . do something only if a powerdown is detected.
 . kill the processes before remounting read only the filesystems.
Corrected a minor cosmetic (-Wall) in apcnetd.c
Added -Wall to default CFLAGS.
Applied kern diffs, see techlogs/kes06Jan00
Kern minor changes to configure.in.
Thanks to Tom Schroll, corrected an execv nasty error in apcexec.c.
Applied kern diffs, see techlogs/kes30Dec99

----> apcupsd-3.7.0-beta2 released.
Bumped version to beta2 and released.
Changed install-apcupsd target in main Makefile.in not to overwrite old apcupsd.conf file and instead create an apcupsd.conf.new file.
Applied Kern diffs, see techlogs/kes19Dec99
Changed the install to warn the user that if an old apcupsd.conf is in place, it is saved to apcupsd.conf.old.
Applied Kern diffs, see techlogs/kes18Dec99.
Reorganized the path construction in include/apc_defines.h so that now file paths are built with autoconf variables and not hardcoded.
Applied Carl Erhorn patches for Solaris, see techlogs/cpe16Dec99.
Now SuSE 5.2 is correctly detected (don't know previous versions).
Cleanups of configure.in.
Corrected a problem with SuSE and halt scripts where killpower where not issued correctly.
Applied Helmut Messerer corrections.
Applied Kern diffs, see techlogs/kes09Dec99
Applied Kern diffs, see techlogs/kes08Dec99
Added a new contrib/ directory where user contributed files are put.
New user contribution for sending sms messages on UPS troubles.

----> apcupsd-3.7.0-beta1 released.
Applied Kern diffs, see techlogs/kes30Nov99
Applied Kern diffs, see techlogs/kes28Nov99
Applied Kern diffs, see techlogs/kes20Nov99
Applied Kern diffs, see techlogs/kes18Nov99
Applied Kern diffs, see techlogs/kes15Nov99
More internationalization.
Applied Kern diffs, see techlogs/kes13Nov99
Moved TIOCM_LE HP-UX define and friends to include/apc_config.h
Internationalization: internationalized and translated in Italian all the messages that are printed with printf and fprintf: more need to be done for error_* functions.

APC UPS management under Linux

Removed #ifdef wrapping around debug code and substituted with debug_level checks.
Cleaned a remaining spit of 'killpower' in apcupsd.c
Help screen output to stdout.
Now (C) and Brian's Support Center are visible in help screen output.
Updates to the cgi files from Kern.
Added a vimrc for TABs.
--debug argument changed meaning. Now is meant to be set to a number that range from 0 to N where increasing numbers means increasing debug output.
Now we use getopt_long [see getopt(3)] for parsing command line.
Documentation updates and reorganization: now 'developers' documentation is in doc/developers/.
Added support for cable 940-0095B.
Applied Kern patches: syslog, getline, code cleanups, new cgi interface. Read his technical log in techlogs/kes03Nov99
Added doc/CodingStyle file.
Added Developers file.
Some minor code documentation.
Removed the "failed to reacquire the lockfile" problem. Now we release the lockfile just before fork() and reacquire it just after.
More duplicated code cleanups and nasty bugs fixed in apcaction.c.
SuSE's apcupsd start script now return green "done" and red "failed" (SuSE 6.2)
apccontrol is installed in /etc/apcupsd/ because we need it when filesystems may be umounted.
Removed powersc script: now it is all done in apccontrol and /etc/rc.d/apcupsd scripts.
Moved _all_ the scripts into scripts/ directory (where they belong).
apccontrol script installation dir is now sysconfdir
\${prefix}/etc/apcupsd
Simplified apcaction.c (removed duplicate code).
Attempt to document some features of apcupsd.conf.
Deleted two alarm()s related to apcreports.c from apcaction.c: this was a bug.
Added setproctitle for setting forked procs's argv[0]. Now we have
apcmain -> waiting for other tasks to exit, generic watchdog.
apcser -> serial task
apcact -> actions task
apcnet -> network task
apcslv -> netslave task
so that now we can tell with a 'ps' which task is doing what.
apcserial+apcreports are now one single thread.
Better locking scheme.
Removed another nasty bug in Old* variables into apcaction.c. Now there is a local structure for these values.
Removed a nasty bug in apcsetup.c.
Updated man pages and apcupsd.conf.
RedHat installation scripts.
Removed 10 seconds sleep() from terminate().
SHM ID for sanity checks on SHM accesses.
Obsolete config options now generate only warnings.
New error handling routines.
doc/README.developers updated.
Syslogging functions are bracketed with HAVE_GCC so that with gcc we can use macros and with other compilers we use functions for compatibility.
Syslog functions are now real functions. There's no point in having incompatible preprocessor macros when a set of little functions can do the job.
Fixed a next_slave label in apcnet.c with a semicolon for HP/UX compiler.

APC UPS management under Linux

OS detection in configure: now Makefiles and sources know about which OS they are supposed to compile for.
Better integration of lib/ sources into the configure mechanism.
configure cleanups.
Added cflags and ldflags selection in Makefile.in and configure.
Fixed getopt_long detection in configure. Now if not found it compile the lib/ version.
Now configure.in have a hardcoded PATH in which search the system programs needed.
Lot of cleanups in global variables and code partitioning and duplication.
More configure cleanups.
Updated RH 6.0 halt script.
Removed apchttp.
Cleaned THREADS. Now forked processes are the only option.
Various cleanups.
SuSE-specific install/uninstall.
All distribution specific directories are now in distributions/.
All scripts are now in scripts/.
Moved default apcupsd.conf in etc/.
Added distribution-specific scripts installation.
Install /etc/apcupsd.conf if not already present.
Removed apc(un)install.sh: no more needed.
Rewritten external scripts: no more system(), better customization support.
Removed Makefile.in.in from po/: it's not needed.
Fixed bogus --enable and --disable behavior of configure.
Cleaned up po/ and intl/ autoconf.
Only powerflute is linked with ncurses libraries.
The program makedepend is not any more vital for compilation.

----> apcupsd-3.6.2
Fixed the "apcupsd -c" configure command.
Fixed two thread bugs in alarm handling.
Fixed two potential security exploits.
More cleaned up autoconf.
Random documentation cleanups.

----> apcupsd-3.6.1
Cleaned up new autoconf stuff from version 3.6.0
Undefined a test flag for testing networks wierdness.
#undef WACKY_NETWORK_ATTEMPS
Network is fully functional under non-threaded compile.
Possible fix for "pipe_master_status" calls on slaves.
Added 940-1524C smart signal cable support.

----> apcupsd-3.6.0
Added autoconf.
Added internationalization support. There is _only_ the support but no current code is written for the intl package. It can be compiled in, but intl strings have still to be translated (to be done in the future).
Reorganized documentation.
Reorganized support for distributions. Now we have a directory for every possible distribution (suse, unixix, debian etc etc) so that the job for the package-men can be easier.

----> apcupsd-3.5.9
Added new configuration options to reduce init time of daemon.
powersc CONFIG
powersc NAME

APC UPS management under Linux

```
powersc BATTERY

----> apcupsd-3.5.8.patch

Fixes a FIFO error that I forgot to include in the rush
to release the code

----> apcupsd-3.5.8

GPL2 source code status finally.....April 7, 1999

Threaded code is stable but requires glibc2 or libc6
Finished SELFTEST setup
Fixed naming of UPS if allowed.
Redesigned "apcsetup.c" to allow for ease of parameter setup.

----> apcupsd-3.5.7

Added 940-0024G cable.
"newbackupspro" to be replaced by "backupspropnp"

If you have a BackUPS Pro UPS that is identified as
BP(SIZE)(EXTRAS) examples BP420SC/BP650SC/BP650PRO-PnP,
then you have a PnP BackUPS Pro that is very near a SmartUPS.
Else anything with the identifier BK(SIZE)PRO or BKPRO(SIZE)
is the very early version of the "dumbed down smartups".
Unfortunately, it has very little to say. It is superior to the
classic BackUPS (simple signals) in that it can at least tell event
histories and if the batteries are faulted.

now powerflute is compiling and working, needs THREADS

----> apcupsd-3.5.6

See Makefile for enabling flags.

# New multi-threading code BETA
#
# THREADS      = 1

# New multi-threading code with http BETA functional needs THREADS
#
# HTTP         = 1

# New/Old powerflute tool
#
# MUSIC        = 1

dual building model possible complete.
intergrated beta http data
breakages in logs and procfs are still present in shm.
code sorting and corrections.
return of ncurses powerflute tool.
fixed Makefile to find two possible locations of
ncurses package; however, there are three methods for this animal.

apcconfig.c: removed check for existance of lock directory. It's not
needed and dangerous when -killpower !
apcconfig.c: new configuration checks
added semun definition to apc_struct.h since it is needed for glibc6
```

APC UPS management under Linux

----> apcupsd-3.5.5

dual building model begun.
intergrated beta shared memory mapping.
breakages in logs and procfs are present.

----> apcupsd-3.5.4

Preparation for GPL status and transfer to GNU.
Fixed Network bug.
Fixed IPC PIPE bug.

----> apcupsd-3.5.3

Fixed UPS killpower bug.

----> apcupsd-3.5.2

Source wide reformat to conform to standard C programing format

----> apcupsd-3.5.1

----> apcupsd-3.5.0

fixes "NO" reports in setup loops for UPSes that do not allow for
changing parameters at initialization. Previously it was assumed that
UPSes did not report anything if the feature was not pollable and/or
changeable.

----> apcupsd-3.4.9

fixes a long missed error that was incorrectly fixed in the past.

----> apcupsd-3.4.8

solved more mystery functions and the UPSlink language may
be completely decoded.

----> apcupsd-3.4.7

now auto-learns features based on UPS's answer to questions.
initial porting to Solaris for i386 is complete.

----> apcupsd-3.4.6 45493 Jul 15 13:14 apcupsd.c
22326 Jul 9 12:50 apcnet.c
12129 Jul 9 12:50 apcpipes.c
25634 Jul 9 12:50 apcconfig.c
8565 Jul 15 13:07 apcsetup.c
11039 Jul 15 13:09 apcreports.c
18137 Jul 15 13:19 apcsmart.c

all logging functions will be moved to "apcreports.c"
all smart mode calls will be moved to "apcsmart.c"
fix ups model reporting in "apcsetup.c"

----> apcupsd-3.4.5 77096 Jul 7 17:18 apcupsd.c
22326 Jul 7 03:12 apcnet.c
12129 Jun 22 15:12 apcpipes.c
25634 Jul 7 16:39 apcconfig.c

APC UPS management under Linux

18477 Jul 7 02:08 apcaccess.c

Added NOLOGON delay for systems with large Matrix UPSes.
Short information polling, with constant values being set
in the extended setup functions.

----> apcupsd-3.4.4

Fixed a missing and needed wait delay for netmaster systems.
This was discovered when mixing flavors of Linux.
Since SuSE, RHS, and Debain require extra time for shutdowns.
man-page is closer to date.....

----> apcupsd-3.4.3

Added new limit features by polling a new command for
internal calculated remaining time on line.
Fixed naming UPS if allowed.

----> apcupsd-3.4.2

Fixed an unknown error for the forced ups-kill for the
backupspro models. This was discovered with a new
BackUPS Pro 1000.

----> apcupsd-3.4.1

Changed ./includes to have an "apc_" prefix.
This is for initial port requirements to FreeBSD.

----> apcupsd-3.4.0 72305 May 19 14:18 apcupsd.c
21597 Apr 16 17:59 apcnet.c
12117 Apr 16 15:44 apcpipes.c
22832 Apr 16 15:42 apcconfig.c

17687 Apr 16 13:47 apcaccess.c

Slave management disconnect/reconnect added to apcaccess.
Fixed excessive loss of UPS communications loggings of ::
 UPSlink Comm. Error, SM != SM
 UPSlink Comm. reestablished, SM == SM

Fixed Signal/Cable Combination errors.

A "SmartUPS" with a "Simple Cable" with report as a "BackUPS".

----> apcupsd-3.3.0 72450 Mar 20 01:09 apcupsd.c
15321 Mar 20 01:08 apcnet.c
11093 Mar 20 01:09 apcpipes.c
22465 Mar 20 01:09 config.c

17300 Mar 20 01:08 apcaccess.c

Network death solved (hopefully)
PowerFlute removed due to unexplainable daemon kills under 2.0.X. and
some cases of 2.1.X. (TCPIP)

This is replaced by the original tool that now works "apcaccess".
"apcaccess" is functional under both 2.0.X and 2.1.X kernels without
killing the daemon "apcupsd".

EPROM programming of many UPS models is now functional.

Example::

apcaccess : polling apcupsd for status.

APC UPS management under Linux

```
APC      : Mar 20 01:24:56
CABLE    : APC Cable 940-0024B
UPSMODEL : SmartUPS
UPSMODE   : Stand Alone
ULINE     : 124.0 Volts
MLINE     : 125.2 Volts
NLINE     : 124.0 Volts
FLINE     : 60.0 Hz
VOUTP     : 124.0 Volts
LOUTP     : 028.0
BOUTP     : 27.3 Volts
BCHAR     : 100.0
BFAIL     : 0x08
SENSE     : HIGH
WAKEUP    : 060 Cycles
SLEEP     : 020 Cycles
LOTRANS   : 103.0 Volts
HITRANS   : 129.0 Volts
CHARGE    : 025.0 Percent
UTEMP     : 49.5 C Internal
ALARM     : Low Batt + 30
DIPSW     : 0x0000
```

```
root@Orion% cat apcupsd.status
```

```
APC      : Mar 20 01:27:31
CABLE    : APC Cable 940-0024B
UPSMODEL : SmartUPS
UPSMODE   : Stand Alone
UPSNAME   : UPS_IDEN
ULINE     : 124.0 Volts
MLINE     : 124.6 Volts
NLINE     : 124.0 Volts
FLINE     : 60.0 Hz
VOUTP     : 124.0 Volts
LOUTP     : 028.0 Load Capacity
BOUTP     : 27.3 Volts
BCHAR     : 100.0 Batt. Charge
BFAIL     : 0x08 Status Flag
SENSE     : HIGH
WAKEUP    : 060 Cycles
SLEEP     : 020 Cycles
LOTRANS   : 103.0 Volts
HITRANS   : 129.0 Volts
CHARGE    : 025.0 Percent
UTEMP     : 49.5 C Internal
ALARM     : Low Batt
DIPSW     : 0x0000
```

```
----> apcupsd-3.2.2 66556 Jan 27 15:24 apcupsd.c
```

Fixed a reporting error in /etc/apcupsd.status for CUSTOM SIMPLE cable.
Fixed a reporting error in /var/log/apcupsd.log for CUSTOM SIMPLE cable.

```
----> apcupsd-3.2.1 67450 Jan 8 13:41 apcupsd.c
```

Fixed INSTALL from "readhat" to "redhat"
Fixed INSTALL errors of "slackware"

Bug search that causes network death, lost slaves under 2.0.X only??
Bug search that causes text-powerflute to kill daemon only under 2.0.X.

APC UPS management under Linux

```
----> apcupsd-3.2.0 66096 Dec 18 16:11 apcupsd.c
apcflute.c all code for reconfiguration complete.

----> apcupsd-3.1.0 65925 Dec  8 21:18 apcupsd.c

----> apcupsd-3.0.0

Now binaries only.....

----> apcupsd-2.9.9 63431 Dec  3 13:05 apcupsd.c

Fixed "for (killcount=0;killcount>3;killcount++)" bug
      ^ killcount.

----> apcupsd-2.9.8 62158 Nov 25 15:13 apcupsd.c

Added user defined magic and security timeouts.
Started powerflute update running master or standalone daemon.
Added another field for SmartUPS procfs file.

----> apcupsd-2.9.7 60945 Nov 18 19:19 apcupsd.c

Added second alternate method for killing power for non SU SmartUPS.
I have one of these older models, and it needs a bigger kick.

Disable lockfiles if slave is a netslave with (ups->cable == APC_NET),
or ethernet.

----> apcupsd-2.9.6 60156 Nov 10 22:34 apcupsd.c

split source files.....

10885 Nov 10 20:40 apcnet.c           :: NetUPS
18805 Nov 10 22:37 config.c          :: configuration

5924 Nov 10 19:54 apcflute.c.src     :: powerflute interface to apcupsd
5264 Nov 10 12:43 powerflute.c.src   :: powerflute tcp management of apcupsd

Brian Schau added lockfiles for serial ports.

NetUPS or EtherUPS now works.
Needs more security likely, you can set your own TCP port number.
Will add user MAGIC at compile in the future.

----> apcupsd-2.9.5 72431 Oct 28 00:27 apcupsd.c

Listing my hack of "powerflute". To enable it edit your /etc/apcupsd.conf
and set NETTIME to any value seconds is something to play around.

Reformat subroutines to get apcupsd to handle networks.

----> apcupsd-2.9.4 70707 Oct 25 13:51 apcupsd.c

Smart V/S is in the same class as a BackUPS Pro.

Riccardo Facchetti has been volunteered to solve the
network part for the project. YEAH!!!!!!!!!!!!!!

Hello Andre,
```

APC UPS management under Linux

I own a SmartUPS v/s and I have played a bit with your apcupsd. I have found that the SmartUPS v/s command set is similar (if not equal) to the one of BackUPS PRO, so I have changed the apcupsd.c code to behave this way and seems to work well.

Another thing: I would like to have an application like powerchute. Of course I am willing to write it, so I will call it powerflute.

My idea is:

```
apcupsd -> daemon
- controls power status
- listen on a TCP socket for incoming connections
  - get a command from TCP line
  - send the response to the command
```

```
powerflute -> application
- connect to apcupsd
  - sends a command to apcupsd
  - get the response from apcupsd
```

apcupsd is listening to the socket with a select so it don't block. After one or more connections are established, it read non-blocking from the TCP file descriptor(s) and when a char is available, send it to UPS and readline(), then send the buffer (terminated with '\n' to the TCP client over the connection file descriptor.

You can note that I have protected apcupsd UPS monitoring from TCP servicing by doing only non-blocking calls to the TCP layer, so there should be no problem for it in detecting power problems without delay caused by TCP connections.

A patch is enclosed for v/s == BKpro and for TCP servicing. Of course this patch is still very preliminar, but I think it is a good idea.

You can try the idea running the apcupsd with this patch applied, telnetting to the port 6669 from localhost (no external connections allowed now for security) and sending to the daemon some commands for the UPS. The daemon will redirect them to the UPS and send the answer to you. You can connect more than one client to the daemon (max 64 connections).

Please send me comments about it.

Ciao,

Riccardo.

--

Riccardo Facchetti | e-mail: riccardo@master.oasi.gpa.it

The TCP patch is not there but my hack job is present.
The package will have an addition called "powerflute".

----> apcupsd-2.9.3 67655 Oct 23 14:57 apcupsd.c

Fixed spelling of "dissable" to "disable"
Forgot to add this function for ShareUPS.

```
int fillShareUPS(int sharefd, UPSINFO *ups);
```

Brian Schau with needed install changes in Makefile

APC UPS management under Linux

for UNIFIX.

----> apcupsd-2.9.2 67410 Oct 22 19:34 apcupsd.c

Changed "case" names in /etc/apcupsd.conf, again.....
This should be the last time.....

ShareUPS project maybe nearing an end.....

make install is almost done.

----> apcupsd-2.9.1 65886 Oct 19 00:30 apcupsd.c

Changed "case" names in /etc/apcupsd.conf to lower case.
UPSTYPE SMARTUPS is now UPSTYPE smartups
Changed and add two new UPSMODE "cases" for ShareUPSes
Added Two new external call command for TIMEOUT and LOADLIMIT.
Shutdown types are now called by event not by a general shutdown.
Added CHANGEME to /sbin/powersc for SmartUPSes to announce that
its needs a batteries changed.

Tests are now run on (2) BackUPS 600's, SmartUPS Net 1400RM 700RM,
and SmartUPS 1250.

SmartUPSes now can call for shutdown based on percent of remaining
battery life or capacity.

Cleaned up manpage.....

----> apcupsd-2.9 58603 Oct 16 22:17 apcupsd.c

Changed to POSIX Style of control.
Now uses an external CONFIG file -> /etc/apcupsd.conf
All events are handled in the /sbin/powersc file, no more
sysvinit flavor hassels.....
Added ManPage, well an attempt.
Other things begun.....

----> apcupsd-2.8 42800 Oct 6 14:36 apcupsd.c

Finally solved the PLUG-N-PLAY Cable.....
Tested on a BackUPS 600 and SmartUPS 700RM

Someone needs to test a BackUPS Pro.....

If main power returns during a shutdown series, "apcupsd" will now reboot.

---->

Better fix and support for #940-0023A cable, but still broke, drat.

---->

Adds new code for UNIFIX Linux

----> apcupsd-2.7.1 35902 Sep 3 21:38 apcupsd.c

By Chris:

More changes and fixes with Pro Series tested.
Better method of "walling" the console.
Minor debugging added.

APC UPS management under Linux

ShareUPS Project Started, BETA.

----> apcupsd-2.7 35413 Aug 14 15:50 apcupsd.c

TESTED on the following setups:

SLACKWARE SMP/NON-SMP (2.1.49) SmartUPS 700 BackUPS 600

REDHAT 4.1 SMP/NON-SMP (2.0.30) BackUPS 600

SUSE 5.0 SMP/NON-SMP (2.0.30) BackUPS 600

NOTE BackUPS Pro Series Untested at this time of release.

All UPSes can be time limited shutdown.

Fixed all possible killpower problems.....

Changed rc.power to include a NOW command.

Changed inittab file and added

What to do when battery power fails.

pn::powerfailnow:/etc/rc.d/rc.power now

Made changes for SuSE distribution.

Change /sbin/init.d/powerfail file

Changes to CONFIG File

USE_SLACKWARE=yes

USE_REDHAT=no

USE_SUSE=no

Auto Updater

USE_MAKE_UPDATES=yes

USE_MAKE_INITTAB=yes

RedHat version for "powerstatus" file

REDHAT_VERSION=42

----> apcupsd-2.6 35718 Jul 10 08:10 apcupsd.c

Added more info to apcupsd.maunal about special external controls.

SEE "6) OTHER" in apcupsd.maunal

----> apcupsd-2.6.pre6 35718 Jul 10 08:10 apcupsd.c

Made changes in rc.power to become more universal.

Made N-th attempt to make use of APC cable #940-0095A "DRAT" in "dumb mode". Will now try "smart mode".

"DOUBLE DRAT" APC has a new cable #940-0095C.

They just want to make things hard for us.....

More Pro Fixes

S.u.S.e work around. This is a major DANGER!!!!

If you call /usr/sbin/apcupsd /dev/apcups killpower, in a non-power problem situation.

"YOU WILL SCREW YOURSELF" this safe guard is a direct override of the daemon.

----> apcupsd-2.6.pre5a 36279 Jun 23 16:39 apcupsd.c

APC UPS management under Linux

OOPS.....forgot an "endif" in the distribution Makefile

----> apcupsd-2.6.pre5 36279 Jun 23 16:39 apcupsd.c

Added Linux Flavor Dependencies in CONFIG file.
Fixed "nologin", so only root can login during a powerfailure.
All other attempts are defeated at the login prompt.

----> apcupsd-2.6.pre4 35616 Jun 19 15:40 apcupsd.c

Possible RedHat 4.2 fix.

----> apcupsd-2.6.pre3 35282 Jun 17 01:41 apcupsd.c

Fixes RedHat install and Makefile to make backup of files
to be changed during installation. Sorry about the mistake
RedHat users.

----> apcupsd-2.6.pre2 35282 Jun 17 01:41 apcupsd.c

More Pro Fixes and stuff for RedHat install and Makefile

----> apcupsd-2.6.pre1 35279 Jun 16 23:04 apcupsd.c

More Pro Fixes and stuff for RedHat install

----> apcupsd-2.5 36558 Jun 1 19:41 apcupsd.c

More Pro Fixes and stuff for RedHat install

----> apcupsd-2.5.pre4 35240 May 23 11:10 apcupsd.c

Can now re-init logfile at boot time with:
/{path}/apcupsd /dev/apcups newlog
This will help keep the log file from eating your root partition

940-0020B retested, fine
940-0095A untested -----!!BETA CODE!!-----

----> apcupsd-2.5.pre3 34829 May 22 17:56 apcupsd.c

major bug check....sorry
tested on smart and back ups, not pro yet
940-0020B not retested yet

----> apcupsd-2.5.pre2 28712 May 21 20:39 apcupsd.c

New logging features.

----> apcupsd-2.5.pre

RedHat Support and Back UPS Pro Support via third party.
Christopher J. Reimer

----> apcupsd-2.4 24409 May 19 14:17 apcupsd.c

Back UPS Pro BETA Support

Karsten Wiborg
Christopher J. Reimer

APC UPS management under Linux

----> apcupsd-2.3 21278 May 13 10:11 apcupsd.c

I made a patch to your apcupsd that allows it to be configured for external commands to run (instead of the built-in wall). This allows me to set it up to send out a call to my pager when the power fails, among other things.

Anyway, here it is.

--

Chris Adams - cadams@ro.com

not public release, yet.

----> apcupsd-2.2 21275 May 8 12:53 apcupsd.c

Fixed psuedo procfs type status file in the /etc directory.
Now works for both SMART and DUMB MODE.

Eric Raymond
Redit of apcupsd.manual.

----> apcupsd-2.1 20926 May 6 23:08 apcupsd.c

Jason Orendorf

"After building and installing the daemon init files I found out why you #define the term _ANNOY_ME...it is rather annoying to get 'wall'ed every 3 seconds - though I do think the feature is important. I made a trivial modification to a few files to make the time delay ('wall' frequency) configurable at build time. If you haven't made this change yourself, feel free to incorporate these changes into your development source if you think it would be useful to others."

Eric Raymond

Edited APC_UPSD.README.1ST to apcupsd.manual.
Many thanks.

Finally added psuedo procfs type status file in the /etc directory.
Broken for DUMB MODE

----> apcupsd-2.0 20119 May 2 17:36 apcupsd.c

Name change to help sunsite.unc.edu "keeper" daemon.

Black Cables #940-0024B and #940-0024C are now supported for APC SmartUPS. They have been tested on an APC SmartUPS SU700/1400RM. I don't know if it will work on APC BackUPS Pro Series.....

Tested SmartUPS SU700RM with SMNP PowerNet(tm) card and SmartMode i/o through DB-9 port with 940-0024B and #940-0024C. This means that you can SMNP manage your SmartUPS with SMNP PowerNet(tm) card. You must have your own SMNP software.....

Added /proc/apcupsd file for pseudo procfs info.

Deleted announce aka -D_ANNOY_ME, for pseudo procfs info file.

Borrowed a bunch of code for Smart-UPS from Pavel Korensky's apcd.c apcd.h.

----> Enhanced-APC-UPS v1.9 April 17, 1997

APC UPS management under Linux

Name change to help sunsite.unc.edu "keeper" daemon.

----> Enhanced_APC_UPS v1.9 Apr 14 15:02 apc_upsd.c

bug found by Tom Kunicki

"I still can cause the UPS to shutoff with a power disturbance by typing 'apc_upsd /dev/apc_ups killpower' before the disturbance." ... "Maybe you could disable the 'killpower' switch except for when the UPS is in the condition where it is supposed to be used..."

I have tested my BackUPS cable on the following and it works as designed:
APC BackUPS 400, APC BackUPS 600, and APC SmartUPS SU1400RM.
I have tested APC's cable #940-0020B on the following and it works as designed:
APC BackUPS 400, APC BackUPS 600, and APC SmartUPS SU1400RM.
I have NOT tested a BackUPS Pro, since I don't have one yet.

NOTE: My SmartUPS CABLE has broken code and more likely a broken cable design.

Black Cables #940-0024B and #940-0024C are still in the works but not high priority, yet.

----> Enhanced_APC_UPS v1.8 Apr 8 00:06 apc_upsd.c

Corrected defines in code for 940-0020B, 940-0023A, and 940-0095A.
-D_SMARTUPS can no longer be define if you select USE_APC=yes
There is no SmartUPS "Smart Mode" support, it has never been available.
I am waiting to hear from APC about pinouts for the following cables
#940-0023A their Unix OS cable and
#940-0095A their Win95OS cable.
If you want to use and APC cable, you must have cable #940-0020B ONLY....

Deleted Enhanced_APC_BackUPS.tar.gz from distribution.
Deleted rc.apc_power, all cables use rc.power.

Not publically released at sunsite.unc.edu

----> Enhanced_APC_UPS v1.7g-fix Apr 3 17:42 Makefile

Error in Makefile, rc.apc_power is for testing APC SmartUPS "black" cable.
All installs should use rc.power only, I forgot to make this change in the distribution Makefile from the Master Makefile.

----> Enhanced_APC_UPS v1.7g Apr 1 15:31 apc_upsd.c

Possible FULL support for Grey cable APC# 940-0020B for BackUPS only
There still may be a bug like the original Enhanced_APC_BackUPS v1.0, but has not shown it face yet after FIVE (5) forced power outages.

NO SUPPORT YET: Black cable APC# 940-0024B for SmartUPS BackUPS PRO only
This is take MUCH longer.....I don't have either SmartUPS BackUPS PRO to test with, but can simulate the effect of PIN#8 on a BackUPS 400 and 600.

----> Enhanced_APC_UPS v1.6b Mar 31 22:03 apc_upsd.c

Added USE_APC in CONFIG for using APC cables
Added for using APC cables rc.apc_power
Basic support like original three-wire daemon.

Black cable APC# 940-0024B for SmartUPS and BackUPS PRO only
Grey cable APC# 940-0020B for BackUPS only

APC UPS management under Linux

VERY BETA.....REPEAT, VERY BETA!!!!, ALMOST ALPHA.....

Not publically released at sunsite.unc.edu

----> Enhanced_APC_UPS v1.5 Mar 31 12:07 apc_upsd.c

Fixed a possible bug on the SmartUPS side of the daemon.
Retested cable design for "Andreas vasquez Mack"
Soon to test "black APC cable" for "Kofi N. Agyemang" and
"Pete Rylko"
APC Black Cable Support soon v1.6?

----> Enhanced_APC_UPS v1.4 Mar 1 17:18 apc_upsd.c

Added ANNOUNCE defeat in CONFIG

----> Enhanced_APC_UPS v1.3 Mar 1 16:41 apc_upsd.c

Fixed bug with momentary power outage less than 10 seconds with announce.
That darn announce.....
Added original Enhanced_APC_BackUPS package to distribution.
Never released.....

----> Enhanced_APC_UPS v1.2 Mar 1 15:46 apc_upsd.c

Fixed a loop order bug.
It now release the power problems message after power returns.

----> Enhanced_APC_UPS v1.1 Feb 28 16:25 apc_upsd.c

Copied new dowall.c from Sysvinit-2.69.
Added support for SmartUPS (non-Smart Accessories Port).
Fixed the above problem of repeated power failures.
Added cable design for SmartUPS (non-Smart Accessories Port).

----> Enhanced_APC_BackUPS v1.0 Sep 20 01:52 backupsd.c

Known Bug in special power situations. IFF (if and only if) the following happens will this daemon fail to do its job.

If your power fails long enough to cause "backupsd" to activate and cleanly shutdown your Linux-Box, and returns later only briefly.

(That is long enough for the UPS to clear the KILL_POWER_BIT and causes)
(your Linux-Box to reboot. But not long enough to recharge you UPS's)
(battery above the low-battery latch state, and power fails again. The)
(daemon cannot currently catch the status change.)

I have never had this happen to me ever, but I can force this to happen under laboratory test conditions. You can too, but it is not advised. Your filesystem will say " !@#%#\$*and much more to you.

Thanks for the support and use of this daemon.

hedrick@astro.dyer.vanderbilt.edu
Andre Hedrick

<http://www.brisse.dk/site/apcupsd/>



Upgrading to Apcupsd 3.7.1

Quite a number of the configuration statements have changed between versions 3.6.2 and 3.7.1, so you should either take the new `apcupsd.conf` file and modify it, or update your existing file. In general, we recommend starting with the new file.

If you have used a prior version of `apcupsd`, the `CONTROL` script file (`/sbin/powersec`) has now been replaced by `/etc/apcupsd/apccontrol`. Consequently, the `CONTROL` configuration statement is obsolete. The following configuration statements have been replaced by scripts called from `/etc/apcupsd/apccontrol`, and thus are obsolete: `BATTCMD`, `LIMITCMN`, `LOADCMD`, `PWRCMD`, `REBOOTCMD`, `REMOTECMD`, `RETCMD`, and `TIMECMD`.

On most systems the script that started and stopped **apcupsd** was called **apcups**. In versions 3.7.0 and later, this script has been renamed to **apcupsd**.

If you use the master/slave networking code, please be aware that the **Apcupsd 3.7.1** network protocol is not compatible with prior versions. Thus to use master/slave networking, you must upgrade your master and all the slaves at the same time.

Since there are now a number of script files and data files used by **apcupsd**, they have been moved from their previous locations to the directory `/etc/apcupsd`. The executable files (**apcupsd**, **apcaccess**, ...) remain in their prior locations (normally `/usr/sbin` or `/sbin` depending on your configuration).

With this release, there are four [Web CGI programs](#) (**multimon.cgi**, **upsstats.cgi**, **upsfstats.cgi**, and **upsimage.cgi**). To have them properly installed, you must run the `./configure` command with `--enable-cgi` and you must specify an installation directory with `--with-cgi-bin=` or load them manually.



New Features in Apcupsd 3.7.2

Version 3.7.2 is a bug fix release, and the changes primarily concerned the non-smart UPSes (sometimes referred to as dumb UPSes).

New Features in Apcupsd 3.7.1

This version of apcupsd, thanks to the efforts of Andre H. Hedrick, Brian Schau, Kern Sibbald and many others has been almost rewritten. Many thanks have to be addressed to APC people that not only have allowed, on April 7 1999, the release of apcupsd under the GPL license, they also offered their technical help to the developers. Thank you !

Apcupsd is a daemon for monitoring APC UPSes. It is able to detect power loss and shutdown the computer if the UPS battery power is getting too low. It is able to shutdown multiple computers powered by a single UPS by running one computer as the master and the others as slaves. The master communicates to the slaves sending them messages by the network.

· Highlights of this release.

- . New CGI interface to see the UPS status over the web
- . New network interface to publish to clients the UPS status
- . New master/server network code that is more fault tolerant
- . Internationalization
- . Use GNU getopt
- . Logging on syslog
- . Events logged in a file like APC's PowerChute
- . Sync code removed: now only async processes are used
- . Better automated installation
- . Updated documentation (HTTP version)
- . Bug fixes and enhancements: too many to tell them all

The software is completely developed under Linux and will compile cleanly and will work under Linux. Porting is being addressed for Solaris and HP-UX, where the actual code should compile cleanly. If you own an APC UPS attached to a Linux machine or you want to realize a setup like this, or you are already an user of older version of **apcupsd**, please upgrade to the latest version. We need testers and any bug reports will be more than welcome. ·

What to do if you find bugs :

send an e-mail to riccardo@master.oasi.gpa.it (Project Manager and main developer) or send an e-mail to apcupsd-devel@apcupsd.org (Developers mailing list) or go to one of the following sites:

<http://www.apcupsd.org>

<http://www.sibbald.com/apcupsd>

Apcupsd Support Center <http://www.brisse.dk/site/apcupsd/>

Change Log

```

/*****
/*          ChangeLog of apcupsd                      */
/*          Riccardo Facchetti <riccardo@master.oasi.gpa.it>      */
/*          http://www.apcupsd.org                          */
/*          http://www.sibbald.com/apcupsd                  */

```

APC UPS management under Linux

```
/*                      ftp://ftp.apcupsd.org/pub/apcupsd                      */
/*****

----> apcupsd-3.7.2 released.
      true and false programs are no more hardcoded into the configure script
      Reorganized some of the distributions/ Makefiles.
      Added FreeBSD installation support.

----> Snapshot released 20000528
      Now apcaccess.c::stat_print() output status to stdout instead of stderr
      Reorganized the kill_* code in apcupsd.c::main()
      Removed the kill_ups_power argument from apcnet.c::kill_net()
      Moved all save_dumb_status() calls to apcserial.c::setup_serial()
      Applied kern's patch for kill_net in apcupsd.c and some other
      beautifications of the code.
      Changed scripts/autoregen.sh to compile with all the options switched o
      Added a web page of thanks to the html manual.
      Moved apcproctitle.c to lib/proctitle.c. Use that source only if
      there is not a system setproctitle(3).
      Don't use 's' subflag to ar(1). It's unnecessary since ranlib is
      being called, and more importantly not all ar implementations have it.
      Install distribution-specific apcupsd.conf files if they exist.
      Fixed some calls to open(2) that didn't specify file create modes.
      Added OpenBSD distribution and minor source patches.
      Modified Slackware distribution: Update README, patch instead of
      replacing rc files, delete obsolete distribution files.
      Updated "makediff" developers' tool.
      Minor manual updates.
      Updated again Developers file: we gained more *BSD help.
      Added the '-p' option to cmdline: was missing.
      Abort on killpower for slaves instead of simply exit, in apcupsd.c
      Removed nologin information from apcnet.c because we don't need. All th
      nologin file is managed by apcaction.c

----> Snapshot released 20000511
      New mantainer for Slackware and OpenBSD ports: updated Developers file.
      Applied David patch for slackware
      Applied kern diffs, see techlogs/kes26Jan00
      po/POTFILES file is now removed by main Makefile since is always built.
      Corrected Slackware distribution Makefile.in: it was tied to
      /usr/src. Thanks to John McSwain.
      Corrected a bug in configure.in and include/apc_defines.h in which
      the nologin file was created in /etc/apcupsd/nologin instead of
      /etc/nologin.
      Moved README.solaris to doc/.
      Moved the patch "./1" to techlogs/kern-patch-17-Feb-2000

----> apcupsd-3.7.1 released 17 February 2000
      Bumped version to 3.7.1 and released by Kern since Riccardo
      is on vacation.
      Update to upsbible.html to include additional credits.
      Fixed dumb UPSes, which did not work in version 3.7.0,
      see Kern's fixes techlogs/kes12Feb00

----> apcupsd-3.7.0 released.
      Bumped version to 3.7.0 and released.
      Powerflute: last 8 events are loaded from events file (if present).
      Revamped powerflute: now curses work correctly.
      Small fix to include/apc_config.h to clean Solaris compile.
      Clean intl/ a bit better.
      Fixed apcnetlib.c and apcnetd.c: now apcupsd compile under FreeBSD too.
```

APC UPS management under Linux

Applied kern diffs, see techlogs/kes26Jan00
Fixed compile for Solaris 2.5.1.
Added check for snprintf when CGI is configured in configure.in.
Fixed internationalization inclusion when msgfmt is not found.
Fixed warning in lib/getopt.c.
Fixed AC_PATH_PROGS in configure.in.
Verified clean compile under Linux and HP-UX.
Added libintl and nls in information at the end of configure.
Corrected stpcpy warning in intl/dcgettext.c.
Changed all `make -C', this fixes HP-UX compilation of additional modules like internationalization and cgi support.
Fixed detection of UP-UX version.
In apcnetlib.c use memset instead of bzero.
In apcupsd.c don't use TIOCNOTTY if not defined.
Corrected a macro bug in arguments of setpgrp() call in apcupsd.

----> apcupsd-3.7.0-rc1 released.
Updated my e-mail and (C) signatures: if you want to make it simple, use vim and the scripts you can find in scripts/.
Restructured include/apc.h inclusion order: now make more sense.
Corrected setpgrp() call in apcupsd.c for BSD/non-BSD.
Added FreeBSD to configure autodetection and placeholder in distributions/.
Other minor fixes to Makefile.in and cgi/Makefile.in
Fixed make clean for cgi.
Applied kern diffs, see techlogs/kes17Jan00
Added slackware scripts from John McSwain.
Added Kern's fixes to distributions/redhat/awkhaltprog.in
Applied kern diffs, see techlogs/kes16Jan00
Added handling of `exit 0' at the end of halt.local as in suse 5.x.
Added autodetection for SuSE 5.x.
SuSE installation: written a shell script for installing apcupsd directives in /etc/rc.config.

----> apcupsd-3.7.0-beta4 released.
Bumped version to beta4 and released.
Fixed a minor po/ problem.
Applied kern diffs, see techlogs/kes13Jan00
Applied kern diffs, see techlogs/kes12Jan00
Fixed CYGWIN detection in configure.in.
Added clean and distclean for distributions/.
Added slackware detection but still manual installation (scripts generated but no actual installation performed: too dangerous doing the code without being able to test).

----> apcupsd-3.7.0-beta3 released.
Bumped version to beta3 and released.
Applied kern diffs, see techlogs/kes09Jan00
Changed clean_threads() in apcexec.c not hang on waitpid, and make sure we do all we can to kill the childs.
Updated INSTALL file to make clear that prior to install a new version of apcupsd over an old version, is advisable to make uninstall.
Corrected a syntax error in distributions/*/apccontrol.sh.in.
Changed suse halt.local script to:
 . do something only if a powerdown is detected.
 . kill the processes before remounting read only the filesystems.
Corrected a minor cosmetic (-Wall) in apcnetd.c
Added -Wall to default CFLAGS.
Applied kern diffs, see techlogs/kes06Jan00
Kern minor changes to configure.in.
Thanks to Tom Schroll, corrected an execv nasty error in apcexec.c.
Applied kern diffs, see techlogs/kes30Dec99

APC UPS management under Linux

----> apcupsd-3.7.0-beta2 released.

Bumped version to beta2 and released.
Changed install-apcupsd target in main Makefile.in not to overwrite old apcupsd.conf file and instead create an apcupsd.conf.new file.
Applied Kern diffs, see techlogs/kes19Dec99
Changed the install to warn the user that if an old apcupsd.conf is in place, it is saved to apcupsd.conf.old.
Applied Kern diffs, see techlogs/kes18Dec99.
Reorganized the path construction in include/apc_defines.h so that now file paths are built with autoconf variables and not hardcoded.
Applied Carl Erhorn patches for Solaris, see techlogs/cpe16Dec99.
Now SuSE 5.2 is correctly detected (don't know previous versions).
Cleanups of configure.in.
Corrected a problem with SuSE and halt scripts where killpower where not issued correctly.
Applied Helmut Messerer corrections.
Applied Kern diffs, see techlogs/kes09Dec99
Applied Kern diffs, see techlogs/kes08Dec99
Added a new contrib/ directory where user contributed files are put.
New user contribution for sending sms messages on UPS troubles.

----> apcupsd-3.7.0-beta1 released.

Applied Kern diffs, see techlogs/kes30Nov99
Applied Kern diffs, see techlogs/kes28Nov99
Applied Kern diffs, see techlogs/kes20Nov99
Applied Kern diffs, see techlogs/kes18Nov99
Applied Kern diffs, see techlogs/kes15Nov99
More internationalization.
Applied Kern diffs, see techlogs/kes13Nov99
Moved TIOCM_LE HP-UX define and friends to include/apc_config.h
Internationalization: internationalized and translated in Italian all the messages that are printed with printf and fprintf: more need to be done for error_* functions.
Removed #ifdef wrapping around debug code and substituted with debug_level checks.
Cleaned a remaining spit of 'killpower' in apcupsd.c
Help screen output to stdout.
Now (C) and Brian's Support Center are visible in help screen output.
Updates to the cgi files from Kern.
Added a vimrc for TABs.
--debug argument changed meaning. Now is meant to be set to a number that range from 0 to N where increasing numbers means increasing debug output.
Now we use getopt_long[see getopt(3)] for parsing command line.
Documentation updates and reorganization: now 'developers' documentation is in doc/developers/.
Added support for cable 940-0095B.
Applied Kern patches: syslog, getline, code cleanups, new cgi interface. Read his technical log in techlogs/kes03Nov99
Added doc/CodingStyle file.
Added Developers file.
Some minor code documentation.
Removed the "failed to reacquire the lockfile" problem. Now we release the lockfile just before fork() and reacquire it just after.
More duplicated code cleanups and nasty bugs fixed in apcaction.c.
SuSE's apcupsd start script now return green "done" and red "failed" (SuSE 6.2)
apcccontrol is installed in /etc/apcupsd/ because we need it when filesystems may be umounted.
Removed powersc script: now it is all done in apcccontrol and /etc/rc.d/apcupsd scripts.

APC UPS management under Linux

Moved `_all_` the scripts into `scripts/` directory (where they belong).
apccontrol script installation dir is now `sysconfdir`
`${prefix}/etc/apcupsd`
Simplified `apcaction.c` (removed duplicate code).
Attempt to document some features of `apcupsd.conf`.
Deleted two `alarm()`s related to `apcreports.c` from `apcaction.c`: this was a bug.
Added `setproctitle` for setting forked procs's `argv[0]`. Now we have
`apcmain` -> waiting for other tasks to exit, generic watchdog.
`apcser` -> serial task
`apcact` -> actions task
`apcnet` -> network task
`apcslv` -> netslave task
so that now we can tell with a ``ps'` which task is doing what.
`apcserial+apcreports` are now one single thread.
Better locking scheme.
Removed another nasty bug in `Old*` variables into `apcaction.c`. Now there is a local structure for these values.
Removed a nasty bug in `apcsetup.c`.
Updated man pages and `apcupsd.conf`.
RedHat installation scripts.
Removed 10 seconds `sleep()` from `terminate()`.
SHM ID for sanity checks on SHM accesses.
Obsolete config options now generate only warnings.
New error handling routines.
`doc/README.developers` updated.
Syslogging functions are bracketed with `HAVE_GCC` so that with gcc we can use macros and with other compilers we use functions for compatibility.
Syslog functions are now real functions. There's no point in having incompatible preprocessor macros when a set of little functions can do the job.
Fixed a `next_slave` label in `apcnet.c` with a semicolon for HPUX compiler.
OS detection in `configure`: now Makefiles and sources know about which OS they are supposed to compile for.
Better integration of `lib/` sources into the `configure` mechanism.
`configure` cleanups.
Added `cflags` and `ldflags` selection in `Makefile.in` and `configure`.
Fixed `getopt_long` detection in `configure`. Now if not found it compile the `lib/` version.
Now `configure.in` have a hardcoded `PATH` in which search the system programs needed.
Lot of cleanups in global variables and code partitioning and duplication.
More `configure` cleanups.
Updated RH 6.0 `halt` script.
Removed `apchttp`.
Cleaned `THREADS`. Now forked processes are the only option.
Various cleanups.
SuSE-specific `install/uninstall`.
All distribution specific directories are now in `distributions/`.
All scripts are now in `scripts/`.
Moved default `apcupsd.conf` in `etc/`.
Added distribution-specific scripts installation.
Install `/etc/apcupsd.conf` if not already present.
Removed `apc(un)install.sh`: no more needed.
Rewritten external scripts: no more `system()`, better customization support.
Removed `Makefile.in.in` from `po/`: it's not needed.
Fixed bogus `--enable` and `--disable` behavior of `configure`.

APC UPS management under Linux

Cleaned up po/ and intl/ autoconf.
Only powerflute is linked with ncurses libraries.
The program makedepend is not any more vital for compilation.

----> apcupsd-3.6.2
Fixed the "apcupsd -c" configure command.
Fixed two thread bugs in alarm handling.
Fixed two potential security exploits.
More cleaned up autoconf.
Random documentation cleanups.

----> apcupsd-3.6.1
Cleaned up new autoconf stuff from version 3.6.0
Undefined a test flag for testing networks wierdness.
 #undef WACKY_NETWORK_ATTEMPS
Network is fully functional under non-threaded compile.
Possible fix for "pipe_master_status" calls on slaves.
Added 940-1524C smart signal cable support.

----> apcupsd-3.6.0
Added autoconf.
Added internationalization support. There is _only_ the support but
no current code is written for the intl package. It can be compiled
in, but intl strings have still to be translated (to be done in the
future).
Reorganized documentation.
Reorganized support for distributions. Now we have a directory for
every possible distribution (suse, unix, debian etc etc) so that the
job for the package-men can be easier.

----> apcupsd-3.5.9
Added new configuration options to reduce init time of daemon.
 powersc CONFIG
 powersc NAME
 powersc BATTERY

----> apcupsd-3.5.8.patch

Fixes a FIFO error that I forgot to include in the rush
to release the code

----> apcupsd-3.5.8

GPL2 source code status finally.....April 7, 1999

Threaded code is stable but requires glibc2 or libc6
Finished SELFTEST setup
Fixed naming of UPS if allowed.
Redesigned "apcsetup.c" to allow for ease of parameter setup.

----> apcupsd-3.5.7

Added 940-0024G cable.
"newbackupspro" to be replaced by "backupspronp"

If you have a BackUPS Pro UPS that is identified as
BP(SIZE)(EXTRAS) examples BP420SC/BP650SC/BP650PRO-PnP,
then you have a PnP BackUPS Pro that is very near a SmartUPS.
Else anything with the identifier BK(SIZE)PRO or BKPRO(SIZE)
is the very early version of the "dumbed down smartups".
Unfortunately, it has very little to say. It is superior to the

APC UPS management under Linux

classic BackUPS (simple signals) in that it can at least tell event histories and if the batteries are faulted.

now powerflute is compiling and working, needs THREADS

----> apcupsd-3.5.6

See Makefile for enabling flags.

```
# New multi-threading code BETA
#
# THREADS    = 1
```

```
# New multi-threading code with http BETA functional needs THREADS
#
# HTTP       = 1
```

```
# New/Old powerflute tool
#
# MUSIC      = 1
```

dual building model possible complete.
intergrated beta http data
breakages in logs and procfs are still present in shm.
code sorting and corrections.
return of ncurses powerflute tool.
fixed Makefile to find two possible locations of
ncurses package; however, there are three methods for this animal.

apccconfig.c: removed check for existence of lock directory. It's not
needed and dangerous when -killpower !
apccconfig.c: new configuration checks
added semun definition to apc_struct.h since it is needed for glibc6

----> apcupsd-3.5.5

dual building model begun.
intergrated beta shared memory mapping.
breakages in logs and procfs are present.

----> apcupsd-3.5.4

Preparation for GPL status and transfer to GNU.
Fixed Network bug.
Fixed IPC PIPE bug.

----> apcupsd-3.5.3

Fixed UPS killpower bug.

----> apcupsd-3.5.2

Source wide reformat to conform to standard C programing format

----> apcupsd-3.5.1

----> apcupsd-3.5.0

fixes "NO" reports in setup loops for UPSes that do not allow for
changing parameters at initialization. Previously it was assumed that

APC UPS management under Linux

UPSes did not report anything if the feature was not pollable and/or changeable.

----> apcupsd-3.4.9

fixes a long missed error that was incorrectly fixed in the past.

----> apcupsd-3.4.8

solved more mystery functions and the UPSlink language may be completely decoded.

----> apcupsd-3.4.7

now auto-learns features based on UPS's answer to questions.
initial porting to Solaris for i386 is complete.

----> apcupsd-3.4.6

45493	Jul	15	13:14	apcupsd.c
22326	Jul	9	12:50	apcnet.c
12129	Jul	9	12:50	apcpipes.c
25634	Jul	9	12:50	apcconfig.c
8565	Jul	15	13:07	apcsetup.c
11039	Jul	15	13:09	apcreports.c
18137	Jul	15	13:19	apcsmart.c

all logging functions will be moved to "apcreports.c"
all smart mode calls will be moved to "apcsmart.c"
fix ups model reporting in "apcsetup.c"

----> apcupsd-3.4.5

77096	Jul	7	17:18	apcupsd.c
22326	Jul	7	03:12	apcnet.c
12129	Jun	22	15:12	apcpipes.c
25634	Jul	7	16:39	apcconfig.c
18477	Jul	7	02:08	apcaccess.c

Added NOLOGON delay for systems with large Matrix UPSes.
Short information polling, with constant values being set in the extended setup functions.

----> apcupsd-3.4.4

Fixed a missing and needed wait delay for netmaster systems.
This was discovered when mixing flavors of Linux.
Since SuSE, RHS, and Debain require extra time for shutdowns.
man-page is closer to date.....

----> apcupsd-3.4.3

Added new limit features by polling a new command for internal calculated remaining time on line.
Fixed naming UPS if allowed.

----> apcupsd-3.4.2

Fixed an unknown error for the forced ups-kill for the backupspro models. This was discovered with a new BackUPS Pro 1000.

----> apcupsd-3.4.1

Changed ./includes to have an "apc_" prefix.
This is for initial port requirements to FreeBSD.

----> apcupsd-3.4.0

72305	May	19	14:18	apcupsd.c
-------	-----	----	-------	-----------

APC UPS management under Linux

```
21597 Apr 16 17:59 apcnet.c
12117 Apr 16 15:44 apcpipes.c
22832 Apr 16 15:42 apcconfig.c

17687 Apr 16 13:47 apcaccess.c
```

Slave management disconnect/reconnect added to apcaccess.
Fixed excessive loss of UPS communications loggings of ::

```
    UPSlink Comm. Error, SM != SM
    UPSlink Comm. reestablished, SM == SM
```

Fixed Signal/Cable Combination errors.

A "SmartUPS" with a "Simple Cable" with report as a "BackUPS".

```
----> apcupsd-3.3.0 72450 Mar 20 01:09 apcupsd.c
        15321 Mar 20 01:08 apcnet.c
        11093 Mar 20 01:09 apcpipes.c
        22465 Mar 20 01:09 config.c

        17300 Mar 20 01:08 apcaccess.c
```

Network death solved (hopefully)

PowerFlute removed due to unexplainable daemon kills under 2.0.X. and
some cases of 2.1.X. (TCPIP)

This is replaced by the original tool that now works "apcaccess".
"apcaccess" is functional under both 2.0.X and 2.1.X kernels without
killing the daemon "apcupsd".

EPROM programming of many UPS models is now functional.

Example::

apcaccess : polling apcupsd for status.

```
APC      : Mar 20 01:24:56
CABLE    : APC Cable 940-0024B
UPSMODEL : SmartUPS
UPSMODE  : Stand Alone
ULINE    : 124.0 Volts
MLINE    : 125.2 Volts
NLINE    : 124.0 Volts
FLINE    : 60.0 Hz
VOUTP    : 124.0 Volts
LOUTP    : 028.0
BOUTP    : 27.3 Volts
BCHAR    : 100.0
BFAIL    : 0x08
SENSE    : HIGH
WAKEUP   : 060 Cycles
SLEEP    : 020 Cycles
LOTRANS  : 103.0 Volts
HITRANS  : 129.0 Volts
CHARGE   : 025.0 Percent
UTEMP    : 49.5 C Internal
ALARM    : Low Batt + 30
DIPSW    : 0x0000
```

root@Orion% cat apcupsd.status

```
APC      : Mar 20 01:27:31
CABLE    : APC Cable 940-0024B
```

APC UPS management under Linux

```
UPSMODEL : SmartUPS
UPSMODE  : Stand Alone
UPSNAME  : UPS_IDEN
ULINE    : 124.0 Volts
MLINE    : 124.6 Volts
NLINE    : 124.0 Volts
FLINE    : 60.0 Hz
VOUTP    : 124.0 Volts
LOUTP    : 028.0 Load Capacity
BOUTP    : 27.3 Volts
BCHAR    : 100.0 Batt. Charge
BFAIL    : 0x08 Status Flag
SENSE    : HIGH
WAKEUP   : 060 Cycles
SLEEP    : 020 Cycles
LOTRANS  : 103.0 Volts
HITRANS  : 129.0 Volts
CHARGE   : 025.0 Percent
UTEMP    : 49.5 C Internal
ALARM    : Low Batt
DIPSW    : 0x0000
```

----> apcupsd-3.2.2 66556 Jan 27 15:24 apcupsd.c

Fixed a reporting error in /etc/apcupsd.status for CUSTOM SIMPLE cable.
Fixed a reporting error in /var/log/apcupsd.log for CUSTOM SIMPLE cable.

----> apcupsd-3.2.1 67450 Jan 8 13:41 apcupsd.c

Fixed INSTALL from "readhat" to "redhat"
Fixed INSTALL errors of "slackware"

Bug search that causes network death, lost slaves under 2.0.X only??
Bug search that causes text-powerflute to kill daemon only under 2.0.X.

----> apcupsd-3.2.0 66096 Dec 18 16:11 apcupsd.c

apcflute.c all code for reconfiguration complete.

----> apcupsd-3.1.0 65925 Dec 8 21:18 apcupsd.c

----> apcupsd-3.0.0

Now binaries only.....

----> apcupsd-2.9.9 63431 Dec 3 13:05 apcupsd.c

Fixed "for (killcount=0;killcount>3;killcount++)" bug
^ killcount.

----> apcupsd-2.9.8 62158 Nov 25 15:13 apcupsd.c

Added user defined magic and security timeouts.
Started powerflute update running master or standalone daemon.
Added another field for SmartUPS procfs file.

----> apcupsd-2.9.7 60945 Nov 18 19:19 apcupsd.c

Added second alternate method for killing power for non SU SmartUPS.
I have one of these older models, and it needs a bigger kick.

APC UPS management under Linux

Disable lockfiles if slave is a netslave with (ups->cable == APC_NET), or ethernet.

----> apcupsd-2.9.6 60156 Nov 10 22:34 apcupsd.c

split source files.....

10885 Nov 10 20:40 apcnet.c :: NetUPS

18805 Nov 10 22:37 config.c :: configuration

5924 Nov 10 19:54 apcflute.c.src :: powerflute interface to apcupsd

5264 Nov 10 12:43 powerflute.c.src :: powerflute tcp management of apcupsd

Brian Schau added lockfiles for serial ports.

NetUPS or EtherUPS now works.

Needs more security likely, you can set your own TCP port number.

Will add user MAGIC at compile in the future.

----> apcupsd-2.9.5 72431 Oct 28 00:27 apcupsd.c

Listing my hack of "powerflute". To enable it edit your /etc/apcupsd.conf and set NETTIME to any value seconds is something to play around.

Reformat subroutines to get apcupsd to handle networks.

----> apcupsd-2.9.4 70707 Oct 25 13:51 apcupsd.c

Smart V/S is in the same class as a BackUPS Pro.

Riccardo Facchetti has been volunteered to solve the network part for the project. YEAH!!!!!!!!!!!!!!

Hello Andre,

I own a SmartUPS v/s and I have played a bit with your apcupsd. I have found that the SmartUPS v/s command set is similar (if not equal) to the one of BackUPS PRO, so I have changed the apcupsd.c code to behave this way and seems to work well.

Another thing: I would like to have an application like powerchute. Of course I am willing to write it, so I will call it powerflute.

My idea is:

apcupsd -> daemon

- controls power status
- listen on a TCP socket for incoming connections
 - get a command from TCP line
 - send the response to the command

powerflute -> application

- connect to apcupsd
 - sends a command to apcupsd
 - get the response from apcupsd

apcupsd is listening to the socket with a select so it don't block. After one or more connections are established, it read non-blocking from the TCP file descriptor(s) and when a char is available, send it to UPS and readline(), then send the buffer (terminated with '\n' to the TCP client over the connection file descriptor.

APC UPS management under Linux

You can note that I have protected apcupsd UPS monitoring from TCP servicing by doing only non-blocking calls to the TCP layer, so there should be no problem for it in detecting power problems without delay caused by TCP connections.

A patch is enclosed for v/s == BKpro and for TCP servicing.
Of course this patch is still very preliminar, but I think it is a good idea.

You can try the idea running the apcupsd with this patch applied, telnetting to the port 6669 from localhost (no external connections allowed now for security) and sending to the daemon some commands for the UPS. The daemon will redirect them to the UPS and send the answer to you. You can connect more than one client to the daemon (max 64 connections).

Please send me comments about it.

Ciao,

Riccardo.

--

Riccardo Facchetti | e-mail: riccardo@master.oasi.gpa.it

The TCP patch is not there but my hack job is present.
The package will have an addition called "powerflute".

----> apcupsd-2.9.3 67655 Oct 23 14:57 apcupsd.c

Fixed spelling of "dissable" to "disable"
Forgot to add this function for ShareUPS.

```
int fillShareUPS(int sharefd, UPSINFO *ups);
```

Brian Schau with needed install changes in Makefile
for UNIFIX.

----> apcupsd-2.9.2 67410 Oct 22 19:34 apcupsd.c

Changed "case" names in /etc/apcupsd.conf, again.....
This should be the last time.....

ShareUPS project maybe nearing an end.....

make install is almost done.

----> apcupsd-2.9.1 65886 Oct 19 00:30 apcupsd.c

Changed "case" names in /etc/apcupsd.conf to lower case.
UPSTYPE SMARTUPS is now UPSTYPE smartups
Changed and add two new UPSMODE "cases" for ShareUPSes
Added Two new external call command for TIMEOUT and LOADLIMIT.
Shutdown types are now called by event not by a general shutdown.
Added CHANGE ME to /sbin/powersc for SmartUPSes to announce that
its needs a batteries changed.

Tests are now run on (2) BackUPS 600's, SmartUPS Net 1400RM 700RM,
and SmartUPS 1250.

SmartUPSes now can call for shutdown based on percent of remaining
battery life or capacity.

APC UPS management under Linux

Cleaned up manpage.....

----> apcupsd-2.9 58603 Oct 16 22:17 apcupsd.c

Changed to POSIX Style of control.

Now uses an external CONFIG file -> /etc/apcupsd.conf

All events are handled in the /sbin/powersc file, no more
sysvinit flavor hassels.....

Added ManPage, well an attempt.

Other things begun.....

----> apcupsd-2.8 42800 Oct 6 14:36 apcupsd.c

Finally solved the PLUG-N-PLAY Cable.....

Tested on a BackUPS 600 and SmartUPS 700RM

Someone needs to test a BackUPS Pro.....

If main power returns during a shutdown series, "apcupsd" will now reboot.

---->

Better fix and support for #940-0023A cable, but still broke, drat.

---->

Adds new code for UNIFIX Linux

----> apcupsd-2.7.1 35902 Sep 3 21:38 apcupsd.c

By Chris:

More changes and fixes with Pro Series tested.

Better method of "walling" the console.

Minor debugging added.

ShareUPS Project Started, BETA.

----> apcupsd-2.7 35413 Aug 14 15:50 apcupsd.c

TESTED on the following setups:

SLACKWARE SMP/NON-SMP (2.1.49) SmartUPS 700 BackUPS 600

REDHAT 4.1 SMP/NON-SMP (2.0.30) BackUPS 600

SUSE 5.0 SMP/NON-SMP (2.0.30) BackUPS 600

NOTE BackUPS Pro Series Untested at this time of release.

All UPSes can be time limited shutdown.

Fixed all possible killpower problems.....

Changed rc.power to include a NOW command.

Changed inittab file and added

What to do when battery power fails.

pn::powerfailnow:/etc/rc.d/rc.power now

Made changes for SuSE distribution.

Change /sbin/init.d/powerfail file

Chanages to CONFIG File

APC UPS management under Linux

```
USE_SLACKWARE=yes
USE_REDHAT=no
USE_SUSE=no
```

Auto Updater

```
USE_MAKE_UPDATES=yes
USE_MAKE_INITTAB=yes
```

```
RedHat version for "powerstatus" file
REDHAT_VERSION=42
```

```
----> apcupsd-2.6 35718 Jul 10 08:10 apcupsd.c
```

Added more info to apcupsd.maunal about special external controls.
SEE "6) OTHER" in apcupsd.maunal

```
----> apcupsd-2.6.pre6 35718 Jul 10 08:10 apcupsd.c
```

Made changes in rc.power to become more universal.
Made N-th attempt to make use of APC cable #940-0095A "DRAT"
in "dumb mode". Will now try "smart mode".
"DOUBLE DRAT" APC has a new cable #940-0095C.
They just want to make things hard for us.....
More Pro Fixes

S.u.S.e work around. This is a major DANGER!!!!
If you call /usr/sbin/apcupsd /dev/apcups killpower,
in a non-power problem situation.
"YOU WILL SCREW YOURSELF" this safe guard is a direct override
of the daemon.

```
----> apcupsd-2.6.pre5a 36279 Jun 23 16:39 apcupsd.c
```

OOPS.....forgot an "endif" in the distribution Makefile

```
----> apcupsd-2.6.pre5 36279 Jun 23 16:39 apcupsd.c
```

Added Linux Flavor Dependencies in CONFIG file.
Fixed "nologin", so only root can login during a powerfailure.
All other attempts are defeated at the login prompt.

```
----> apcupsd-2.6.pre4 35616 Jun 19 15:40 apcupsd.c
```

Possible RedHat 4.2 fix.

```
----> apcupsd-2.6.pre3 35282 Jun 17 01:41 apcupsd.c
```

Fixes RedHat install and Makefile to make backup of files
to be changed during installation. Sorry about the mistake
RedHat users.

```
----> apcupsd-2.6.pre2 35282 Jun 17 01:41 apcupsd.c
```

More Pro Fixes and stuff for RedHat install and Makefile

```
----> apcupsd-2.6.pre1 35279 Jun 16 23:04 apcupsd.c
```

More Pro Fixes and stuff for RedHat install

```
----> apcupsd-2.5 36558 Jun 1 19:41 apcupsd.c
```


APC UPS management under Linux

More Pro Fixes and stuff for RedHat install

----> apcupsd-2.5.pre4 35240 May 23 11:10 apcupsd.c

Can now re-init logfile at boot time with:

{path}/apcupsd /dev/apcups newlog

This will help keep the log file from eating your root partition

940-0020B retested, fine

940-0095A untested -----!!BETA CODE!!-----

----> apcupsd-2.5.pre3 34829 May 22 17:56 apcupsd.c

major bug check....sorry

tested on smart and back ups, not pro yet

940-0020B not retested yet

----> apcupsd-2.5.pre2 28712 May 21 20:39 apcupsd.c

New logging features.

----> apcupsd-2.5.pre

RedHat Support and Back UPS Pro Support via third party.

Christopher J. Reimer

----> apcupsd-2.4 24409 May 19 14:17 apcupsd.c

Back UPS Pro BETA Support

Karsten Wiborg

Christopher J. Reimer

----> apcupsd-2.3 21278 May 13 10:11 apcupsd.c

I made a patch to your apcupsd that allows it to be configured for external commands to run (instead of the built-in wall). This allows me to set it up to send out a call to my pager when the power fails, among other things.

Anyway, here it is.

--

Chris Adams - cadams@ro.com

not public release, yet.

----> apcupsd-2.2 21275 May 8 12:53 apcupsd.c

Fixed psuedo procfs type status file in the /etc directory.

Now works for both SMART and DUMB MODE.

Eric Raymond

Redit of apcupsd.manual.

----> apcupsd-2.1 20926 May 6 23:08 apcupsd.c

Jason Orendorf

"After building and installing the daemon init files I found out why you #define the term _ANNOY_ME...it is rather annoying

APC UPS management under Linux

to get 'wall'ed every 3 seconds - though I do think the feature is important. I made a trivial modification to a few files to make the time delay ('wall' frequency) configurable at build time. If you haven't made this change yourself, feel free to incorporate these changes into your development source if you think it would be useful to others."

Eric Raymond

Edited APC_UPSD.README.1ST to apcupsd.manual.
Many thanks.

Finally added psuedo procfs type status file in the /etc directory.
Broken for DUMB MODE

----> apcupsd-2.0 20119 May 2 17:36 apcupsd.c

Name change to help sunsite.unc.edu "keeper" daemon.

Black Cables #940-0024B and #940-0024C are now supported for APC SmartUPS. They have been tested on an APC SmartUPS SU700/1400RM. I don't know if it will work on APC BackUPS Pro Series.....

Tested SmartUPS SU700RM with SMNP PowerNet(tm) card and SmartMode i/o through DB-9 port with 940-0024B and #940-0024C. This means that you can SMNP manage your SmartUPS with SMNP PowerNet(tm) card. You must have your own SMNP software.....

Added /proc/apcupsd file for pseudo procfs info.

Deleted announce aka -D_ANNOY_ME, for pseudo procfs info file.

Borrowed a bunch of code for Smart-UPS from Pavel Korensky's apcd.c apcd.h.

----> Enhanced-APC-UPS v1.9 April 17, 1997

Name change to help sunsite.unc.edu "keeper" daemon.

----> Enhanced-APC-UPS v1.9 Apr 14 15:02 apc_upsd.c

bug found by Tom Kunicki

"I still can cause the UPS to shutoff with a power disturbance by typing 'apc_upsd /dev/apc_ups killpower' before the disturbance." ... "Maybe you could disable the 'killpower' switch except for when the UPS is in the condition where it is supposed to be used..."

I have tested my BackUPS cable on the following and it works as designed:

APC BackUPS 400, APC BackUPS 600, and APC SmartUPS SU1400RM.

I have tested APC's cable #940-0020B on the following and it works as designed:

APC BackUPS 400, APC BackUPS 600, and APC SmartUPS SU1400RM.

I have NOT tested a BackUPS Pro, since I don't have one yet.

NOTE: My SmartUPS CABLE has broken code and more likely a broken cable design.

Black Cables #940-0024B and #940-0024C are still in the works but not high priority, yet.

----> Enhanced-APC-UPS v1.8 Apr 8 00:06 apc_upsd.c

Corrected defines in code for 940-0020B, 940-0023A, and 940-0095A.

-D_SMARTUPS can no longer be define if you select USE_APC=yes

There is no SmartUPS "Smart Mode" support, it has never been available.

APC UPS management under Linux

I am waiting to hear from APC about pinouts for the following cables
#940-0023A their Unix OS cable and
#940-0095A their Win95OS cable.
If you want to use and APC cable, you must have cable #940-0020B ONLY....

Deleted Enhanced_APC_BackUPS.tar.gz from distribution.
Deleted rc.apc_power, all cables use rc.power.

Not publically released at sunsite.unc.edu

----> Enhanced_APC_UPS v1.7g-fix Apr 3 17:42 Makefile

Error in Makefile, rc.apc_power is for testing APC SmartUPS "black" cable.
All installs should use rc.power only, I forgot to make this change in the
distribution Makefile from the Master Makefile.

----> Enhanced_APC_UPS v1.7g Apr 1 15:31 apc_upsd.c

Possible FULL support for Grey cable APC# 940-0020B for BackUPS only
There still may be a bug like the original Enhanced_APC_BackUPS v1.0,
but has not shown it face yet after FIVE (5) forced power outages.

NO SUPPORT YET: Black cable APC# 940-0024B for SmartUPS BackUPS PRO only
This is take MUCH longer.....I don't have either SmartUPS BackUPS PRO to
test with, but can simulate the effect of PIN#8 on a BackUPS 400 and 600.

----> Enhanced_APC_UPS v1.6b Mar 31 22:03 apc_upsd.c

Added USE_APC in CONFIG for using APC cables
Added for using APC cables rc.apc_power
Basic support like original three-wire daemon.

Black cable APC# 940-0024B for SmartUPS and BackUPS PRO only
Grey cable APC# 940-0020B for BackUPS only

VERY BETA.....REPEAT, VERY BETA!!!!, ALMOST ALPHA.....

Not publically released at sunsite.unc.edu

----> Enhanced_APC_UPS v1.5 Mar 31 12:07 apc_upsd.c

Fixed a possible bug on the SmartUPS side of the daemon.
Retested cable design for "Andreas vasquez Mack"
Soon to test "black APC cable" for "Kofi N. Agyemang" and
"Pete Rylko"
APC Black Cable Support soon v1.6?

----> Enhanced_APC_UPS v1.4 Mar 1 17:18 apc_upsd.c

Added ANNOUNCE defeat in CONFIG

----> Enhanced_APC_UPS v1.3 Mar 1 16:41 apc_upsd.c

Fixed bug with momentary power outage less than 10 seconds with announce.
That darn announce.....
Added original Enhanced_APC_BackUPS package to distribution.
Never released.....

----> Enhanced_APC_UPS v1.2 Mar 1 15:46 apc_upsd.c

Fixed a loop order bug.

APC UPS management under Linux

It now release the power problems message after power returns.

----> Enhanced_APC_UPS v1.1 Feb 28 16:25 apc_upsd.c

Copied new dowall.c from Sysvinit-2.69.

Added support for SmartUPS (non-Smart Accessories Port).

Fixed the above problem of repeated power failures.

Added cable design for SmartUPS (non-Smart Accessories Port).

----> Enhanced_APC_BackUPS v1.0 Sep 20 01:52 backupsd.c

Known Bug in special power situations. IFF (if and only if) the following happens will this daemon fail to do its job.

If your power fails long enough to cause "backupsd" to activate and cleanly shutdown your Linux-Box, and returns later only briefly.

(That is long enough for the UPS to clear the KILL_POWER_BIT and causes)
(your Linux-Box to reboot. But not long enough to recharge you UPS's)
(battery above the low-battery latch state, and power fails again. The)
(daemon cannot currently catch the status change.)

I have never had this happen to me ever, but I can force this to happen under laboratory test conditions. You can too, but it is not advised. Your filesystem will say " !@#%#\$*and much more to you.

Thanks for the support and use of this daemon.

hedrick@astro.dyer.vanderbilt.edu

Andre Hedrick

<http://www.brisse.dk/site/apcupsd/>

