



Ceci est un extrait électronique d'une publication de
Diamond Editions :

<http://www.ed-diamond.com>

Ce fichier ne peut être distribué que sur le CDROM offert
accompagnant le numéro 100 de **GNU/Linux Magazine France**.

La reproduction totale ou partielle des articles publiés dans Linux
Magazine France et présents sur ce CDROM est interdite sans accord
écrit de la société Diamond Editions.

Retrouvez sur le site tous les anciens numéros en vente par
correspondance ainsi que les tarifs d'abonnement.

Pour vous tenir au courant de l'actualité du magazine, visitez :

<http://www.gnulinuxmag.com>

Ainsi que :

<http://www.linux-pratique.com>

et

<http://www.miscmag.com>



NUT : gérez efficacement vos onduleurs

Pouvoir à une coupure d'électricité c'est bien, arrêter correctement ses ordinateurs lorsque les batteries des onduleurs sont vides, c'est mieux.

Malgré sa grande fiabilité logicielle et la mise en oeuvre de redondance matérielle (RAID, multi NIC, multiprocesseur, etc.), un système GNU/Linux reste toujours dépendant de la fourniture d'électricité et des problèmes qui en découlent. Le couple NUT/onduleur est là pour vous.



NUT [NUT] pour Network UPS Tools est une collection de programmes distribués sur un réseau TCP/IP permettant de surveiller et d'administrer de multiples onduleurs. Sa conception en couches inter-connectées lui permet de fournir une vue unifiée et homogène d'un ensemble d'onduleurs souvent hétérogènes.

Le rôle principal de NUT est de permettre aux ordinateurs, protégés par des onduleurs, de s'arrêter proprement lors d'une coupure prolongée d'électricité.

Il est publié comme Logiciel libre sous la GPL. Il fonctionne avec GNU/Linux, les BSDs, MAC OS X, Solaris, IRIX, HP/UX, Tru64 et AIX ; un client existe pour Windows [WinNUT].

Le 23 mars 2004 est sortie la version 2.0.0 de NUT succédant à la série 1.4.x. Cette nouvelle version comme son numéro l'indique est une évolution majeure dans le développement et l'utilisation de NUT, bien que la majeure partie de ces évolutions soit interne, donc non encore visible pour l'utilisateur. Cet article présentera les deux versions de NUT, 1.4.3 et 2.0.0. L'article est basé sur une distribution Debian GNU/Linux mixte Woody/Sarge. Les chemins des fichiers s'y rapportent.

Nous découvrirons dans le présent article une description des onduleurs et de leurs technologies, puis nous nous intéresserons au pourquoi des onduleurs et au besoin de les administrer.

Nous poursuivrons par le fonctionnement de NUT, sa configuration et son histoire. Nous finirons par l'étude d'un cas réel.



Un onduleur kesaco ?

Un onduleur (en anglais *Uninterruptible Power Supply*, aussi noté UPS) protège un équipement électrique des perturbations du réseau électrique (dit « secteur »).

L'onduleur est connecté d'un côté au réseau électrique et de l'autre à l'alimentation de l'équipement électrique.

Dans le cadre d'une installation informatique, ces équipements correspondent à tout ce qui peut se brancher sur le secteur, postes de travail, serveurs, switch, routeurs, modems, imprimantes...

Les onduleurs sont aussi utilisés dans d'autres secteurs que l'informatique, dès que des équipements électriques doivent fonctionner sans perturbation, comme dans le milieu hospitalier, l'industrie...

Les perturbations du réseau électrique sont de plusieurs ordres :

- Coupures de secteur.
- Variations de tension.
- Surtensions – Baisse de tension.
- Parasites.
- Variations de fréquence.

Dans le cas d'une coupure prolongée du secteur, un groupe électrogène peut prendre la relève, lorsque les équipements électriques ne doivent pas être interrompus.

L'onduleur donne le temps au groupe électrogène de démarrer et de stabiliser sa tension de sortie, avant que l'alimentation ne bascule sur ce dernier.

Selon la technologie de l'onduleur, toutes ou quelques-unes de ces perturbations sont traitées. Il en existe trois types, avec diverses dénominations :

- PASSIVE STAND-BY, STAND-BY, OFF LINE,
- LINE INTERACTIVE, IN LINE, STAND-BY INTERACTIVE,
- DOUBLE CONVERSION, ON LINE.

Ces diverses dénominations ont été normalisées par la CEI (Commission Electrotechnique Internationale) avec la norme CEI 62040-3 et son pendant Européen par le Cenelec (Comité Européen de Normalisation) avec la norme ENV 50091-3 [Norme].

Les trois dénominations normalisées sont :

- PASSIVE STAND-BY,
- LINE INTERACTIVE,
- DOUBLE CONVERSION.

La technologie PASSIVE STAND-BY

Cette technologie est utilisée par des onduleurs de faibles puissances (de 250 VA à 1400 VA), et pour des équipements électriques non stratégiques. Ceux-ci sont en permanence alimentés par le secteur non régulé.

Lors d'une coupure de secteur, un commutateur rapide bascule sur la batterie. Seule cette perturbation est traitée.

L'alimentation et la sortie sont monophasées, le signal de sortie est non sinusoïdal (appelé « pseudo-sinusoïdal » ou « trapézoïdal »).

La technologie LINE INTERACTIVE

Cette technologie est utilisée par des onduleurs de moyenne puissance (de 420 VA à 5000 VA) et pour les postes de travail, les petits et moyens serveurs et les équipements d'interconnexions. Ceux-ci sont en permanence alimentés par le secteur régulé en tension et filtré.

Les faibles variations de tension sont corrigées par le régulateur sans utiliser la batterie. Les variations plus importantes sont corrigées en utilisant la batterie. Un filtre de sortie élimine les parasites. Lors d'une coupure de secteur, le commutateur bascule sur la batterie. Toutes les perturbations sont traitées, sauf les variations de fréquence.

L'alimentation et la sortie sont monophasées, le signal de sortie est sinusoïdal.

La technologie DOUBLE CONVERSION

Cette technologie est utilisée par des onduleurs de moyenne et forte puissance (de 1000 VA à plus de 1000 kVA) et pour des parcs d'équipement électrique et des applications critiques.

Ces onduleurs sont généralement associés à des groupes électrogènes et alimentent des salles de machine ou des bâtiments.

Cette technologie est considérée comme la plus performante, toutes les perturbations électriques sont traitées. Les équipements électriques sont en permanence alimentés par un courant recréé par l'onduleur.

Le signal de sortie est une sinusoïdale quasi parfaite générée à partir d'un signal continu, lui-même généré à partir du secteur, d'où le nom de « double conversion ». La régulation permanente de la tension et de la fréquence du signal de sortie est assurée sans utiliser la batterie. Celle-ci est utilisée seulement en cas de coupure, de variation importante de la tension ou de la fréquence du secteur.

L'alimentation et la sortie peuvent être respectivement : monophasée/monophasée, triphasée/monophasée, triphasée/triphasée.

Le couple onduleur ordinateur

Plaçons-nous maintenant dans le cas d'une relation entre un onduleur et un ordinateur. Lors d'une coupure du secteur ou d'un auto-test, l'onduleur utilise sa batterie pour continuer à alimenter l'ordinateur.

L'autonomie d'une batterie est dépendante de sa capacité (mesurée en Volt Ampère, noté VA) et de la consommation de l'ordinateur.

Après un temps d'utilisation de la batterie, en limite d'autonomie, l'onduleur informe de son état l'ordinateur qu'il protège. Cette information circule par l'intermédiaire d'une liaison qui peut être soit série, soit USB.

Si l'onduleur et l'ordinateur sont connectés au même réseau TCP/IP, ils peuvent en outre utiliser le protocole SNMP [SNMP].

Les onduleurs de petites et moyennes capacités disposent souvent des deux types de connecteurs série et USB.

Les onduleurs de moyenne capacité ont généralement la possibilité d'adjonction de carte SNMP.

Pour les onduleurs de grande capacité, les trois types de connecteurs sont disponibles.

On a donc deux types de liaison entre l'onduleur et l'ordinateur, une liaison

d'énergie (ou de puissance) et une liaison d'information.

Si la batterie de l'onduleur atteint une valeur minimale avant le rétablissement du secteur, l'onduleur informe l'ordinateur qu'il va s'arrêter. Cette valeur minimale doit offrir à l'ordinateur un délai suffisant pour effectuer les opérations nécessaires à son arrêt.

Si le secteur revient avant la fin de l'autonomie de la batterie, l'onduleur en informe aussi l'ordinateur.

En règle générale, tous les changements d'état de l'onduleur sont notifiés à l'ordinateur.

Ces informations transmises par l'onduleur sont reçues par NUT qui les traite en fonction de son paramétrage.

Ces traitements peuvent être : l'arrêt du système, l'inscription dans les journaux du système, l'envoi de mail, la communication avec les autres parties de NUT distribuées sur le réseau, l'exécution d'un binaire ou d'un *script shell*.

NUT, le principe

Le principe de NUT est le suivant : un ou plusieurs ordinateurs ont leur alimentation électrique protégée par un ou plusieurs onduleurs – les liaisons de puissance.

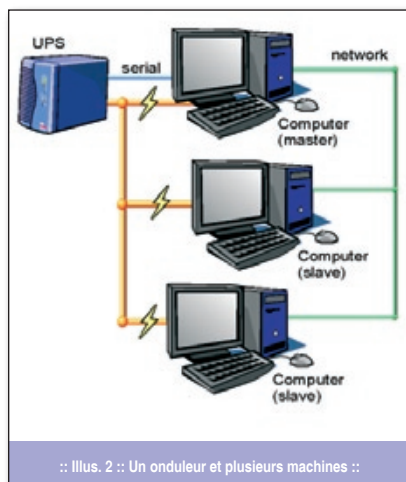
L'un des ordinateurs dit « maître » communique avec un ou plusieurs onduleurs, cette communication peut être réalisée par des liaisons séries, USB ou SNMP – les liaisons d'information.

Cet ordinateur maître surveille l'état des onduleurs et le communique par le réseau TCP/IP aux autres ordinateurs dits « esclaves ».

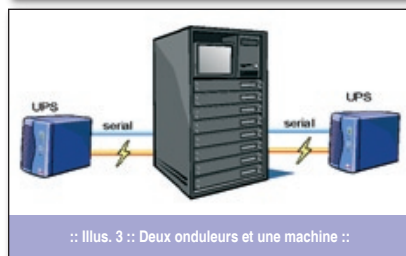
Comme on le constate, NUT peut s'adapter à de multiples configurations, du simple onduleur pour un poste de travail à la salle d'hébergement, en passant par le gros serveur nécessitant plusieurs onduleurs ; le tout en images :



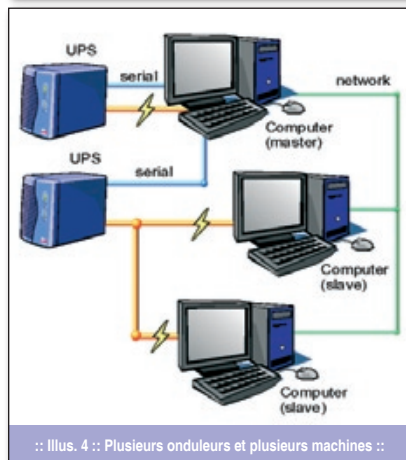
:: Illus. 1 :: Un onduleur et une machine ::



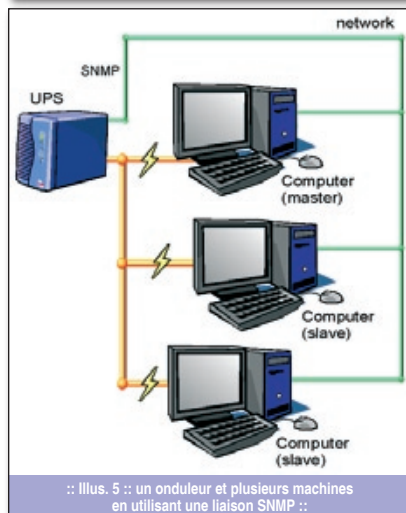
:: Illus. 2 :: Un onduleur et plusieurs machines ::



:: Illus. 3 :: Deux onduleurs et une machine ::



:: Illus. 4 :: Plusieurs onduleurs et plusieurs machines ::



:: Illus. 5 :: un onduleur et plusieurs machines en utilisant une liaison SNMP ::

NUT en détail

Cette flexibilité d'utilisation implique une architecture modulaire. De ce fait, NUT est composé de plusieurs programmes réalisant une action propre et possédant chacun un fichier de configuration dédié. Ces programmes sont classés en quatre catégories : les drivers, le serveur, les clients et les scripts CGI.

La communication entre un onduleur et un ordinateur repose sur un protocole propriétaire propre à une gamme ou à un fabricant. NUT dispose pour répondre à cette diversité de nombreux drivers.

Ceux-ci offrent au serveur UPSD une interface unifiée et homogène d'accès aux onduleurs. Il est la seule voie d'accès aux onduleurs. Il garde en cache les informations fournies par les drivers et répond aux clients autorisés en leur transmettant celles-ci.

UPSD dispose de nombreuses fonctions de contrôle d'accès pour limiter la surveillance et l'administration des onduleurs aux clients habilités.

La communication entre UPSD et les clients repose sur TCP/IP, ce qui permet de les déployer sur un ou plusieurs ordinateurs.

Les scripts CGI permettent à l'aide d'un simple navigateur de surveiller et d'administrer plusieurs onduleurs.

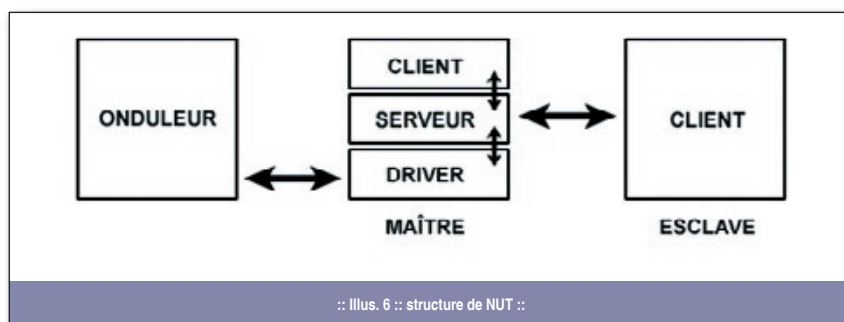
Tous les programmes de NUT sont documentés par une page de manuel. Les fichiers de configuration ont aussi chacun une page de manuel et le répertoire `/usr/share/doc/nut/examples` contient un exemple de ces fichiers. Les exemples des fichiers de configuration des scripts CGI sont disponibles dans le répertoire `/usr/share/doc/nut-cgi/examples`.

N'hésitez pas à consulter la documentation contenue dans les répertoires `/usr/share/doc/nut*` et sur le site du projet NUT [Doc].

Tous les fichiers de configuration sont placés dans le répertoire `/etc/nut/`. Dans le cas particulier où l'un d'eux serait dans un autre répertoire le chemin complet sera indiqué.

Les drivers

Les drivers sont des programmes qui communiquent directement avec l'onduleur. Ils doivent être installés seulement sur l'ordinateur qui supporte la liaison d'information avec l'onduleur, c'est-à-dire sur le maître. Ils sont considérés



:: Illus. 6 :: structure de NUT ::

NUT permet à de nombreux ordinateurs d'être protégés par un unique onduleur de grande capacité et à un unique ordinateur de surveiller de multiples onduleurs.

Il existe divers clients répondant à des besoins particuliers, comme UPSmon qui a la lourde tâche de surveiller l'état de l'onduleur et, le cas échéant, d'arrêter proprement le système protégé par cet onduleur ou UPSlog qui enregistre régulièrement le statut des onduleurs dans les journaux du système, etc.

comme des programmes car ils peuvent être utilisés en ligne de commande. Ce type d'utilisation permet de tester la connexion et la communication entre le driver et l'onduleur.

La syntaxe est :

```
/chemin/<driver> /dev/port
```

Sur une distribution Debian, les drivers sont placés dans le répertoire `/lib/nut/`. Listez ce répertoire pour vérifier la disponibilité d'un driver pour votre onduleur.

Pour un onduleur MGE connecté au deuxième port série :

```
# /lib/nut/mge-shut /dev/ttyS1
Network UPS Tools - MGE UPS SYSTEMS/SHUT driver 0.59 (1.4.3)
# ps aux | grep mge
nut 9385 0.0 0.1 1728 688 ? Ss 11:59 0:00 /lib/nut/mge-shut /dev/ttyS1
# kill -15 9385
```

Il existe plusieurs drivers selon le support et le protocole de communication avec l'onduleur. Le support de la communication peut être de trois types, liaison par câble série, liaison par câble USB ou liaison réseau avec le protocole SNMP.

Dans le cas d'une liaison par câble série le protocole de communication est dépendant du constructeur de l'onduleur. Il existe de ce fait plusieurs drivers pour les liaisons par câble série. Ils permettent à chacun de communiquer avec plusieurs modèles d'onduleur d'une même marque.

Un cas particulier pour le driver GenericUPS qui permet de communiquer avec de nombreux onduleurs qui utilisent un protocole basique appelé « *contact closure* ».

Celui-ci transmet peu d'information, l'état de la batterie et le niveau de puissance. Il est utilisé par des onduleurs bas de gamme.

Les liaisons USB et SNMP sont gérées par des drivers dédiés, respectivement appelés NewhiUPS (basé sur LibUSB ou son prédécesseur spécifique à GNU/Linux, HidUPS) et SNMP-UPS. Ils sont inclus dans les paquets NUT-USB et NUT-SNMP. Ces drivers reposent sur les standards UPS, respectivement Power Device Class USB [USB] et IETF UPS Mib pour SNMP.

Il existe des drivers pour les marques suivantes : APC, Belkin, Best Power, Clary, Cyber Power Systems, Effekta, Energy Sistem, ETA, Ever UPS, Fairstone, Fenton Technologies, Fideltronik, Gemini, HP, Liebert, MGE UPS SYSTEMS, Microdowell, Nitram, Oneac, Online, Orvaldi, Powercom, PowerGuard, PowerKinetics, PowerTech, Powerware, Powerwell, Repotec, SMS, Socomec Sicon, SOLA, SOLA/BASIC, Soltec, Sweex, Tripp-Lite, UPSonic et enfin Victron/IMV.

La liste complète des drivers et des onduleurs supportés peut être trouvée sur le site web du projet NUT [Liste] ou dans le fichier README [Readme].

Il est à remarquer que la société française MGE UPS SYSTEMS [MUS] collabore

activement au projet NUT en fournissant les spécifications de leurs onduleurs et héberge le miroir européen du site web [NUT] ainsi que diverses ressources liées à NUT [NUT].

Cette société emploie aussi le second développeur du projet NUT, Arnaud Quette [AQ]. Ce dernier est responsable du développement Linux/UNIX pour celle-ci, auteur des drivers MGE-Utalk/Shut, SNMP-UPS, [New]hidUPS,... Il est aussi mainteneur des paquets NUT pour la distribution Debian, auteur de divers projets associés (WMNut, Walnut, Agent SNMP, etc.) et responsable de la coordination externe (*packaging*, USB, SNMP, etc.).

Les drivers sont utilisés pour contrôler les onduleurs connectés à l'ordinateur. NUT utilise le fichier *ups.conf* pour décrire chaque onduleur. Ce fichier est constitué d'une suite de sections décrivant chacun d'eux.

Dans le cas d'un onduleur APC appelé UPS1, utilisant le driver NewAPC et connecté sur le deuxième port série, nous avons dans le fichier *ups.conf* une section comme suit :

```
[UPS1]
driver = newapc
port = /dev/ttyS1
cable = 940-00958
```

Les directives *driver* et *port* sont les deux seules obligatoires. Certains onduleurs nécessitent des paramètres supplémentaires, comme la description du câble de liaison pour les onduleurs utilisant le driver NewAPC. Pour cela, on ajoute la directive *cable* dans la section de description de l'onduleur. La version 2.0 de NUT ajoute de nouvelles directives à ces trois dernières, optionnelles, elles répondent à des besoins particuliers. Des informations détaillées sont fournies dans le fichier *ups.conf*.

Pour démarrer ou arrêter un driver on utilise la commande *upsdrvctl*, dont la syntaxe est :

```
upsdrvctl [OPTIONS] (start | stop | shutdown) [ups]
```

Pour tester la configuration de l'onduleur UPS1, on utilise la commande :

```
# upsdrvctl -t start UPS1
Network UPS Tools - UPS driver controller 1.4.3
*** Testing mode: not calling exec/kill
exec: /lib/nut/newapc -a UPS1
```

Pour démarrer tous les onduleurs décrits dans le fichier *ups.conf*, on utilise la commande :

```
# upsdrvctl start
Network UPS Tools - UPS driver controller 1.4.3
Network UPS Tools (version 1.4.3) - APC Smart protocol driver
Driver version 1.99.1a, command table version 2.0
Detected Back-UPS Pro 650 [NB9929351695] on /dev/ttyS1
```

Pour les arrêter :

```
# upsdrvctl stop
Network UPS Tools - UPS driver controller 1.4.3
Stopping UPS: UPS1
```

Le serveur

Le serveur UPSD communique aux clients et scripts CGI les données fournies par les drivers. Il doit donc être installé sur l'ordinateur supportant les drivers, c'est-à-dire sur le maître. Cette communication avec les clients et scripts CGI utilise par défaut les ports TCP et UDP 3493 pour les versions 1.4 et antérieures de NUT. La version 2.0 n'utilise plus que le port TCP.

Le daemon UPSD est lancé par le script */etc/init.d/nut*. Il convient au préalable de modifier le fichier */etc/default/nut* et de changer le paramètre *START_UPSD* de la valeur *no* à *yes*. Les fichiers de configuration utilisés par le daemon UPSD sont *ups.conf*, *upsd.conf* et *upsd.users*.

Le fichier *ups.conf* permet au daemon UPSD de connaître les onduleurs qu'il doit surveiller.

Le fichier *upsd.conf* est utilisé pour contrôler l'accès au daemon UPSD. Il contient la liste des hôtes autorisés à se connecter. Cette description débute par l'association des noms d'hôte aux adresses IP :

```
ACL all 0.0.0.0/0
ACL localhost 127.0.0.1/32
ACL HOST1 192.168.0.1/32
```

Puis se poursuit par la définition des droits. La syntaxe diffère selon la version de NUT.

Pour la version 1.4 de NUT la syntaxe est :

```
ACCESS action level aclname
```

L'action peut être de deux types *grant* ou *deny*.

SYSTEME

Il y a trois niveaux d'actions : `base`, `monitor` et `all`. Le niveau `base` limite les connexions au protocole TCP et aux requêtes simples. Le niveau `monitor` permet de totalement gérer l'onduleur. Le niveau `all` remplace tous les niveaux et est surtout utilisé pour interdire les accès.

`AcIname` est le nom de l'hôte défini plus haut.

Voici la suite du fichier `upsd.conf` :

```
ACCESS grant monitor HOST1
ACCESS grant monitor localhost
ACCESS deny all all
```

La version 2.0 de NUT clarifie l'utilisation de la directive `ACCESS`. Elle est tout simplement remplacée par les deux directives `ACCEPT` et `REJECT` selon la syntaxe :

```
ACCEPT list_acIname
REJECT list_acIname
```

`list_acIname` est une liste de noms d'hôte définis par la section `ACL` et séparés par une espace.

Voici la suite du fichier `upsd.conf` pour NUT 2.0 :

```
ACCEPT HOST1 localhost
REJECT all
```

Le fichier `upsd.users` décrit les propriétés des utilisateurs autorisés à se connecter au daemon UPSD. Ces utilisateurs possèdent comme propriétés un nom, un mot de passe, un nom d'hôte et une liste d'actions et de paramétrages autorisés.

Quelques considérations de sécurité. Le fichier `upsd.users` contenant des mots de passe doit appartenir à l'utilisateur `root` et avoir des droits positionnés à `0600`. D'autre part, la communication entre le daemon UPSD et les clients n'est pas cryptée. Les mots de passe circulent donc en clair sur le réseau et peuvent être sniffés. Une version de NUT supportant le protocole SSL est à l'étude.

Chaque utilisateur défini dans le fichier `upsd.users` est décrit par une section commençant par son nom entre crochet. Tous les paramètres d'un utilisateur doivent être décrits dans cette section. Les paramètres possibles sont :

`password` : le mot de passe de l'utilisateur.

`allowfrom` : liste des noms d'hôte à partir desquels des connexions sont autorisées.

`actions` : autorise l'utilisateur à changer des variables, des drapeaux sur l'onduleur. Il existe deux valeurs pour ce paramètre.

■ `SET` autorise l'utilisateur à changer les valeurs de certaines variables de l'onduleur.

■ `FSD` autorise l'utilisateur à changer le drapeau d'arrêt forcé de l'onduleur.

`instcmds` : liste des commandes que l'utilisateur est autorisé à envoyer à l'onduleur. Il existe plusieurs valeurs pour ce paramètre dont `ALL` qui autorise toutes les commandes. La liste complète des commandes supportées par l'onduleur est fournie par la commande `upscmd -l ups`. Le fichier `/usr/share/nut/cmdvartab` contient une description des commandes possibles.

Le dernier paramètre possible pour le fichier `upsd.users` est `upsmon`. Celui-ci est lié au client éponyme le plus important de NUT (décrit ci-dessous). Ce paramètre peut prendre deux valeurs `master` ou `slave`. La valeur `master` définit l'ordinateur maître et la valeur `slave` l'ordinateur esclave.

Un utilisateur `user_master` lié au maître `HOST1` sera défini ainsi :

```
[user_master]
password = aZerTyuIoP
allowfrom = HOST1
upsmon master
```

Les clients

Les clients communiquent avec le serveur UPSD via le réseau, ils peuvent être installés selon l'usage sur le maître ou les esclaves. Il existe plusieurs clients, chacun ayant un rôle et un usage particulier. Certains se détachent de par leur rôle (les scripts CGI) ou leur importance (UPSmon). Ces derniers seront étudiés à part. En voici quelques autres :

UPSC

UPSC permet d'interroger un serveur UPSD et d'obtenir les valeurs des variables d'un des onduleurs gérés par ce serveur. Par défaut UPSC liste l'ensemble des valeurs des variables connues. La valeur d'une variable peut être connue en précisant le nom de cette variable sur la ligne de commande. La syntaxe est :

```
upsc ups [variable]
```

Du fait de sa simplicité d'utilisation, UPSC peut être facilement utilisé comme un outil de diagnostic ou dans des *scripts shell*.

L'interrogation de l'onduleur UPS1 relié à l'ordinateur maître `HOST1` donne :

```
$ upsc UPS1@HOST1
battery.alarm.threshold: 0
battery.charge: 100.0
battery.charge.restart: 15
battery.date: 03/01/02
battery.runtime: 1000
battery.runtime.low: 120
battery.voltage: 14.02
battery.voltage.nominal: 012
driver.name: newapc
driver.parameter.cable: 940-0095B
driver.parameter.port: /dev/ttyS1
driver.version: 1.4.3
driver.version.internal: 1.99.1a
input.frequency: 50.00
input.quality: FF
input.sensitivity: H
input.transfer.high: 253
input.transfer.low: 208
input.transfer.reason: S
input.voltage: 234.7
input.voltage.maximum: 234.7
input.voltage.minimum: 234.7
output.voltage: 234.7
output.voltage.target.battery: 230
ups.delay.shutdown: 180
ups.delay.start: 000
ups.firmware: 12.6.1
ups.id: UPS1
ups.load: 053.9
ups.mfr: APC
ups.mfr.date: 07/16/99
ups.model: Back-UPS Pro 650
ups.serial: NB9929351695
ups.status: OL
ups.test.interval: 604800
ups.test.result: NO
```

UPSLOG

Upslog est un daemon de journalisation. Il interroge à intervalle régulier le serveur UPSD et enregistre les valeurs des variables de l'onduleur dans un journal d'évènement.

La syntaxe est :

```
upslog [options]
```

La liste des variables à enregistrer doit être fournie par une chaîne de format passée avec l'option `-f`. Par défaut, UPSlog journalise par ligne d'enregistrement les valeurs des variables suivantes :

- La date selon le format AAAAMMJJ hhmmss.
- La charge de la batterie,
- La tension du secteur,
- La charge de l'onduleur,
- Le statut de l'onduleur,
- La température de l'onduleur,
- La fréquence du secteur.

L'ensemble des options d'UPSlog peut être consulté dans la page du manuel.

Les informations enregistrées dans le journal d'évènement peuvent être utilisées pour tracer des graphes avec GNUplot.

Pour lancer le daemon UPSlog on peut utiliser la commande :

```
$ upslog -l /var/log/ups.log -s UPS1@HOST1
```

UPSRW

UPSRW permet d'afficher et de changer les valeurs des variables accessibles en lecture/écriture d'un onduleur. Tous les onduleurs et les drivers ne permettent pas le changement de leurs variables. Dans le cas où les changements sont possibles, l'utilisateur doit posséder les droits nécessaires, ceux-ci sont définis dans le fichier `upsd.users`.

L'interrogation de l'onduleur UPS1 relié à l'ordinateur maître HOST1 donne :

```
$ upsrw UPS1@HOST1
[battery.alarm.threshold]
Battery alarm threshold
Type: ENUM
Option: "0" SELECTED
Option: "T"
Option: "L"
Option: "N"

[battery.charge.restart]
Minimum battery level for restart after power off
Type: ENUM
Option: "00"
Option: "15" SELECTED
Option: "50"
Option: "90"

[...]

[ups.delay.shutdown]
Interval to wait after shutdown with delay command
Type: ENUM
Option: "020"
Option: "180" SELECTED
Option: "300"
Option: "600"

[ups.test.interval]
Interval between self tests
Type: ENUM
Option: "1209600"
Option: "604800" SELECTED
Option: "0"
```

UPSCMD

UPSCMD permet d'envoyer des commandes (en anglais *instant command*) à l'onduleur. Ces commandes, si elles sont supportées et autorisées, déclenchent une action sur l'onduleur. Ces actions peuvent être de démarrer ou d'arrêter le test de la batterie, du panneau de contrôle, de passer l'onduleur sur la batterie, d'arrêter l'onduleur, etc.

La syntaxe est :

```
upscmd [options] ups [commande]
```

La liste des commandes supportées par le driver et l'onduleur peut être obtenue par l'option `-l` de la commande `upscmd`. Leurs descriptions peuvent être consultées à la fin du fichier `/usr/share/nut/cmdvartab`.

Les droits pour pouvoir utiliser UPSCMD et envoyer une commande à l'onduleur sont définis dans le fichier `upsd.users` par le paramètre `instcmds`.

L'interrogation de l'onduleur UPS1 relié à l'ordinateur maître HOST1 donne :

```
$ upscmd -l UPS1@HOST1
Instant commands supported on UPS [UPS1@HOST1]:

load.on - Turn on the load immediately
test.panel.start - Start testing the UPS panel
calibrate.start - Start run time calibration
calibrate.stop - Stop run time calibration
shutdown.stayoff - Turn off the load and remain off
shutdown.return - Turn off the load and return when power
is back
test.failure.start - Start a simulated power failure
test.battery.start - Start a battery test
test.battery.stop - Stop the battery test
load.off - Turn off the load immediately
```

UPSmmon

UPSmmon est le client le plus important dans cette collection de programme appelée NUT. Il a pour rôle principal de surveiller l'état de l'onduleur et d'arrêter proprement l'ordinateur avant la fin de l'autonomie de la batterie de l'onduleur.

Le daemon UPSmmon est lancé par le script `/etc/init.d/nut`. Il convient au préalable de modifier le fichier `/etc/default/nut` et de changer le paramètre `START_UPSMON` de la valeur `no` à `yes`. Le fichier de configuration utilisé par le daemon UPSmmon est `upsmon.conf`. UPSmmon peut être associé au programmeur (en anglais *scheduler*) UPSsched. Celui-ci utilise le fichier de configuration `upssched.conf`.

Le fichier `upsmon.conf` définit toutes les actions réalisées par le daemon UPSmmon. Parmi celle-ci, UPSmmon doit communiquer avec UPSD. On trouvera donc dans le fichier `upsmon.conf` le login et mot de passe de l'utilisateur autorisé à surveiller le maître. Il convient donc d'appliquer à ce fichier les mêmes règles de sécurité que celles appliquées aux fichiers `upsd.conf` et `upsd.users`.

UPSmmon peut être utilisé selon trois types de configuration :

1) Master

UPSmmon est défini comme `master` lorsque le système sur lequel il fonctionne

est alimenté par un onduleur et est directement connecté à lui par la liaison d'information.

Dans cette configuration, UPSmmon arrête le système qu'il gère après l'arrêt de tous les esclaves.

2) Slave

UPSmmon est défini comme `slave` lorsque le système sur lequel il fonctionne est alimenté par un onduleur et n'est pas directement connecté à lui par la liaison d'information. Il connaît l'état de l'onduleur en communiquant avec l'ordinateur défini comme maître.

Dans cette configuration, UPSmmon arrête le système qu'il gère avant l'arrêt du maître.

3) Monitor-only

UPSmmon est défini comme `monitor-only` lorsque le système sur lequel il fonctionne notifie l'état, ou les changements d'état d'un onduleur, sans devoir s'arrêter lorsque celui-ci arrive en fin d'autonomie.

Dans le cas d'une configuration simple avec un ordinateur connecté à un onduleur, l'ensemble des composants de NUT (drivers, UPSD et UPSmmon) sont déclarés comme maître.

Voici les paramètres les plus importants du fichier `upsmon.conf` :

```
MONITOR <system> <powervalue> <username> <password> ("master"|"slave")
```

Cette directive obligatoire déclare le système qui doit être surveillé. La directive `MONITOR` doit être déclarée autant de fois qu'il y a de systèmes à surveiller.

Le paramètre `system` indique l'onduleur à surveiller. Il se rapporte à la déclaration du fichier `ups.conf`.

L'entier `powervalue` indique le nombre d'onduleurs directement reliés à l'ordinateur (gestion de la redondance). Dans le cas d'un ordinateur défini comme `monitor-only`, cette valeur est nulle.

Les paramètres `username` et `password` indiquent le nom d'utilisateur et le mot de passe à utiliser pour se connecter au serveur UPSD. Ils doivent être identiques à ceux définis dans le fichier `upsd.users`.

Le dernier paramètre définit quand UPSmmon doit arrêter le système : la valeur `master` indique que le système

SYSTEME

doit s'arrêter en dernier, après l'arrêt des esclaves. La valeur `slave` indique que le système doit s'arrêter dès la notification de fin d'autonomie de l'onduleur.

Sur le maître HOST1 UPSmon sera défini ainsi :

```
MONITOR UPS1@HOST1 1 user_master aZeRtYuIoP master
MINSUPPLIES <num>
```

Certains systèmes ont plusieurs alimentations reliées chacune à un onduleur. La directive `MINSUPPLIES` indique le nombre minimal d'alimentations effectives avant de déclencher l'arrêt du système.

```
POLLFREQ <n>
```

Ce paramètre indique l'intervalle en seconde entre deux interrogations du serveur UPSD par UPSmon lors d'une activité normale. Par défaut ce paramètre est fixé à cinq secondes.

```
POLLFREQUALERT <n>
```

Ce paramètre indique l'intervalle en seconde entre deux interrogations du serveur UPSD par UPSmon lorsque l'onduleur utilise la batterie. Par défaut ce paramètre est fixé à cinq secondes.

```
DEADTIME <n>
```

UPSmon interroge régulièrement l'onduleur pour connaître son état, par l'intermédiaire du serveur UPSD. Si l'interrogation échoue, l'onduleur est considéré dans l'état `stale`. Si l'onduleur reste dans cet état plus de `DEADTIME` secondes, l'onduleur est considéré dans l'état mort.

Cette valeur doit être un multiple des deux valeurs précédentes, en règle générale trois fois la plus grande de ces valeurs. Par défaut ce paramètre est fixé à quinze secondes.

Un onduleur précédemment connu comme utilisant la batterie et passant dans l'état mort indique que sa batterie est faible et qu'il devra très bientôt s'arrêter. Dans ce cas, UPSmon déclenche la procédure d'arrêt du système.

```
SHUTDOWNCMD "<command>"
```

La directive `SHUTDOWNCMD` définit la commande utilisée pour arrêter le système.

En général, on a :

```
SHUTDOWNCMD "/sbin/shutdown -h +0"
NOTIFYCMD <command>
```

La directive `NOTIFYCMD` définit la commande utilisée par UPSmon pour envoyer des notifications. Pour utiliser un *script shell* local :

```
NOTIFYCMD /usr/local/sbin/nut_notifie
NOTIFYFLAG <notify type> <flag>[+<flag>][+<flag>] ...
```

La directive `NOTIFYFLAG` associe à un événement des méthodes d'envoi de notification.

Le paramètre `notify type` identifie l'évènement affectant l'onduleur. Il peut prendre plusieurs valeurs :

■ `ONLINE` : l'onduleur est de nouveau alimenté par le secteur.

■ `ONBATT` : l'onduleur utilise la batterie.

■ `LOWBATT` : la batterie de l'onduleur est faible, signalant la fin d'autonomie de l'onduleur.

■ `FSD` : le maître demande l'arrêt de l'onduleur (`FSD` = "Forced Shutdown").

■ `COMMOK` : la communication avec l'onduleur est établie.

■ `COMMBAD` : la communication avec l'onduleur est perdue.

■ `SHUTDOWN` : le système va être arrêté.

■ `REPLBATT` : la batterie de l'onduleur est défectueuse ou en fin de vie et doit être remplacée.

■ `NOCOMM` : l'onduleur est inaccessible pour la supervision.

Les drapeaux indiquent la méthode d'envoi de la notification :

■ `SYSLOG` : écrit le message dans les journaux.

■ `WALL` : écrit le message sur la console de tous les utilisateurs connectés.

■ `EXEC` : exécute la commande définie par la directive `NOTIFYCMD`, et lui passe le message en paramètre.

■ `IGNORE` : ne fait rien.



Les scripts CGI

Les scripts CGI permettent à partir d'un navigateur de visualiser l'état d'un ou plusieurs onduleurs, de les administrer

en modifiant la valeur de variable ou d'envoyer des commandes.

L'utilisation de scripts CGI a pour prérequis l'installation et la configuration d'Apache (ou tout autre serveur web) sur le serveur d'installation de ces scripts.

Il y a trois scripts CGI `upsstats.cgi`, `upsimage.cgi` et `upsset.cgi` :

■ `UPSstats` génère une page HTML agrégeant des informations d'état liées à un ou plusieurs onduleurs.

■ `UPSimage` est utilisé par `UPSstats` pour générer des jauges de tension de sortie, état de charge de la batterie et charge d'un onduleur.

■ `UPSset` permet d'administrer un onduleur, il rassemble les fonctionnalités des clients `UPSRW` et `UPSCMD` dans une interface web.

Les scripts CGI utilisent quatre fichiers de configuration. Chacun a un modèle dans le répertoire `/usr/share/doc/nut-cgi/examples`.

Ces fichiers sont à copier dans le répertoire `/etc/nut`. Deux sont à modifier `hosts.conf` et `upsset.conf`.

Les deux autres `upsstats-single.html` et `upsstats.html` sont des gabarits utilisés par les CGI, leurs modifications sont en général inutiles.

Le fichier `hosts.conf` permet aux scripts CGI de connaître la liste des onduleurs avec lesquels ils peuvent interagir. Chaque onduleur est déclaré par une directive `MONITOR` :

```
MONITOR <system> "<host description>"
```

Le paramètre `system` identifie l'onduleur : **UPS1@HOST1.mondomaine.com**

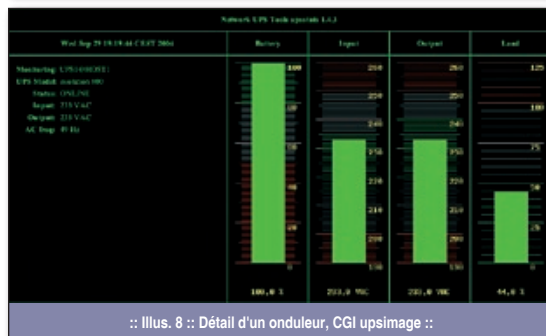
Le paramètre `host description` est utilisé par les scripts CGI pour afficher une description succincte de l'onduleur.

Il est recommandé de configurer le serveur web pour limiter l'accès des scripts CGI aux seuls hôtes autorisés.

L'utilisation d'un fichier `htaccess` dans le répertoire contenant les scripts CGI est fortement conseillé. Une fois la configuration sécurisée, la ligne `I_HAVE_SECURED_MY_CGI_DIRECTORY` du fichier `upsset.conf` peut être dé-commentée.

Network UPS Tools upsstats 1.4.3 Wed Sep 29 19:19:31 CEST 2004						
System	Model	Status	Battery	Input	Output	Load UPS Temp
UPS1	Back-UPS Pro 650	ONLINE	100.0 %	233.2 VAC	233.2 VAC	046.8 %
UPS2	Back-UPS Pro 650	ONLINE	100.0 %	227.5 VAC	227.5 VAC	043.5 %
UPS3	evolution 800	ONLINE	100 %	234 VAC	233 VAC	44 %

:: Illus. 7 :: Page principale, CGI upsstats ::



:: Illus. 8 :: Détail d'un onduleur, CGI upsimage ::

NUT l'histoire

L'histoire de NUT débute en mars 1998 lorsque Russell Kroll publie la première version du logiciel Smartupstools numérotée 0.10, sous la GPL. Cette première version est dédiée aux onduleurs Smart-UPS de la société APC.

Dans cette version on découvre déjà l'architecture client/serveur actuelle de NUT : le daemon serveur de communication avec l'onduleur (UPSD), le daemon client de surveillance (UPSmon), les CGI de génération de la page de statut (`upsstats.cgi` et `upsimage.cgi`).

Le protocole de communication réseau est bâti sur UDP. Il permet déjà aux trois parties de Smartupstools (UPSD, UPSmon, et les CGI) d'être distribuées sur des ordinateurs différents.

Les séries 0.40 (juin 1999) et 0.41 (juillet 1999) voient une partie du code réécrite pour permettre le support de nouveaux onduleurs de marque APC (Smart-UPS, Back-UPS, et Back-UPS pro) ainsi que de la société Trust (KingPro 425 et 625). L'architecture se voit adjoindre de nouveaux éléments : les clients UPSC et UPSlog.

Toutes ces évolutions incitent les auteurs à trouver un nouveau nom pour leur logiciel. La série 0.41 se termine avec la version 0.41.4, puis suit en novembre 1999 la version 0.42 baptisée NUT, nom trouvé par Kern Sibbald.

Après quatre ans de développement la version 1.0.0 de NUT apparaît le 19 août 2002. Elle marque l'inclusion de plus d'une dizaine de nouveaux drivers, la stabilisation du code, de l'architecture et du protocole de communication, ainsi que l'officialisation par l'IANA du port 3493 dédié au projet NUT.

L'évolution de NUT se poursuit. Le 29 juillet 2003 la version 1.4.0 est dupliquée pour devenir la nouvelle branche de développement 1.5. Celle-ci débouche le

PUBLICITÉ

2 SITES INCONTOURNABLES

Toute l'actualité du magazine sur :

www.gnulinuxmag.com



Abonnements et anciens numéros en vente sur :

www.ed-diamond.com

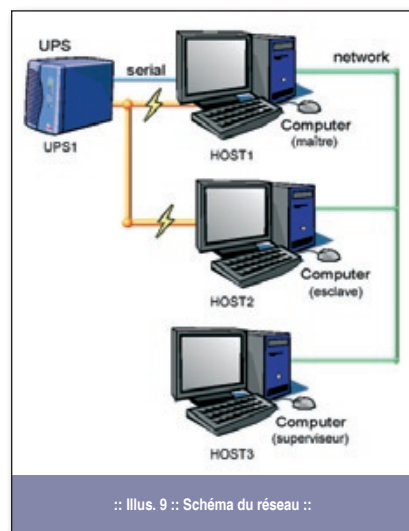
16 mars 2004 sur la version 1.5.15 qui devient le 23 la nouvelle série stable de NUT numérotée 2.0.0. La série 1.4 se clôt avec la version 1.4.3. L'expérience acquise par les versions antérieures a permis l'homogénéisation du code et de l'architecture de la version 2.0.0. L'ensemble des composants de NUT utilise le même schéma de *nommage* pour les commandes et les variables.

L'évolution du protocole de communication entre le serveur et les clients, et du schéma de nommage, rompt partiellement la compatibilité avec les versions antérieures de NUT. L'évolution de l'infrastructure a permis de simplifier et d'alléger le code, de faciliter l'ajout de nouvelles fonctionnalités et l'écriture des composants de NUT. Vous trouverez sur cette page [Contrib] des informations sur les multiples contributeurs (individuels ou sociétés) au projet NUT.

Maintenant en pratique

Après la théorie, la pratique, prenons le cas de deux ordinateurs HOST1 (192.168.0.1) et HOST2 (192.168.0.2), dont les alimentations sont protégées par un onduleur MGE Pulsar Évolution appelé UPS1. Ces ordinateurs font partie d'un réseau TCP/IP.

HOST1 est le maître, il est relié à UPS1 par un câble série connecté sur son deuxième port série. HOST2 est l'esclave, il communique avec le maître via le port TCP 3493. Un troisième ordinateur HOST3 (192.168.0.3) supervise le réseau et les onduleurs.



:: Illus. 9 :: Schéma du réseau ::

La communication entre les différentes parties de NUT, réparties sur les ordinateurs, utilise le réseau TCP/IP. Il va sans dire que les alimentations des équipements d'interconnexion (switch, hub, etc.) doivent être protégées par un ou plusieurs onduleurs, dont l'autonomie doit être supérieure à celle des onduleurs protégeant les ordinateurs.

Cette étude de cas utilise sur la version 1.4.3 de NUT. À la date de rédaction de cet article, il s'agit de la version disponible pour la version Sarge de la distribution Debian GNU/Linux.

HOST1 : Le maître

Seul le paquet `nut` est à installer sur le maître :

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

Cette commande télécharge et installe le paquet `nut`. Le script d'installation ajoute l'utilisateur système nommé `nut` et le groupe du même nom.

HOST1 devant communiquer avec l'onduleur UPS1 via un câble série, le driver de NUT doit pouvoir accéder en lecture/écriture au deuxième port série (`ttyS1`).

```
$ ls -l /dev/ttyS1
crw-rw---- 1 root dialout 4, 65 Sep 29 2003 /dev/ttyS1
```

Comme on peut le voir, ce fichier de périphérique appartient à l'utilisateur `root` et au groupe `dialout`.

La politique du système Debian pour permettre à un utilisateur particulier d'accéder à ce type de fichier est de l'ajouter au groupe `idoine`.

```
# adduser nut dialout
```

Ce qui donne :

```
$ id nut
uid=103(nut) gid=103(nut) groups=103(nut),20(dialout)
```

Plusieurs fichiers de configuration sont à personnaliser, le fichier `ups.conf` pour le driver, les fichiers `upsd.conf` et `upsd.users` pour le serveur, le fichier `upsmon.conf` pour la partie client, le « monitoring » du maître. Les valeurs par défaut conviennent pour beaucoup de directives ; elles ne sont à modifier qu'en cas de difficultés, ou de configuration particulière.

Le fichier `ups.conf` est assez simple à comprendre, on choisit le driver et le port série.

```
[UPS1]
driver = mge-shut
port = /dev/ttyS1
```

Des tests peuvent être nécessaires pour bien choisir son driver. Ils sont réalisés en modifiant le fichier `ups.conf` et en utilisant la commande `upsdrvctl`.

```
# upsdrvctl -v start UPS1
```

Le fichier `upsd.conf` contient la liste et la description des hôtes autorisés à se connecter au serveur. On veillera à interdire l'accès au serveur à tout hôte non défini par la directive « `ACCESS deny all all` ».

```
ACL all 0.0.0.0/0
ACL HOST1 192.168.0.1/32
ACL HOST2 192.168.0.2/32
ACL HOST3 192.168.0.3/32
```

```
ACCESS grant monitor HOST1
ACCESS grant monitor HOST2
ACCESS grant monitor HOST3
ACCESS deny all all
```

Le fichier `upsd.users` contient la définition des utilisateurs et leurs rôles. Quatre utilisateurs seront utilisés :

■ `user_master` pour définir le maître HOST1 ;

■ `user_slave` pour définir les esclaves ; ici il n'y en a qu'un seul HOST2 ;

■ `cgi` pour les hôtes où fonctionnent les scripts CGI : cet utilisateur ne permet que la consultation de l'état de l'onduleur ;

■ `admin` pour permettre à certains hôtes d'administrer l'onduleur.

```
[user_master]
password = aZerYuiOp
allowfrom = HOST1
upsmon master
```

```
[user_slave]
password = p0iUyTrEzA
allowfrom = HOST2
upsmon slave
```

```
[cgi]
password = qSdFgHjKlM
allowfrom = HOST3
```

```
[admin]
password = mKjhGfDsQ
allowfrom = HOST1 HOST3
actions = set
instcmds = all
```

Le fichier `upsmon.conf` définit comment le daemon `upsmon` surveille le maître HOST1 et comment il l'arrête. Beaucoup de directives sont déjà renseignées ou ont une valeur par défaut, d'autres doivent être personnalisées :

■ `MONITOR` indique le système à surveiller ;

NOTIFYCMD indique la commande à utiliser pour notifier l'administrateur, elle se réfère à la valeur **EXEC** de la directive **NOTIFYFLAG** ;

NOTIFYFLAG indique pour un type d'évènement comment la notification doit être émise.

```
MONITOR UPS1@HOST1 1 user_master aZeRtYuIoP master
SHUTDOWNCMD "/sbin/shutdown -h +0"
NOTIFYCMD /usr/local/sbin/nut_notifie
NOTIFYFLAG ONLINE SYSLOG+MAIL+EXEC
NOTIFYFLAG ONBATT SYSLOG+MAIL+EXEC
NOTIFYFLAG LOWBATT SYSLOG+MAIL+EXEC
NOTIFYFLAG FSD SYSLOG+MAIL+EXEC
NOTIFYFLAG COMMOK SYSLOG+MAIL+EXEC
NOTIFYFLAG COMMBAD SYSLOG+MAIL+EXEC
NOTIFYFLAG SHUTDOWN SYSLOG+MAIL+EXEC
NOTIFYFLAG REPLBATT SYSLOG+MAIL+EXEC
NOTIFYFLAG NOCOMM SYSLOG+MAIL+EXEC
```

Le script **nut_notifie** envoie un mail à l'administrateur pour lui notifier le changement d'état de l'onduleur :

```
#!/bin/bash
echo "$*" | sendmail -F "$UPNAME" admin@mondomaine.com
```

Ce script doit être exécutable :

```
$ ls -l /usr/local/sbin/nut_notifie
-rwxr-xr-x 1 root root 78 Sep 29 2003 nut_notifie
```

Dans le fichier **/etc/default/nut** la valeur des directives **START_UPSD** et **START_UPSMON** est à changer de « no » à « yes », pour pouvoir lancer les daemons **upsd** et **upsmmon** avec le script de démarrage :

```
# /etc/init.d/nut start
```

HOST2 : L'esclave

Comme pour le maître, seul le paquet **nut** est à installer sur l'esclave :

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

Le fichier **upsmmon.conf** définit comment le daemon **upsmmon** surveille l'esclave **HOST2** et comment il l'arrête. Les directives à modifier sont identiques à celles du maître, seule la directive **MONITOR** varie. Ici aussi on utilise le script **nut_notifie** pour envoyer des mails de notification à l'administrateur.

```
MONITOR UPS1@HOST1 1 user_slave p0iUyTrEzA slave
SHUTDOWNCMD "/sbin/shutdown -h +0"
NOTIFYCMD /usr/local/sbin/nut_notifie
NOTIFYFLAG ONLINE SYSLOG+MAIL+EXEC
NOTIFYFLAG ONBATT SYSLOG+MAIL+EXEC
NOTIFYFLAG LOWBATT SYSLOG+MAIL+EXEC
NOTIFYFLAG FSD SYSLOG+MAIL+EXEC
NOTIFYFLAG COMMOK SYSLOG+MAIL+EXEC
NOTIFYFLAG COMMBAD SYSLOG+MAIL+EXEC
NOTIFYFLAG SHUTDOWN SYSLOG+MAIL+EXEC
NOTIFYFLAG REPLBATT SYSLOG+MAIL+EXEC
NOTIFYFLAG NOCOMM SYSLOG+MAIL+EXEC
```

Dans le fichier **/etc/default/nut** la valeur de la directive **START_UPSMON** est à changer de « no » à « yes », pour pouvoir lancer

le daemon **upsmmon** avec le script de démarrage :

```
# /etc/init.d/nut start
```

HOST3 : Le superviseur

HOST3 supervise le réseau et l'onduleur **UPS1** en utilisant les scripts CGI de **NUT**. L'utilisation des scripts CGI a pour prérequis l'installation et la configuration d'Apache (ou tout autre serveur Web) sur cette machine. Les CGI de **NUT** sont installés avec le paquet **nut-cgi**.

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

Les quatre fichiers de configuration des scripts CGI ont un modèle dans le répertoire **/usr/share/doc/nut-cgi/examples** ; ces fichiers sont à copier dans le répertoire **/etc/nut**. Deux sont à modifier : **hosts.conf** et **upsset.conf** ; les deux autres, **upsstats-single.html** et **upsstats.html** sont des gabarits utilisés par les CGI, leurs modifications est en général inutile. Le fichier **hosts.conf** contient la liste des onduleurs à surveiller, un par directive **MONITOR** :

```
MONITOR UPS1@HOST1.mondomaine.com "HOST1 MGE UPS"
```

Pour la configuration d'Apache, il faut ajouter la directive **ScriptAlias** dans la section **VirtualHost** *ad hoc* du fichier **/etc/apache/httpd.conf** ; pour une consultation uniquement locale, il suffit d'utiliser celle définissant l'hôte **localhost** :

```
<VirtualHost localhost>
DocumentRoot /var/www
ServerName localhost

ScriptAlias /cgi-bin/nut /usr/lib/cgi-bin/nut

ErrorLog /var/log/apache/localhost-error.log
TransferLog /var/log/apache/localhost-access.log
</VirtualHost>
```

Les scripts CGI étant facilement utilisables, vous devez impérativement ajouter un fichier **.htaccess** dans le répertoire contenant le script **upsset.cgi** pour en limiter l'utilisation. Ce fichier **.htaccess** doit limiter l'utilisation des scripts aux seules machines autorisées. Dans le répertoire **/usr/lib/cgi-bin/nut**, ajoutez un fichier **.htaccess** contenant les lignes suivantes :

```
<Files upsset.cgi>
Order Deny,Allow
deny from all
allow from listes_des_IP_autorisées
</Files>
```

Une fois la configuration sécurisée, la ligne « **I_HAVE_SECURED_MY_CGI_DIRECTORY** » du fichier **upsset.conf** peut être décommentée.

Pour accéder aux pages produites par les CGI, les liens HTML suivants peuvent être utilisés :

```
<a href="/cgi-bin/nut/upsstats.cgi">UPS status viewer</a>
<a href="/cgi-bin/nut/upsset.cgi">UPS administration</a>
```

D'un simple navigateur, vous pouvez maintenant superviser et modifier l'état de vos onduleurs.

Conclusion

Avec cet article, j'espère vous avoir fait découvrir l'intérêt d'utiliser **NUT** pour gérer vos onduleurs. Les clients décrits ici ne sont que ceux inclus dans le projet **NUT**, la page [Clients] liste une dizaine d'autres clients et plugins pour Big Sister, Nagios, Window Maker, KDE, Gnome, Windows, etc. Je ne peux que vous inviter à suivre, utiliser et pourquoi pas participer à ce projet (contacter [AQ]).

Références

- [AQ] Arnaud Quette, responsable développement Linux/UNIX pour MGE UPS SYSTEMS, 2nd développeur **NUT**, mainteneur des paquets Debian (...): quette@debian.org / arnaud.quette@mgeups.com / [@free.fr](http://free.fr)
- [Clients] Liste des clients pour le serveur **upsd**: <http://eu1.networkupstools.org/client-projects>
- [Contrib] Liste des contributeurs aux projets **NUT**: <http://eu1.networkupstools.org/acknowledgements.html>
- [Doc] Documentation officielle du projet **NUT**: <http://eu1.networkupstools.org/doc>
- [Liste] Liste des onduleurs supportés par le projet **NUT**: <http://eu1.networkupstools.org/compat/stable.html>
- [MUS] Page dédiée aux projets open source de MGE UPS SYSTEMS: <http://opensource.mgeups.com>, support: opensource@mgeups.com
- [Norme] Topologies d'UPS et normalisation: <http://www.mgeups.com/techinfo/techpap/articles/0248-f.pdf>
- [NUT] Les miroirs pour l'Europe du site de **NUT**:
 - MGE: <http://eu1.networkupstools.org>
 - Signetic: <http://eu2.networkupstools.org>
- [Readme] **/usr/share/doc/nut/README**
- [SNMP] RFC 1157 - Simple Network Management Protocol et FC 1628 - Uninterruptible Power Supply MIB.
- [USB] USB Power Device Class: http://www.usb.org/developers/devclass_docs/pdcdv10.pdf
- [WinNUT] Client **NUT** pour Windows: <http://csociety.ecn.purdue.edu/~delpha/winnut>