Le HOWTO des onduleurs (UPS)

Harvey J. Stein, `hjstein@math.huji.ac.il`, Berger Financial Research, Ltd. adaptation française B.Choppy, 10 mai 1998 v2.42, 18 novembre 1997

Ce document vous aidera à connecter un onduleur sur une machine Linux... si vous avez de la chance ... Copyright (c) 1994, 1995, 1996, 1997 Harvey J. Stein. Vous pouvez utiliser ce document comme vous l'entendez, tant qu'il reste intact. En particulier, cette note (ainsi que les contributions ci-dessous) doit rester intouchée. NdT : La traduction de ce HOWTO est particulierement délicate, en ce qu'il intègre de nombreux messages échangés sur InterNet reproduits à l'identique. Autant que possible, on aura conservé le sens général de ces messages, sans toutefois les reproduire systématiquement en entier, leur lecture en devenant vite fastidieuse. Dans certains cas, une partie en anglais est conservée, en particulier dans les en-têtes de messages. Les noms de signaux et broches de connexion ont été traduits, mais leur abréviation originelle conservé (par exemple, DCD : Détection de porteuse). Une copie de la notice originale de copyright est conservée, y compris dans les sources des programmes. Le traducteur attire l'attention de ses lecteurs sur le risque d'erreur introduite lors de la traduction des commentaires et messages dans les programmes source, pouvant rendre ceux-ci impossibles à compiler ou inutilisables. La référence en ce cas est, comme toujours, la version originelle du document (en anglais). Version originale de la notice de copyright : You may use this document as you see fit, as long as it remains intact. In particular, this notice (along with the contributions below) must remain untouched.

Contents

1 Introduction

Ce HOWTO concerne la connexion d'un onduleur sur un PC sous Linux. L'idée est d'établir la connexion de telle sorte que Linux puisse s'arrêter proprement lorsque le courant s'arrête.

Cela inclut la référence à des paquetages logiciels existants facilitant l'établissement de ce genre de communications, et la manière dont celles-ci sont réalisées. Ce dernier point est souvent superfétatoire si vous pouvez trouver un paquetage tout configuré pour votre onduleur. Sinon, il vous faudra lire ce qui suit.

Dans une large mesure, le présent document est encore plus redondant que lorsque j'en ai écrit la première version en 1994. Toutes les informations de base ont toujours été présentes dans les pages de man de *powerd* fournies avec le paquetage *SysVinit*. Alors qu'en 1994 il arrivait souvent que les distributions ne comportent même pas lesdites pages de man, je ne crois pas que ce soit encore le cas.

De plus, lorsque j'ai écrit la première version de ce Howto, il n'existait aucun autre logiciel que *powerd* pour la communication et le contrôle entre Linux et les onduleurs. Maintenant, il existe un certain nombre de paquetages de contrôle d'onduleurs dans le répertoire *UPS*

de Sunsite.

Malgré tout, je continue à maintenir le Howto des onduleurs. Pourquoi donc ? Eh bien :

- une seconde source d'informations peut aider à la compréhension de la méthode de connexion de Linux à un onduleur, même s'il s'agit simplement de la même information, écrite différemment ;
- le HOWTO peut servir de repository pour les données spécifiques des onduleurs - de nombreux onduleurs ne sont pas encore gérés par les paquetages généraux ;
- le HOWTO contient des détails supplémentaires qui ne se trouvent pas dans d'autres documents ;
- ce document semble avoir maintenant sa vie propre. La nécessité de création d'un Howto se fait sentir clairement. Il est moins évident de définir l'instant où il doive être mis en sommeil.

1.1 Contributeurs

Je suis débiteur à vie de ceux dont j'ai reçu de l'aide, des suggestions, ainsi que des données spécifiques d'onduleurs. La liste inclut :

- Hennus Bergman (hennus@sky.nl.mugnet.org) ;
- Charli (mephistos@impsat1.com.ar) ;
- Ciro Cattuto (Ciro Cattuto) ;
- Nick Christenson (npc@minotaur.jpl.nasa.gov) ;
- Lam Dang (angit@netcom.com) ;
- Markus Eiden (Markus@eiden.de) ;
- Dan Fandrich (dan@fch.wimsey.bc.ca) ;
- Ben Galliart (bgallia@orion.it.luc.edu) ;
- Danny ter Haar (dth@cistron.nl) ;
- Christian G. Holtje (docwhat@uiuc.edu) ;
- Raymond A. Ingles (inglesra@frc.com) ;
- Peter Kammer (pkammer@ics.uci.edu) ;
- Marek Michalkiewicz (ind43@sun1000.ci.pwr.wroc.pl) ;
- Jim Ockers (ockers@umr.edu) ;
- Evgeny Stambulchik (fnevgenv@plasma-gate.weizmann.ac.il) ;
- Clive A. Stubbings (cas@vjet.demon.co.uk) ;
- Miquel van Smoorenburg (miquels@cistron.nl) ;
- Slavik Terletsky (ts@polynet.lviv.ua) ;
- Tom Webster (webster@kaiwan.com).

Notez que les adresses e-mail apparaissant dans les extraits de courriers ci-après peuvent être obsolètes. Ce qui précède l'est propablement aussi, mais quelques-unes sont plus récentes que ce qui se trouve plus bas.

Mes excuses aussi à quiconque j'aie oublié de citer dans cette liste. Envoyez-moi un e-mail et je vous ajouterai.

1.2 Avertissement important

Je ne peux réellement pas garantir que quelque partie de ceci fonctionne pour vous. Connecter un onduleur à un ordinateur peut être un travail d'astuce. L'un ou l'autre, ou les deux peuvent brûler, exploser, mettre le feu, ou commencer la Troisième Guerre Mondiale. De plus, je n'ai une expérience directe que de l'onduleur Advice 1200 A, et je n'ai pas eu à fabriquer de câble. Donc, *SOYEZ PRUDENT, RASSEMBLEZ TOUTE L'INFORMATION POSSIBLE SUR VOTRE ONDULEUR. REFLECHISSEZ D'ABORD. NE CROYEZ PAS A PRIORI CE QUE VOUS LISEZ ICI NI AILLEURS*.

D'un autre côté, j'ai réussi à tout faire fonctionner avec mes onduleurs, sans beaucoup d'informations des constructeurs, et sans faire fumer quoi que ce soit, donc c'est possible.

1.3 Autres documents

Ce document ne traite pas des fonctions et possibilités générales des onduleurs. Pour ce genre d'informations, voyez la Foire Aux Questions *La FAQ UPS* . Elle peut aussi être trouvée sur *ftp://rtfm.mit.edu/pub/usenet-by-hierarchy/comp/answers/UPS-faq* . Elle est maintenue par *Nick Christenson* , mais semble n'avoir pas été mise à jour depuis 1995. Si vous lui envoyez un e-mail, il souhaiterait qu'apparaisse *UPS* ou *UPS FAQ* ou quelque chose de similaire dans la ligne Subject de votre message.

Il y a aussi de plus en plus de constructeurs d'onduleurs présent sur le Net. Certains d'entre aux fournissent réellement des informations utiles sur leur site Web. Une liste pratique des sites web des constructeurs est disponible sur *Répertoire des onduleurs* . Le même site propose aussi une *FAQ des onduleurs* .

2 Note importante concernant l'obsolescence des informations

Je viens de découvrir qu'une partie de la documentation ci-dessous est obsolète. En particulier, le daemon *init* fourni avec le dernier *paquetage sysinit*

est plus sophistiqué que ce que j'ai décrit. Bien qu'il semble que la compatibilité ascendante soit assurée pour ce qui est écrit ici, il apparaît que certaines fonctions non documentées sont **très importantes** pour la gestion des onduleurs.

Le mécanisme de contrôle indiqué ci-après permet seulement à *powerd* d'envoyer à *init* un des messages *powerfail* ou *powerok*. *init* exécute une commande lorsqu'il reçoit *powerfail* et une autre lorsqu'il reçoit *powerok*. Cela complexifie la logique de *powerd* pour la gestion des signaux de batterie faible et autres sortes de situations spéciales.

Les nouvelles versions d'*init* (depuis la version 2.58, apparemment) sont plus sophistiquées. Il est possible de leur demander d'exécuter un script parmi *trois*. Ainsi, *init* peut avoir un script *powerfail* pour traiter une coupure de courant, un script *powerfailnow* pour réaliser un arrêt immédiat et un script *powerok* pour bloquer tout arrêt en cours. C'est nettement plus propre que les circonvolutions nécessaires avec le mécanisme détaillé plus bas.

Bien qu'une grande partie du document soit fondée sur l'ancienne méthode de communication avec *init*, je viens d'ajouter deux nouvelles sections dans lesquelles les auteurs utilisent la nouvelle méthode. Il s'agit de 8.3 (Trust Energy Protector 400/600) et *APC Smart-UPS 700* . La première est particulièrement détaillée. Les deux comportent un *powerd.c* qui demande à *init* un shutdown immédiat lorsqu'un signal de batterie faible est reçu, ainsi que les lignes correspondantes de */etc/inittab*. Pour tout le reste, je peux juste vous dire de regarder dans le code source de *init*.

Aussi, pour autant que je sache, de nombreux paquetages cités ci-dessous utilisent aussi la nouvelle méthode de communication.

[1]

3 Onduleur bête, onduleur intelligent

Les onduleurs peuvent se classer dans deux catégories : "intelligents" ou "bêtes". La différence entre les deux réside dans la quantité d'informations que l'on peut obtenir de et le niveau de contrôle que l'on peut exercer sur l'onduleur.

Onduleur "bête"

- se connecte à l'ordinateur par le port série ;
- utilise les lignes de contrôle modem pour communiquer avec celui-ci ;
- peut indiquer si le courant est présent ou non ;
- peut typiquement indiquer si la batterie est faible ;
- accepte habituellement un ordre d'arrêt de la part de l'ordinateur.

Onduleur "intelligent"

- se connecte à l'ordinateur par le port série ;
- communique avec celui-ci par transfert de données normal sur le port série ;
- dispose typiquement d'une sorte de langage de commandes que l'ordinateur peut utiliser pour obtenir diverses informations, positionner certains paramètres de fonctionnement et contrôler l'onduleur (pour arrêter celui-ci, par exemple).

Habituellement, les onduleurs intelligents peuvent fonctionner en mode bête. C'est utile, car pour autant que je sache, les entreprises qui construisent les onduleurs les plus populaires (notamment APC) ne diffusent leur protocole de communication qu'aux entreprises qui signent un accord de confidentialité.

Autant que je sache, les seuls onduleurs intelligents avec lesquels il soit simple de communiquer sont ceux faits par Best. De plus, Best documente complètement le mode intelligent (ainsi que le mode bête) et fournit le source de programmes qui communiquent avec leurs onduleurs.

Tous les paquetages indiqués dans la section 4 (Logiciels) communiqueront avec les onduleurs en mode bête. C'est tout ce dont on a réellement besoin. Ceux spécifiques des onduleurs APC annoncent diverses possibilités d'utilisation en mode intelligent, mais je ne sais pas exactement ce qu'ils permettent. Une implémentation complète vous affichera une fenêtre avec toutes sortes de jauges affichant diverses statistiques de l'onduleur, telles que charge, température intérieure, historique des coupures, voltages d'entrée et de sortie, etc. Il semble que le paquetage *smupsd-0.9-1.i386.rpm* (section 4 (Logiciels) se rapproche de cela. Je ne suis pas sûr pour les autres.

Le reste de ce document est essentiellement limité à la configuration de votre système avec un onduleur bête. L'idée générale est à peu près la même avec un onduleur intelligent, mais les détails de fonctionnement de *powerd* et le type de câble sont différents pour un onduleur intelligent.

[1] NdT : Il semble que la plupart des contributeurs à ce Howto s'appuient sur le redémarrage de l'ordinateur (`/sbin/reboot`), couplé à une temporisation, pour réaliser l'extinction de l'onduleur. Cette méthode semble hasardeuse. Le traducteur propose humblement au lecteur d'étudier la possibilité de remplacer la relance complète (`reboot`) par un arrêt système propre (`shutdown -h`) ou moins propre (`halt`). Cette méthode permet d'éviter totalement le risque que l'onduleur s'arrête "trop tard" dans le processus de démarrage de l'ordinateur (i.e. lorsque les systèmes de fichiers sont déjà montés).

4 Logiciels

Fondamentalement, tout ce qu'il vous faut est un exécutable *powerd*, habituellement placé dans */sbin/powerd*. Il fait habituellement partie du paquetage *SysVinit*. Pour autant que je sache, toutes les distributions actuelles de Linux contiennent une version récente de *SysVinit*. Les versions très anciennes ne comportaient pas *powerd*.

Le seul problème que vous puissiez rencontrer est que votre câble ne corresponde pas à la configuration de *powerd*, auquel cas vous devrez, soit rebrocher votre câble, soit trouver une copie de *powerd.c* et le modifier pour le faire fonctionner avec votre câble. Ou, pour cela, vous pouvez toujours utiliser l'un des paquetages suivants, dont de nombreux permettent la configuration du câble.

Comme indiqué, une alternative au *powerd* du paquetage *SysVinit* est l'utilisation de l'un des paquetages disponibles maintenant. il existe de nombreux paquetages qui aident à configurer la communication entre l'ordinateur et un onduleur. Aucun d'entre eux n'était disponible lorsque j'ai écrit ce Howto pour la première fois, c'est pourquoi j'ai eu à l'écrire. En fait, il y a de bonnes chances que vous puissiez utiliser l'un de ces paquetages logiciels et éviter totalement le présent Howto !

Au 15 mars 1997 à peu près, le répertoire *UPS*

de Sunsite disposait d'un certain nombre de paquetages. D'autres sites semblent avoir aussi des paquetages de contrôle d'onduleurs. Voici ce que j'ai trouvé (tous sur Sunsite sauf deux) :

Enhanced APC BackUPS.tar.gz

Un paquetage de contrôle des onduleurs intelligents APC Smart-UPS. Il semble suivre basiquement le BUPS-Howto (Back-UPS-Howto, inclus ci-après), mais semble aussi disposer d'une sorte de signal de batterie faible.

Enhanced APC UPSD-v1.4.tar.gz

Le fichier *.lsm* dit qu'il s'agit du même paquetage que le précédent, sous forme de *.tar.gz* dans un *.tar.gz* ! La documentation est légère. Il semble gérer les onduleurs APC dans les deux modes bête et intelligent, mais je ne peux m'en assurer.

Enhanced APC UPSD-v1.4.tar.gz

Un autre paquetage de contrôle des onduleurs APC Smart-UPS. Semble inclure une sorte de support maître/esclave (i.e. une machine en prévient une autre de s'arrêter lorsque le courant est coupé). Semble utiliser les onduleurs en mode intelligent, par opposition à la bascule des lignes modem.

smupsd-0.9-1.i386.rpm

smupsd-0.9-1.src.rpm

L'auteur (*David E. Myers*) écrit :

smupsd surveille un *APC Smart-UPS[TM]*

sous *Red Hat[TM] Linux* . Si le courant est coupé, smupsd arrêtera le système et l'onduleur de manière correcte.

smupsd a les fonctionnalités suivantes :

- arrêt du système et de l'onduleur en fonction de la charge résiduelle de ce dernier ou du temps écoulé depuis la coupure de courant ;
- surveillance des paramètres de l'onduleur en temps réel depuis toute machine à l'aide du programme graphique upsmon, écrit en Java$TM]$; $tracedesparamtresdel'onduleurdansunfichierpouranalyseetdition$;

- mode maître/esclave permettant à des systèmes additionnels partageant le même onduleur de lire les paramètres de celui-ci sur la machine qui lui est connectée par port série ;
  - contrôle des accès réseau à l'aide du fichier */etc/hosts.allow*.

genpower-1.0.1.tgz

Un paquetage général de gestion d'onduleurs. Inclut des configurations pour beaucoup d'onduleurs - deux pour TrippLite et trois pour APC. Contient une bonne documentation. Un bon achat.

powerd-2.0.tar.gz

Un *powerd* de remplacement de celui du paquetage *SysVinit*. A l'opposé des commentaires de la documentation, il ne semble pas avoir été fusionné avec ce dernier (du moins jusqu'à la version 2.62). Ses avantages résident dans le fait qu'il puisse agir comme serveur pour d'autres *powerd* tournant sur d'autres machines (lorsque plusieurs machines d'un réseau partagent le même onduleur) et être configuré par le biais d'un fichier - le source ne nécessite donc ni édition ni recompilation.

upsd-1.0.tgz

Un autre *powerd* de remplacement. Semble être assez comparable en fonctionnalités avec powerd-2.0.tar.gz.

checkups.tar

Ce paquetage est destiné à contrôler les onduleurs Best. Il provient directement du site Web de Best. Comporte des binaires pour de nombreux *unix* mais, plus important, inclut le code source, il est donc possible de l'essayer sous Linux, et s'il ne fonctionne pas, de tenter de le corriger. Le source inclut aussi bien les "contrôles de base (basic checkups)" que les "contrôles avancés (advanced checkups)" qui sont un peu plus sophistiqués - ils déclenchent un shutdown lorsque l'onduleur indique une durée d'alimentation restante de X minutes, plutôt qu'au bout de Y minutes après la coupure de courant. Le programme de contrôles avancés déclenche aussi sur diverses alarmes telles que "température ambiante élevée", "batterie proche du minimum", "tension de sortie faible" ou "alarme test déclenchée par l'utilisateur".

bestups-0.9.tar.gz

Un paquetage qui peut bien se trouver sur Sunsite à l'instant où vous lisez ceci. C'est une paire de modules de communication qui travaillent avec les onduleurs Best Ferrups. Il gère l'onduleur en mode intelligent. Il inter-opère correctement avec *powerd-2.0* - utile si vous avez un gros Ferrups pour toutes les machines d'un réseau.

Note : ce paquetage doit encore être chargé vers Sunsite. Je continue à presser l'auteur de le finir et de le charger, mais il doit encore en trouver le temps.

LanSafe III

Deltec Electronics (et Exide) vendent un paquetage logiciel appelé LanSafe III. Il existe une version Linux. Il est fourni avec leurs onduleurs. Ils disent qu'il fonctionne aussi avec d'autres onduleurs (en mode bête).

apcupsd-2.8.tar.gz

L'auteur (*Andre Hedrick*) écrit :

apcupsd-2.1.tar.gz remplace Enhanced_APC_UPSD.tar.gz.

C'est un paquetage très complet pour les onduleurs APC. Il gère toute leur gamme. J'ai maintenant ajouté un mode intelligent au paquetage et un support pour les câbles APC ou maison si aucun câble APC n'est géré.

smartups-1.1.tgz

Du fichier *.lsm* :

Un powerd et un utilitaire graphique sous X11 qui vous montre les voltages, fréquences, pourcentages de charge et niveau de batterie en temps réel. Les protocoles “Safeware” et “Tripplite” sont gérés. Source et binaires ELF.

ups.tar.gz

Du fichier *.lsm* :

Programme qui interagit avec les sauvegardes batteries (onduleurs Powerbox).

usvd-2.0.0.tgz

Du fichier *.lsm* :

uvsd est un daemon qui surveille l'état d'un onduleur et réagit aux changements d'états (coupure de courant, retour du courant, batterie faible). Vous pouvez écrire vos propres scripts qui sont appelés dans ces cas. Il ne nécessite *pas* SysVinit.

Notez que j'ai seulement jeté un coup d'oeil aux paquetages. Je ne les ai pas utilisés. Nous étions proches d'utiliser *bestups-0.9.tar.gz*

et *powerd-2.0.tar.gz*

mais nous ne l'avons jamais fait.

5 Faites-le vous-même

Ce chapitre est spécifiquement destiné au contrôle des onduleurs bêtes. Néammoins, une grande partie du processus est à peu près identique pour les onduleurs intelligents. La principale différence réside dans la manière dont le daemon (typiquement *powerd*) de surveilance communique avec l'onduleur.

Avant de faire quoi que ce soit, je suggère l'algorithme suivant :

- parcourir ce document ;
- télécharger et étudier tous les paquetages qui semblent adaptés spécifiquement à son onduleur ;
- télécharger et étudier les paquetages plus génériques. Notes que certains d'entre eux sont en fait plus puissants, mieux documentés et plus faciles d'emploi que leurs équivalents spécifiques ;
- si les choses ne se passent pas bien ou si certains points restent obscurs, lire le présent document avec attention et bidouiller avec ardeur et précaution...

5.1 Que faut-il faire (sommairement) ?

- brancher l'ordinateur sur l'onduleur ;
- connecter le port série de l'ordinateur à l'onduleur avec un câble spécial ;
- lancer *powerd* (ou un de ses équivalents) sur l'ordinateur ;
- configurer *init* pour réaliser quelque chose de raisonnable sur les événements *powerfail* et *powerok* (comme lancer un *shutdown* et tuer tout *shutdown* en cours respectivement, par exemple).

5.2 Comment est-ce supposé fonctionner ?

Travail de l'onduleur

Lorsque le courant s'arrête, l'onduleur continue d'alimenter le PC et signale l'arrêt du courant par bascule d'un relais ou d'un optocoupleur sur son port de contrôle.

Travail du câble

Le câble est conçu de telle manière que lorsque l'onduleur bascule ledit relais, cela monte un signal de contrôle particulier de la ligne série (typiquement *DCD*, détection de porteuse)

Travail de powerd

Le daemon *powerd* contrôle le port série. Il maintient levés/baissés les signaux de contrôle du port série dont l'onduleur a besoin (typiquement *DTR*, Terminal de Données Prêt, doit rester levé, et touts les signaux qui coupent l'onduleur doivent être maintenus baissés). Lorsque *powerd* voit le signal de contrôle de l'onduleur monter, il écrit `FAIL` dans `/etc/powerstatus` et envoie un signal `SIGPWR` au process *init* (les anciennes versions de *powerd* et *init* écrivent dans `/etc/powerfail`). Lorsque le signal de contrôle redescend, il écrit `OK` dans `/etc/powerstatus` et envoie un signal `SIGPWR` à *init*.

Travail de init (en plus de tout ce qu'il fait par ailleurs)

Lorsqu'il reçoit un signal `SIGPWR`, il regarde dans `/etc/powerstatus`. Si celui-ci contient `FAIL`, il exécute l'entrée `powerfail` du fichier `/etc/inittab`. S'il contient `OK`, il exécute l'entrée `powerokwait` de `inittab`.

5.3 Comment configurer tout ça ?

Ce qui suit présuppose que vous disposez d'un câble qui fonctionne correctement avec *powerd*. Si vous n'en êtes pas sûr, voyez la section : 6.2 (Analyse de câbles et modification de powerd.c) pour toute information sur les câbles mal décrits et la reconfiguration de *powerd*. Les sections 6.3 (Assignement des broches du port série) et 6.4 (Correspondance entre ioctl et RS232) seront aussi utiles.

Si vous devez fabriquer un câble, voyez la section : 6.1 (Comment réaliser un câble ?) pour les détails généraux, et la sous-section de : 8 (Informations sur un certain nombre d'onduleurs) qui se rapporte à votre onduleur. Cette dernière peut aussi contenir des informations sur les câbles fournis par le constructeur. Vous voudrez probablement parcourir toute la section 8 (Informations sur un certain nombre d'onduleurs) car chaque section contient quelques détails supplémentaires généralement utiles.

- Editez `/etc/inittab`. Placez-y quelque chose de ce genre :

```
# Que faire si le courant s'arrete
# (arreter le systeme et vider la batterie :) :
pf::powerfail:/etc/powerfailscript +5

# Si le courant revient avant la fin du shutdown, arreter celui-ci
pg:0123456:powerokwait:/etc/powerokscript
```

- Ecrivez les scripts `/etc/powerfailscript` et `/etc/powerokscript` pour arrêter le système après cinq minutes, ou mener toute action appropriée, et tuer le shutdown en cours, respectivement. En fonction de votre version de shutdown, cela sera, soit si trivial que vous n'aurez même pas à écrire de script, soit un script d'une ligne *bash*, quelque chose du genre :

```
kill `ps -aux | grep "shutdown" | grep -v grep | awk '{print $2}'`
```

et vous conserverez les scripts (au cas où cela ne vous arriverait pas dans un parfait état, la première apostrophe sur la ligne ci-dessus est une quote inversée, la seconde et la troisième sont des apostrophes, et la dernière est aussi une quote inversée).

- Dites à *init* de relire le fichier `inittab` avec :

  ```
  telinit q
  ```

- Editez rc.local pour lancer *powerd* lors du lancement. Syntaxe :

  ```
  powerd <ligne>
  ```

  Remplacez <`ligne`> par le port série modem sur lequel sera connecté l'onduleur, comme dans : `/dev/cua1`.

- Connectez le port série du PC à celui de l'onduleur. NE BRANCHEZ PAS ENCORE LE PC SUR L'ONDULEUR.
- Branchez une lampe sur l'onduleur.
- Allumez l'onduleur et la lampe.
- Lancez *powerd*.
- Testez la configuration :
  - Débranchez l'onduleur.
    - Contrôlez que la lampe reste allumée,
    - Contrôlez que `/etc/powerfailscript` est lancé,
    - Contrôlez que le shutdown est lancé.
  - Rebranchez l'onduleur.
    - Contrôlez que la lampe reste allumée,
    - Contrôlez que `/etc/powerokscript` est lancé,
    - Contrôlez que `/etc/powerfailscript` n'est pas lancé,
    - Contrôlez que le shutdown est bien arrêté.
  - Redébranchez l'onduleur. Laissez-le débranché et vérifiez que le PC s'arrête proprement dans un délai correct.
  - **La Partie Délicate.** Une fois que tout semble correct, arrêtez le PC et branchez-le sur l'onduleur. Lancez un script qui synchronise le disque dur toutes les secondes ou à peu près (sync). Simultanément, lancez un second script qui exécute un find sur votre disque entier. Le premier sert à rendre l'opération plus sûre, et le second, à consommer le maximum de puissance. Maintenant, tirez sur la prise de l'onduleur, vérifiez une fois de plus que le PC est lancé, et attendez. Assurez-vous que le PC s'arrête correctement avant que la batterie soit vide. C'est dangereux, car si la batterie n'assure pas le délai d'arrêt du PC, vous pouvez vous retrouver avec un système de fichiers corrompu, et peut-être même la perte de tous vos fichiers. Vous devriez probablement réaliser une sauvegarde complète avant ce test, et positionner un délai de shutdown très court pour commencer.

Félicitations ! Vous avez maintenant un PC sous Linux protégé par onduleur qui va s'arrêter proprement lors d'une coupure de courant !

5.4 Améliorations Utilisateur

- Bidouillez `powerd.c` pour surveiller la ligne indiquant un faible niveau de batterie. Dans ce cas, exécutez un shutdown **immediate** ;
- Modifiez la procédure de shutdown, afin que lorsqu'elle s'exécute dans des conditions de coupure de courant, elle éteigne l'onduleur après avoir effectué tout le nécessaire.

6 Notes sur le matériel

6.1 Comment réaliser un câble ?

Cette section est juste composée de messages que j'ai vus sur le Net. Je ne l'ai pas réalisé, donc je ne peux parler d'expérience. Si quelqu'un le peut, qu'il écrive cette section pour moi :). Voir aussi le message concernant le GPS1000 dans la section 8.11 (GPS1000 d'ACCODATA) pour ne pas citer toutes les données spécifiques de la section 8 (Informations sur un certain nombre d'onduleurs)

```
>From miquels@caution.cistron.nl.mugnet.org Wed Jul 21 14:26:33 1993
Newsgroups: comp.os.linux
Subject: Re: Interface onduleur pour Linux ?
From: miquels@caution.cistron.nl.mugnet.org (Miquel van Smoorenburg)
Date: Sat, 17 Jul 93 18:03:37
Distribution: world
Organization: Cistron Electronics.

Dans l'article <1993Jul15.184450.5193@excaliber.uucp>
joel@rac1.wam.umd.edu (Joel M. Hoffman) ecrit :
>Je ne vais pas tarder a acheter un onduleur, et ai remarque que certains
>d'entre eux ont des interfaces reseau pour prevenir celui-ci lorsque le
>courant est coupe.
>
>Y a-t-il une telle interface pour Linux ?
>
>Merci..
>
>-Joel
>(joel@wam.umd.edu)
>

Lorsque je travaillais sur la derniere version de SysVinit (2.4
actuellement), j'ai eu temporairement un onduleur sur mon ordinateur,
donc j'ai ajoute le support de celui-ci.
Tu as peut-etre vu que dans le dernier fichier d'en-tete <signal.h>,
il y a maintenant un #define SIGPWR 30 :-).
Malgre tout, je n'avais pas une telle interface speciale,
mais la sortie de nombreux onduleurs est juste un relais qui s'ouvre ou se
ferme en cas de coupure de courant.
J'ai reflechi a une methode simple pour connecter ca sur la ligne DCD du
port serie.
Dans le paquetage SysVinit, il y a un demon appele "powerd" qui garde
un oeil sur cette ligne serie et envoie SIGPWR a init lorsque l'etat
change, pour qu'init puisse faire quelque chose (comme arreter le systeme
dans les 5 minutes).
```

```
La methode de connexion de l'onduleur a la ligne serie est decrite dans le
source "powerd.c", mais je vais le dessiner ici pour explications :

                     +------------------------o  DTR
                     |
                   +---+
                   |   | resistance
                   |   | 10 kilo-Ohm
                   |   |
                   +---+                            Vers le port serie
                     |
       +-----o-------+------------------------o  DCD
       |             |
       o  relais     |
     \     de l'     |
      \   onduleur   |
       |             |
       +-----o-------+------------------------o  GND

Joli dessin, hein ?

J'espère que cela peut etre utile.
SysVinit peut etre trouve sur sunsite (et tsx-11 probablement) dans
SysVinit2.4.tar.z

Mike.
```

-

```
Miquel van Smoorenburg, <miquels@cistron.nl.mugnet.org>
Ibmio.com: cannot open CONFIG.SYS: file handle broke off.


>From danny@caution.cistron.nl.mugnet.org Wed Jul 21 14:27:04 1993
Newsgroups: comp.os.linux
Subject: Re: Interface onduleur pour Linux ?
From: danny@caution.cistron.nl.mugnet.org (Danny ter Haar)
Date: Mon, 19 Jul 93 11:02:14
Distribution: world
Organization: Cistron Electronics.

Dans l'article <9307174330@caution.cistron.nl.mugnet.org>
miquels@caution.cistron.nl.mugnet.org (Miquel van Smoorenburg) ecrit :
>La methode de connexion de l'onduleur a la ligne serie est decrite dans le
>source "powerd.c", mais je vais le dessiner ici pour explications :

Le dessin n'etait pas vraiment clair, utilisez plutot celui-ci !
>
>                     +------------------------o  DTR
>                     |
>                   +---+
>                   |   | resistance
>                   |   | 10 kilo-Ohm
>                   |   |
>                   +---+                           Vers le port serie
```

```
>                     |
>       +-----o-------+------------------------o  DCD
>       |
>       o  relais
>     \     de l'
>      \   onduleur
>       |
>       +-----o--------------------------------o  GND
>

Le DTR est maintenu haut.
Lorsque le courant de l'onduleur s'arrete, le relais se ferme.
L'ordinateur controle la descente de la ligne DCD.
Lorsque cela arrive, il lance une sequence shutdown...

-----
Danny

--
<====================================================================>
Danny ter Haar  <dannyth@hacktic.nl> or <danny@cistron.nl.mugnet.org>
Robins law #103: 'a couple of lightyears can't part good friends'
```

6.2 Analyse de câbles et modification de powerd.c

Essayez d'obtenir la documentation des câbles que votre revendeur d'onduleurs fournit. En particulier, recherchez :

- quelles lignes doivent être maintenues hautes ;
- quelle(s) ligne(s) éteint(gnent) l'onduleur ;
- quelles lignes l'onduleur modifie pour indiquer que :
  - le courant est coupé,
  - la batterie est faible.

Il vous faut ensuite modifier `powerd.c` en conséquence, ou utiliser l'un des paquetages configurables cités plus haut (voir *genpower-1.0.1.tgz*, *power-2.0.tar.gz* ou *upsd-1.0.tgz* décrits dans la section 4 (Logiciels). Si vous utilisez l'un des paquetages, suivez les instruction correspondantes. Si vous voulez bidouiller `powerd.c`, lisez ce qui suit.

Si vous avez des problèmes pour obtenir les informations précitées, ou si vous voulez juste les contrôler (une *bonne* idée), le programme suivant peut vous y aider. C'est une version bidouillée de powerd.c. Il vous permet de positionner les signaux du port depuis la ligne de commande, puis il contrôle le port, en affichant l'état des signaux chaque seconde. Je l'ai utilisé en "upscheck /dev/cua1 2" (par exemple) pour monter le deuxième bit (*DTR*) et descendre les autres. Le nombre en base 2 indique les bits à monter, ainsi par exemple pour monter les bits 1, 2 et 3 (et descendre les autres), utilisez 7. Voir le code pour les détails.

Voici le programme (non testé) upscheck.c. Il n'est pas testé car j'ai modifié la version que j'avais utilisée au départ pour le rendre plus clair, et que je ne peux tester la nouvelle version pour le moment.

NdT : La traduction des commentaires et messages peut aussi avoir altéré le comportement du programme.

```
/*
 * upscheck     Controle comment l'ordinateur et l'onduleur communiquent
 *
 * Usage:       upscheck <peripherique> <bits a monter>
 *              Par exemple, upscheck /dev/cua4 4 pour monter le bit 3 et
 *              controler /dev/cua4.
 *
 * Author:      Harvey J. Stein <hjstein@math.huji.ac.il>
 *              (mais en realite juste une modification mineure de Miquel van
 *              Smoorenburg's <miquels@drinkel.nl.mugnet.org> powerd.c
 *
 * Version:     1.0 19940802
 *
 */
#include <sys/types.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <errno.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <signal.h>

/* Programme principal. */
int main(int argc, char **argv)
{
  int fd;

/*  Ces parametres TIOCM_* sont definis dans <linux/termios.h>, qui  */
/*  est inclus indirectement ici.                                    */
  int dtr_bit = TIOCM_DTR;
  int rts_bit = TIOCM_RTS;
  int set_bits;
  int flags;
  int status, oldstat = -1;
  int count = 0;
  int pc;

  if (argc < 2) {
        fprintf(stderr, "Usage: upscheck <peripherique> <bits-a-positionner>\n");
        exit(1);
  }

  /* Ouvre le peripherique a controler. */
  if ((fd = open(argv[1], O_RDWR | O_NDELAY)) < 0) {
    fprintf(stderr, "upscheck: %s: %s\n", argv[1], sys_errlist[errno]);
    exit(1);}

  /* Recupere les bits a positionner sur la ligne de commande */
  sscanf(argv[2], "%d", &set_bits);

  while (1) {
    /* Positionne les bits specifies sur la ligne de commande (et */
    /* seulement eux).                                           */
```

```
    ioctl(fd, TIOCMSET, &set_bits);
    fprintf(stderr, "Positionnement de %o.\n", set_bits);

    sleep(1);

    /* Recupere les bits actuellement positionnes */
    ioctl(fd, TIOCMGET, &flags);
    fprintf(stderr, "Les signaux sont %o.\n", flags);

/*  Piochez ici en changeant TIOM_CTS par un autre TIOCM jusqu'a   */
/*  ce que le programme detecte que le courant est coupe lorsque   */
/*  vous debranchez l'onduleur. Ensuite, vous saurez comment       */
/*  modifier powerd.c                                              */
    if (flags & TIOCM_CTS)
      {
        pc = 0 ;
        fprintf(stderr, "Le courant est la.\n");
      }
    else
      {
        pc = pc + 1 ;
        fprintf(stderr, "Le courant est coupe.\n");
      }
    }

  close(fd);
}
```

6.3 Assignement des broches du port série

La section qui précède présuppose la connaissance de la correspondance entre les signaux de terminal et les broches du port série. Voici une référence de cette correspondance, reprise du document de David Tal : "Câbles et connecteurs fréquemment utilisés". J'inclus un diagramme illustrant les connecteurs, et une table donnant la corresopondance entre les numéros de broches et les signaux de ligne de terminal.

Si vous avez besoin d'une référence générale sur le brochage de câbles, celle de David Tal en est une bonne, mais je n'arrive plus à localiser ce document sur le Net. Mais j'ai trouvé un bon livre de remplacement, c'est *The Hardware Book* .

NdT : si un lecteur français veut proposer une référence dans la langue de Molière, qu'il n'hésite pas à me contacter.

Autres sites utiles :

- *Yost Serial Device Wiring Standard*

  qui contient des informations intéressantes sur les connecteurs RJ-45 et les câbles quatre paires pour toutes les connexions série ;

- *Stokely consulting*

  pour l'information générale sur Unix et en particulier leur Unix Serial Port Resources ;

- *Unix Workstation System Administration Education Certification*

  qui contient : *RS-232: Connectors and Cables* .

DB-25 Broche	DB-9 Broche	Nom	EIA	CCITT	DTE-DCE	Description
1		FG	AA	101	—	Masse de chassis GND
2	3	TD	BA	103	—>	Données transmises, TxD
3	2	RD	BB	104	<—	Données reçues, RxD
4	7	RTS	CA	105	—>	Requête pour envoyer
5	8	CTS	CB	106	<—	Prêt à envoyer
6	6	DSR	CC	107	<—	Jeu de données prêt
7	5	SG	AB	102	—-	Masse de signal, GND
8	1	DCD	CF	109	<—	Détection de porteuse
9		–	–	-	-	Tension positive continue de test
10		–	–	-	-	Tension négative continue de test
11		QM	–	-	<—	Mode d'égalisation
12		SDCD	SCF	122	<—	Détection de porteuse secondaire
13		SCTS	SCB	121	<—	Prêt à envoyer secondaire
14		STD	SBA	118	—>	Données transmise secondaires
15		TC	DB	114	<—	Signal d'horloge de l'émetteur
16		SRD	SBB	119	<—	Signal d'horloge secondaire du récepteur
17		RC	DD	115	—>	Signal d'horloge du récepteur
18		DCR	–	-	<—	Horloge divisée du récepteur
19		SRTS	SCA	120	—>	Requête pour émettre secondaire
20	4	DTR	CD	108.2	—>	Terminal de données prêt
21		SQ	CG	110	<—	Détection de qualité de signal
22	9	RI	CE	125	<—	Indicateur de sonnerie
23		–	CH	111	—>	Sélecteur de vitesse de données
24		–	CI	112	<—	Sélecteur de vitesse de données
25		TC	DA	113	<—	Horloge transmise

Table 1: Assignement des broches de port série (RS-232C), DB-25 et DB-9

Incidemment, il semble que le paquetage Linuxdoc-sgml ne formate plus les tableaux très bien en sortie *html*. Si vous voulez pouvoir lire la table qui suit, vous devrez probablement vous référer à la version *DVI* ou texte simple du présent document.

```
      1                         13         1         5

    -------------------------------      ---------------
    \  . . . . . . . . . . . . .  /      \  . . . . .  /     Connecteurs
     \  . . . . . . . . . . . .  /        \  . . . .  /      RS-232 vus de
      ---------------------------          -----------       l'arrière de
      14                      25            6       9        l'ordinateur

   DTE : Equipement terminal de données (i.e. ordinateur)
   DCE : Equipement de communication de données (i.e. modem)
   RxD : Données reçues; 1 est transmis "bas", 0 "haut"
   TxD : Données envoyées; 1 est transmis "bas", 0 "haut"
   DTR : DTE annonce qu'il est alimenté et prêt à communiquer
   DSR : DCE annonce qu'il est prêt à communiquer; "bas" raccroche le modem
   RTS : DTE demande à DCE la permission d'envoyer des données
   CTS : DCE agrée la RTS
   RI  : DCE indique au DTE qu'il tente d'établir une connexion
```

```
DCD : DCE annonce qu'une connexion est établie
```

6.4 Correspondance entre ioctl et RS232

Puisque vous pouvez aussi devoir modifier powerd.c pour monter et descendre les signaux corrects, vous pouvez aussi avoir besoin des valeurs numériques des différents signaux de terminal. Ils peuvent être trouvés dans `/usr/include/linux/termios.h`, mais sont reproduits ici comme référence. Puisqu'ils peuvent être sujets à changements, vous auriez avantage à les vérifier avec ledit fichier.

```
/* lignes modem */
#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
```

Notez que la troisième colonne est en hexadécimal.

7 Que faire si l'on n'en sort pas ?

Voici une nouvelle solution pour le contrôle lorsque l'onduleur et l'ordinateur ne s'entendent pas. Je dois dire qu'à chaque fois que je lis cela, je suis effaré de l'intelligence de cette solution.

```
From: " Raymond A. Ingles" <inglesra@frc.com>
To: hjstein@math.huji.ac.il
Subject: UPS HOWTO tip
Date: Mon, 24 Feb 1997 11:48:32 -0500 (EST)


Je ne sais pas si d'autres trouveront ca utile, mais je pense pouvoir
diffuser ceci pour inclusion possible dans le HOWTO. Merci de maintenir
un HOWTO que je trouve si utile !

--------------------

Ma fiancee m'a offert un onduleur, un Tripp-Lite 400, je crois.
Il etait le bienvenu et semble fonctionner comme prevu, mais malheureusement,
ne dispose pas d'interface serie pour prevenir l'ordinateur d'une coupure
de courant.
Il semble prevu pour une utilisation personnelle quand l'ordinateur ne
reste pas seul.

Evidemment, cela etait inacceptable et j'ai commence a travailler sur un
systeme de surveillance de ligne, en imaginant ouvrir la boite et voir
comment ajouter le hard que le constructeur avait omis. J'ai alors realise
qu'il y avait une solution plus simple et moins chere (bien qu'un peu moins
```

```
dotee en fonctionnalites).

J'avais un vieux modem 2 400 baud que je n'utilisais pas, que j'ai branche
sur un port serie inutilise de l'ordinateur. Je l'ai ensuite branche sur une
prise anti-surtensions, elle-meme branchee sur la prise murale.
J'ai configure powerd avec les options suivantes :

----
serialline   /dev/ttyS1
monitor      DCD
failwhen     low
----

Maintenant, lorsque le courant est coupe (ou, bien que cela ne soit pas
arrive recemment, lorsque je debranche le parasurtenseur pour tester la
configuration), le modem tombe mais l'onduleur commence a alimenter
l'ordinateur.
Lorsque powerd se rend compte que le modem a descendu DCD, il declenche la
sequence powerfail.

Evidemment, il y a certaines limitations.
Il n'est pas possible de faire indiquer par le modem que la batterie est
faible, etc.
On peut seulement indiquer que le courant est coupe.
Maintenant, ce n'est pas cher et je deteste laisser un equipement
informatique inutilise.
Ces temps-ci, il est possible d'avoir un modem 2 400 baud quasi gratuitement.

Je continue a conseiller un onduleur avec des possibilites de communication
completes, mais si l'on est coince avec un qui n'en a pas, cela peut au
moins etre utile.

  Sincerement,

Ray Ingles           (810) 377-7735           inglesra@frc.com

 "Anybody who has ever seen a photograph showing the kind of damage that
a trout traveling that fast can inflict on the human skull knows that
such photographs are very valuable. I paid $20 for mine." - Dave Barry
```

8 Informations sur un certain nombre d'onduleurs

Cette section contient des informations spécifiques de certains onduleurs. Ce que je souhaiterais serait de disposer des informations sur le port de contrôle de l'onduleur (ce que fait chaque broche et ce qu'elle attend qui soit fait), sur le câble fourni par le constructeur (ce qu'il connecte et où), ainsi qu'une version modifiée de powerd.c qui fonctionne avec l'onduleur. Ce que j'ai actuellement est une description à peu près complète de chaque onduleur. Je voudrais essayer d'affiner chaque information, mais comme je ne peux tester chaque onduleur, il est difficile de décider exactement de ce qui est nécessaire. De plus, chaque onduleur semble avoir quelques trucs supplémentaires qui sont bien décrits par les auteurs de chaque section. Ainsi, pour l'heure, je laisse tout en place. Tout pour un HOWTO épais.

Veuillez m'envoyer vos expériences pour les inclure ici.

8.1 Expériences générales.

J'ai conservé les commentaires des gens, mais n'ai pas encore obtenu la permission de les inclure ici. Voici un sommaire général de ce que j'ai entendu dire.

APC :

Ne donneront pas d'informations sur leur mode "intelligent" sans votre signature d'un accord de confidentialité. Donc, les gens sont forcés d'utiliser leurs onduleurs "intelligents" en mode "bête", comme souligné plus bas. Diverses tentatives de rétro-ingénierie ont été soldées par des niveaux de réussite différents.

Best :

Serviables et aimables. Fournissent le code source et la documentation pour les deux modes.

Tripp Lite :

Une personne a dit que Tripp ne diffuserait pas non plus d'information.

Upsonic :

Quelqu'un a dit qu'Upsonic a discuté de détails techniques au téléphone, répondu aux questions par fax et est serviable en général.

8.2 Advice 1200 A

Onduleurs d'Advice Electronics, Tel Aviv, Israël (Tout leur matériel porte une étiquette à leur nom).

Spécification des broches du port de contrôle.

- 2 - Coupure de courant.
- 5 - Batterie faible.
- 6 - Extinction de l'onduleur.
- 4 - Masse commune des broches 2, 5 et 6.

Ils m'ont aussi donné le dessin suivant qui ne m'a servi à rien, mais peut vous être utile si vous souhaitez fabriquer vous-même un câble :

```
              2 ----------+
                          |
                          \
                           \|
                            |--------------
                           /|
                         \/      <--- Le "\/" indique le type de ce
                         |            transistor. J'ai oublié ce que
                         |            cela veut dire, mais ce n'est
                      +-----+         pas fondamental.
                     /  /  /


              5 ----------+
                          |
                          \
```

```
             \|
              |-------------
             /|
           \/
           |
           |
        +-----+
       /  /  /


              +------------
              |
              /
     10K    |/
6 --\/\/\/--|
            |\
              \/
              |
              |
           +-----+
          /  /  /


4 ----------+
            |
            |
         +-----+
        /  /  /
```

Câble fourni.

Ils m'ont d'abord donné un câble qui appartenait à un paquetage DOS de contrôle de l'onduleur appelé RUPS. Je l'ai utilisé pour les tests. Une fois ceux-ci satisfaisants, ils m'ont donné un câble qu'ils utilisent pour les serveurs Netware connectés à des onduleurs. Il fonctionnait à l'identique. Voici les détails :

- DTR - Alimentation du câble (powerd.c doit le monter) ;
- CTS - Courant présent (descend quand le courant est coupé) ;
- DSR - Batterie faible (descend lorsque la batterie faiblit) ;
- RTS - Extinction de l'onduleur (à monter pour éteindre).

(le powerd.c inclus dans SysVinit place ou laisse RTS haut, causant l'arrêt de l'onduleur immédiatement lors du lancement de powerd !)

8.3 name="Trust Energy Protector 400/600"

Cette section n'est pas utile seulement pour le Trust Energy Protector. Elle illustre les nouvelles fonctionnalités d'*init*.

Comment utiliser un Trust Energy Protector 400/650 sous Linux ?

par *Ciro Cattuto*

Version 1.0 - 31 mars 1997

Connexion PC-onduleur

Le Trust Energy Protector 400/650 est équipé d'un port de signaux. A l'aide d'un câble adapté, il est possible de connecter celui-ci sur un ordinateur pour réagir aux événements concernant l'alimentation électrique.

Le port de signaux de l'onduleur

L'assignement des broches du port de signaux DB-9 de l'onduleur est le suivant, comme indiqué dans le manuel utilisateur :

broche 2
Ce relais est fermé lorsque le courant d'alimentation est coupé.

broche 4
Masse des broches 2 et 5.

broche 5
Ce relais est fermé lorsque la batterie dispose de moins d'une minute et demi d'autonomie.

broche 6
L'utilisateur peut envoyer un signal haut (+5V à +12V) durant plus d'une milliseconde pour éteindre l'onduleur. Cette option ne peut être activée que durant une coupure de courant.

broche 7
Masse de la broche 6.

le câble

Voici le câble que j'ai utilisé pour connecter l'onduleur au port série de mon ordinateur.

```
cote ordinateur (DB-15)                          cote onduleur (DB-9)
====================================================================

 6 DSR --+                                 [R] = resistance 10 kohm
         |
20 DTR --+----+
         |    |
        [R]  [R]                                             +--- 7
         |    |                                              |
 8 DCD --+----|--------------            ---------------------|--- 2
              |                                              |
 7 GND -------|--------------            ---------------------+--- 4
              |                 ...
 5 CTS -------+--------------            ------------------------- 5

 2 TX  ----------------------            ------------------------- 6

====================================================================
```

Pour un port série DB-9, les broches 6, 20, 8, 7, 5 et 2 correspondent respectivement aux broches 6, 4, 1, 5, 8 et 3.

Comment fonctionne le câble ?

L'ordinateur monte `DTR` et vérifie que `DSR` soit haut pour s'assurer que le câble soit connecté à l'ordinateur. Tant que le courant est là, `DCD` et `CTS` sont hauts tous les deux (à cause des résistances).

Lorsque le courant est coupé, le relais entre les broches 2 et 4 de l'onduleur se ferme, et `DCD` descend pour signaler la coupure.

De même, lorsque les batteries sont faibles, le relais entre les broches 5 et 4 se ferme, faisant descendre `CTS`.

Durant une coupure de courant, l'ordinateur peut éteindre l'onduleur en montant `TX` durant `1 ms` au moins. Cela peut être réalisé aisément en envoyant un octet `0xFF` au port série avec une vitesse faible.

le daemon powerd

Pour utiliser les informations disponibles sur le port série, il faut utiliser un programme qu surveille celui-ci, décode le signal et envoie les messages appropriés au système d'exploitation, en l'occurence au processus `init`. Ce dernier peut exécuter des scripts et programmes conçus pour gérer (proprement !) l'événement de coupure de courant.

compiler powerd

En annexe A se trouve le code de `powerd`, le daemon que j'utilise pour surveiller le Trust Energy Protector 400/650. Pour le compiler, il faut le source du paquetage SysVinit (j'ai utilisé celui de sysvinit-2.60). Ecrasez simplement le `powerd.c` d'origine et compilez-le.

Comment fonctionne powerd ?

Dès le démarrage, *powerd* ouvre le périphérique série connecté à l'onduleur et monte `DTR`. Ensuite, il forke un daemon et se termine en laissant celui-ci tourner. Le daemon *powerd* peut se trouver dans l'un des trois états suivants :

Etat 0 - le courant est bon

Dans cet état, *powerd* lit le port série toutes les `T0_SLEEP` secondes (voir les lignes `#define` au début du code source). Si `DCD` descend, *powerd* bascule en état 1. Si `CTS` descend, *powerd* bascule en état 2 (cela ne doit pas arriver si `DCD` n'est pas descendu avant, mais j'ai préféré assurer le coup).

Etat 1 - le courant est coupé

Une coupure de courant a été détectée. `DCD` est bas et *powerd* lit le port de l'onduleur toutes les `T1_SLEEP` secondes. Si `DCD` remonte, il bascule en état 0. Si `CTS` tombe, il bascule en état 2.

Etat 2 - la batterie est faible

La batterie de l'onduleur est faible. Le daemon *powerd* reste dans cet état.

A chaque changement d'état de *powerd*, il prévient le processus *init* afin que l'action appropriée soit effectuée. Ces événements sont tracés à l'aide du système de trace du système d'exploitation (NdT : *syslogd*).

Si `DSR` est bas, c'est qu'il y a un problème au niveau du câble. *powerd* continue à surveiller la ligne `DSR` et envoit un message d'avertissement toutes les deux minutes au système de trace.

Utiliser powerd

Le daemon *powerd* doit être lancé par les scripts d'initialisation durant le démarrage du système. J'ai ajouté les lignes suivantes dans mon script `/etc/rc.d/rc.local` :

```
# Ajout du support de l'onduleur
echo "Demarrage du processus powerd..."
rm -f /etc/turnUPSoff
stty -crtscts speed 75 < /dev/cua3 > /dev/null
if [ -x /usr/sbin/powerd ]
then
        /usr/sbin/powerd /dev/cua3
fi
```

En premier, on efface (si nécessaire) le fichier `/etc/turnUPSoff`. Celui-ci est utilisé par le script de shutdown (`/etc/rc.d/rc.0` dans mon cas) pour décider s'il faut arrêter l'onduleur ou non. Voir plus bas pour plus d'informations.

Ensuite, on désactive le contrôle de flux matériel sur le périphérique série connecté à l'onduleur et on positionne la vitesse à 75 bauds. Maintenant, nous sommes sûr que le signal `TX` restera haut suffisamment longtemps pour arrêter l'onduleur si nous envoyons un caractère `0xFF` au port série (à nouveau, voir plus bas).

Enfin, nous lançons le daemon *powerd* en lui indiquant le port à surveiller. Notez que nous n'avons pas à lire de caractères sur ce port, donc pas d'inquiétude en cas de conflit d'interruptions - il n'aura aucune influence.

Le fichier inittab et les scripts de shutdown

Le processus *powerd* tourne maintenant, et il enverra des signaux à `init` en cas de coupure de courant. Il faut maintenant configurer le système afin qu'il puisse réagir de manière utile lorsque ces signaux sont reçus.

Modification de inittab

Ajoutez les lignes suivantes à proximité du début de votre fichier `/etc/inittab` :

```
# Quoi faire lorsque le courant est coupe (shutdown temporise)
pf::powerfail:/etc/powerfail_script

# Si le courant revient avant le shutdown, arreter celui-ci
pg::powerokwait:/etc/powerokay_script

# Si la batterie de l'onduleur est faible, faire un shutdown immediat
pc::powerfailnow:/etc/powerfailnow_script
```

Les scripts

Les scripts `powerfail_script`, `powerokay_script` et `powerfailnow_script` sont exécutés lorsque *init* reçoit le signal correspondant. Il ont la responsabilité d'arrêter le système de manière propre ou d'arrêter un shutdown en cours au cas où le courant reviendrait. Voici les scripts que j'utilise actuellement :

`/etc/powerfail_script`

```
#!/bin/sh
/bin/sync
/usr/bin/sleep 10m
kill -9 `ps auxw | \
         grep "shutdown" | \
         grep -v grep | \
         awk '{print $2}'` >/etc/turnUPSoff
/sbin/shutdown -t30 -h +3 "Coupure de courant"
```

Mon Trust Energy Protector 400 n'alimente que l'ordinateur, j'ai donc une réserve de courant assez importante. Dans mon secteur, les coupures de courant ne durent souvent que quelques minutes, donc le système réagit à celles-ci de la manière suivante : Il attent 10 minutes (habituellement, le courant revient avant) puis arrête le système, en laissant aux utilisateurs le temps de fermer leurs applications et de se déconnecter. Avant d'exécuter la commande *shutdown*, je vérifie qu'il n'y a pas d'autre shutdown en cours. Je crée aussi le fichier `/etc/turnUPSoff` afin que le système arrête l'onduleur.

`/etc/powerokay_script`

```
#!/bin/sh
kill    `ps auxw | \
         grep "powerfail_script" | \
         grep -v grep | \
         awk '{print $2}'`
kill -9 `ps auxw | \
         grep "shutdown" | \
         grep -v grep | \
         awk '{print $2}'`
rm -f /etc/turnUPSoff
```

Si le courant revient, on tue le script *powerfail_script* et tout *shutdown* en cours. On n'oublie pas de supprimer `/etc/turnUPSoff`.

`/etc/powerfailnow_script`

```
#!/bin/sh
kill -9 `ps auxw | \
         grep "shutdown" | \
         grep -v grep | \
         awk '{print $2}'` >/etc/turnUPSoff
/sbin/shutdown -h now "Batterie de l'onduleur faible. ARRET IMMEDIAT."
```

Si la batterie faiblit, on s'assure qu'aucun *shutdown* ne soit en cours, on crée le fichier `/etc/turnUPSoff` puis on arrête le système immédiatement.

Le script d'arrêt système

Lorsque l'arrêt du système est effectué, on peut arrêter l'onduleur en montant le signal `TX` du port série durant plus d'une milliseconde. Celui-ci est déjà configuré correctement par la commande `stty` du script `rc.local`. Si le fichier `/etc/turnUPSoff` est présent, on envoit l'octet `0xFF` (tous les bits à 1) sur le port série.

Pour cela, on ajoute les lignes suivantes autour de la fin du script d'arrêt (`/etc/rc.d/rc.0` dans mon cas). L'emplacement correct dépend de la manière dont le système est configuré, mais il doit pouvoir se situer avant la commande *echo* qui affiche le message “System is halted”.

```
# Est-on dans un cas de coupure de courant ?
if [ -f /etc/turnUPSoff ]
then
        echo "Arret de l'onduleur"
        sleep 5
        echo -e "\377" >/dev/cua3
        exit 1
fi
```

Remarques générales

Ce document contient des choses que j'ai apprises en tentant de configurer *mon* système Linux avec le Trust Energy Protector 400. Certaines informations (le chemin d'accès aux scripts d'initialisation, par exemple) peuvent être spécifiques à mon système, et il vous faudra vraisemblablement faire quelques adaptations. Néammoins, j'espère que ce document sera une trace utile pour ceux qui essaieront d'utiliser un onduleur de ce type sous Linux. Si vous rencontrez des difficultés, recherchez des informations plus générales dans le reste de ce Howto. Bonne chance !

Retour d'informations

J'apprécierais énormément tout retour d'informations concernant ce document, afin de pouvoir affiner celui-ci et y corriger de possibles erreurs (je sais que l'anglais que j'utilise n'est pas excellent, mais après tout, je suis italien ![2]). Envoyez tout commentaire/suggestion/critique à l'adresse suivante :

ciro@stud.unipg.it

Si vous rencontrez des problèmes d'utilisation de l'onduleur Trust Energy Protector 400/650 sous Linux, vous pouvez aussi me contacter. J'essaierai de vous aider.

Informations légales

Je n'ai aucune relation avec Trust Networking Products.

L'information contenue dans ce document est livrée "telle quelle". Vous pouvez l'utiliser à vos risques et périls. Je ne puis être tenu responsable d'un quelconque dommage ni perte de données résultant de l'utilisation du code ni des informations données ici.

Ciro Cattuto

Appendix A - Code source du daemon powerd>

```
powerd.c

/*
 * powerd       Recoit les evenements de coupure de courant
 *              depuis un Trust Energy Protector 400/650
 *              et previent init
 *
 * Usage:       powerd <port serie>
 *
 * Author:      Ciro Cattuto <ciro@stud.unipg.it>
 *
 * Version 1.0 - 31 Mars 1997
 *
 * Ce code est largement fonde sur le powerd.c original de
 * Miquel van Smoorenburg <miquels@drinkel.ow.org>.
 *
 *
 * 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.
 *
 * Ce programme est un logiciel libre ; vous pouvez le distribuer
 * et/ou le modifier selon les termes de la Licence Publique Generale
 * GNU publiee par la Free Software Foundation version 2 ou (comme
 * vous le voulez) toute version ulterieure.
 *
 */

/* etat 0 - le courant est la */
#define T0_SLEEP        10      /* intervalle de lecture du port en
                                   secondes                            */
#define T0_DCD          3       /* duree avec DCD monte avant de realiser
                                   une action                          */
#define T0_CTS          3       /* duree avec CTS monte avant de realiser
                                   une action                          */
```

[2] NdT : On se demande quelquefois s'il faut vraiment tout traduire :-))

```
/* etat 1 - le courant est coupe */
#define T1_SLEEP        2       /* intervalle de lecture du port          */
#define T1_DCD          3       /* idem    T0_DCD                          */
#define T1_CTS          3       /* idem    T0_CTS                          */

#define DSR_SLEEP       2
#define DSR_TRIES       60

/* On utilise le nouveau mode de communication avec init. */
#define NEWINIT

#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <errno.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <signal.h>
#include <syslog.h>
#include <string.h>
#include "paths.h"
#ifdef NEWINIT
#include "initreq.h"
#endif

#ifndef SIGPWR
#  define SIGPWR SIGUSR1
#endif

#ifdef NEWINIT
void alrm_handler()
{
}
#endif

/* Dire a init que le courant est coupe (1), revenu (0) ou que
   les batteries de l'onduleur sont faibles (2). */
void powerfail(int event)
{
  int fd;
#ifdef NEWINIT
  struct init_request req;

  /* On remplit la structure necessaire */
  memset(&req, 0, sizeof(req));
  req.magic = INIT_MAGIC;
  switch (event)
        {
        case 0:
                req.cmd = INIT_CMD_POWEROK;
                break;
        case 1:
                req.cmd = INIT_CMD_POWERFAIL;
```

```
                break;
        case 2:
        default:
                req.cmd = INIT_CMD_POWERFAILNOW;
        }

  /* On ouvre le fifo (avec timeout) */
  signal(SIGALRM, alrm_handler);
  alarm(3);
  if ((fd = open(INIT_FIFO, O_WRONLY)) >= 0
                && write(fd, &req, sizeof(req)) == sizeof(req)) {
        close(fd);
        return;
  }
  /* On revient a l'ancienne methode... */
#endif

  /* On cree un fichier info pour init */
  unlink(PWRSTAT);
  if ((fd = open(PWRSTAT, O_CREAT|O_WRONLY, 0644)) >= 0) {
  switch (event)
        {
        case 0:
                write(fd, "OK\n", 3);
                break;

        case 1:
                write(fd, "FAIL\n", 5);
                break;

        case 2:
        default:
                write(fd, "LOW\n", 4);
                break;
        }
  close(fd);
  }

  kill(1, SIGPWR);
}

/* Programme principal. */
int main(int argc, char *argv[])
{
  int fd;
  int dtr_bit = TIOCM_DTR;
  int flags;
  int DCD, CTS;
  int status = -1;
  int DCD_count = 0, CTS_count = 0;
  int tries;

  if (argc < 2) {
        fprintf(stderr, "Usage: powerd <peripherique>\n");
        exit(1);
```

```
}

/* On demarre syslog. */
openlog("powerd", LOG_CONS|LOG_PERROR, LOG_DAEMON);

/* On ouvre le port a surveiller. */
if ((fd = open(argv[1], O_RDWR | O_NDELAY)) < 0) {
      syslog(LOG_ERR, "%s: %s", argv[1], sys_errlist[errno]);
      closelog();
      exit(1);
}

/* La ligne est ouverte, donc DTR est haut.
   On le force tout de meme pour plus de surete */
ioctl(fd, TIOCMBIS, &dtr_bit);

/* On passe en daemon. */
switch(fork()) {
      case 0: /* Fils */
              closelog();
              setsid();
              break;
      case -1: /* Erreur */
              syslog(LOG_ERR, "Impossible de forker.");
              closelog();
              exit(1);
      default: /* Pere */
              closelog();
              exit(0);
}

/* On relance syslog. */
openlog("powerd", LOG_CONS, LOG_DAEMON);

/* Maintenant, on echantillonne la ligne DCD */
while(1) {
      /* On lit le statut. */
      ioctl(fd, TIOCMGET, &flags);

      /* On controle la connexion.
         DSR doit etre haut */
      tries = 0;
      while((flags & TIOCM_DSR) == 0) {
              /* On continue a essayer, et on previent
                 toutes les deux minutes */
              if ((tries % DSR_TRIES) == 0)
                  syslog(LOG_ALERT, "Erreur de connexion onduleur");
              sleep(DSR_SLEEP);
              tries++;
              ioctl(fd, TIOCMGET, &flags);
      }
      if (tries > 0)
              syslog(LOG_ALERT, "Connexion onduleur OK");

      /* On calcule l'etat en cours. */
```

```
DCD = flags & TIOCM_CAR;
CTS = flags & TIOCM_CTS;

if (status == -1)
        {
        status = (DCD != 0) ? 0 : 1;
        if (DCD == 0)
                {
                syslog(LOG_ALERT, "Coupure de courant. Onduleur actif.");
                powerfail(1);
                }
        }

switch (status)
        {
        case 0:
                if ((DCD != 0) && (CTS != 0))
                        {
                        DCD_count = 0;
                        CTS_count = 0;
                        sleep(T0_SLEEP);
                        continue;
                        }
                if (DCD == 0)
                        DCD_count++;
                if (CTS == 0)
                        CTS_count++;
                if ((DCD_count < T0_DCD) && (CTS_count < T0_CTS))
                        {
                        sleep(1);
                        continue;
                        }
                if (CTS_count == T0_CTS)
                        {
                        status = 2;
                        syslog(LOG_ALERT, "Batteries faibles !");
                        break;
                        }
                status = 1;
                DCD_count = 0;
                syslog(LOG_ALERT, "Coupure de courant. Onduleur actif.");
                break;

        case 1:
                if ((DCD == 0) && (CTS != 0))
                        {
                        DCD_count = 0;
                        CTS_count = 0;
                        sleep(T1_SLEEP);
                        continue;
                        }
                if (DCD != 0)
                        DCD_count++;
                if (CTS == 0)
                        CTS_count++;
```

```
                                if ((DCD_count < T1_DCD) && (CTS_count < T1_CTS))
                                        {
                                        sleep(1);
                                        continue;
                                        }
                                if (CTS_count == T1_CTS)
                                        {
                                        status = 2;
                                        syslog(LOG_ALERT, "Batteries faibles !");
                                        break;
                                        }
                                status = 0;
                                DCD_count = 0;
                                CTS_count = 0;
                                syslog(LOG_ALERT, "Courant present.");
                                break;

                        case 2:
                                sleep(1);
                                continue;

                        default:
                                break;
                        }

                powerfail(status);
        }
        /* N'arrive jamais */
        return(0);
}
```

8.4 Trust UPS 400-A

J'ai reçu un message à propos du Trust UPS 400-A. Je ne sais pas si c'est le même que le Trust Energy Protector 400, donc voici le message[3] :

Marcel Amerlaan

16 juillet 1997

disponibilité

Cet onduleur ne semble plus être fabriqué par son *constructeur* , mais cela ne veut pas dire qu'il ne soit plus disponible : j'ai acheté le mien très peu cher il y a seulement un mois. De plus, cette entreprise réétiquette souvent ses produits.

câble

Il est facile à fabriquer à l'aide du câble d'origine pour powerd et de la documentation de Trust.

Il présente deux améliorations :

- indication de batterie faible ;
- extinction de l'onduleur.

[3](NdT : le texte qui suit a été reformaté. Le document d'origine comporte une copie de courrier électronique)

```
Type               : "pleur"
Cable power        : {TIOCM_DTR, 0}
Inverter Kill      : {TIOCM_RTS, 1}
Inverter Kill Time : 5
Power Check        : {TIOCM_CTS, 0}
Battery Check      : {TIOCM_CAR, 0}
Cable Check        : {TIOCM_RI,  0}
```

La fonction “cable check” n'est pas utilisée car l'onduleur ne semble pas la reconnaître.

conclusion

Voilà tout ce que je crois savoir. Si vous voulez plus d'informations sur l'onduleur, le câble ou le logiciel, contactez-moi.

Et souvenez-vous que tout ce qui est décrit ici fonctionne pour moi mais je ne garantis pas que ce soit le cas pour vous.

```
Marcel Ammerlaan
CEO Pleursoft (cela explique le nom du cable, n'est-ce pas :-)
Pays Bas
```

8.5 Sustainer S-40a

Informations sur le Sustainer S-40a[4] :

Evgeny Stambulchik

10 septembre 1995

environnement

Sustainer S-40a avec le paquetage unipower (récemment renommé genpower), câble maison (cf. infra). J'ai envoyé une copie de tout ça à Tom Webster, l'auteur du paquetage, et cela devrait apparaître dans la nouvelle version.

câble

- Note : les broches d'arrêt indiquées dans la documentation technique (4 et 6) sont incorrectes. Les bonnes sont 6 et 7, comme sur le schéma joitnt ;
- Note 2 : les broches côté PC entre parenthèse sont pour un connecteur 25 broches, les autres pour un 9 broches.

```
      COTE ONDULEUR                      COTE LINUX

        2 COUPURE DE COURANT                     1 (8)
   +----o----------------------------+------------------o  DCD
   |                                 |
   o                                 |
    /                                |
   /                                 |
   |    4 MASSE COMMUNE              |           5 (7)
   +----o-------------+--------------|------------------o  GND
   |                  |              |
   \                  |              |
    \                 |              |
```

[4](NdT : le texte qui suit a été reformaté. Le document d'origine comporte une copie de courrier électronique)

```
          o                  |              |
          |    5 BATTERIE    | FAIBLE       |               8 (5)
          +----o-------------|--------------|--------+---------o  CTS
                             |              |        |
                                          +-+-+    +-+-+
                             |            |   |    |   |
                             | Resistances|   |    |   |
                             |            |   |    |   |
                             | 3 x 10 kohm|   |    |   |
                             |            +-+-+    +-+-+
                             |              |        |      4 (20)
                             |              +--------+---------o  DTR
                             |
 |             6 ARRET DE    | L'ONDULEUR    +-------+      7 (4)
 +-+       +---o-------------|---------------+       +---------o  RTS
   \       |                 |               +-------+
    \|    -+-                |
     | <- \ /                |
    /|    -+-                |
  /        |   7             |
  |        +---o-------------+
--+--
 ---
  -
```

fichier unipowerd.h

```
/************************************************************************/
/* Fichier               : unipowerd.h                                  */
/* Programme             : unipowerd                  Version: 1.0.0    */
/* Auteur                : Tom Webster <webster@kaiwan.com>             */
/* Creation              : 1994/04/20                                   */
/* Modification          : Tom Webster                Date: 1995/04/09  */
/* Modification          : Evgeny Stambulchik (pour onduleur Sustainer) */
/*                                                                      */
/* Compilation           : GCC 2.5.8                                    */
/* Compilateur           : Linux 1.0.9                                  */
/* ANSI C Compatible     : Non                                          */
/* POSIX Compatible      : Oui ?                                        */
/*                                                                      */
/* But                   : Fichier d'entete pour unipowerd.             */
/*                       : Contient les informations de configuration   */
/*                       : de unipowerd. Editez ce fichier comme indique */
/*                       : pour activer les fonctionnalites et ajuster  */
/*                       : unipowerd pour votre onduleur.               */
/*                                                                      */
/* Copyright             : GNU Copyleft                                 */
/************************************************************************/

/* Lignes de controle RS232                      */
/*                                               */
/*                                           D D */
/*                                           T C */
/* Macro           Anglais                   E E */
/* --------------------------------------------- */
/* TIOCM_DTR       DTR - Data Terminal Ready  --> */
```

```
/* TIOCM_RTS       RTS - Ready to send        --> */
/* TIOCM_CTS       CTS - Clear To Send        <-- */
/* TIOCM_CAR       DCD - Data Carrier Detect  <-- */
/* TIOCM_RNG       RI  - Ring Indicator       <-- */
/* TIOCM_DSR       DSR - Data Signal Ready    <-- */

#define HIGH            (1)
#define LOW             0
#define PWRSTAT         "/etc/powerstatus"
#define UPSSTAT         "/etc/upsstatus"

/* CABLEPOWER est la ligne qui alimente le cable  */
/* pour la surveillance normale.                  */
#define CABLEPOWER      TIOCM_DTR

#define POWERBIT        TIOCM_CAR
#define POWEROK         HIGH

/* CABLECHECK vaut 1 pour surveiller la batterie ??*/
/* CABELCHECK vaut 0 pour ne rien surveiller       */
#define CABLECHECK      0
#define CABLEBIT        TIOCM_RNG
#define CABLEOK         HIGH

/* BATTCHECK vaut 1 pour surveiller la batterie    */
/* BATTCHECK vaut 0 pour ne rien surveiller        */
#define BATTCHECK       1
#define BATTBIT         TIOCM_CTS
#define BATTOK          HIGH

/* INVERTERKILL vaut 1 pour gerer l'arret de l'onduleur   */
/* INVERTERKILL vaut 0 pour ne rien gerer.                */
/* INVERTERBIT est la ligne qui eteint l'onduleur en      */
/*    mode powerfail.                                     */
/* INVERTERTIME est la duree en secondes de maintien haut */
/* de la ligne INVERTERBIT en haut pour eteindre.         */
#define INVERTERKILL    1
#define INVERTERBIT     TIOCM_RTS
#define INVERTERTIME    5

/************************************************************************/
/* Fin du fichier unipowerd.c                                           */
/************************************************************************/
```

fichier genpowerd.h

Pour la nouvelle version du logiciel (*genpowerd*), je pense qu'il faut ajouter la ligne suivante :

```
/* Evgeny's Sustainer S-40A */
{"sustainer", {TIOCM_DTR,0}, {TIOCM_RTS,1}, 5, {TIOCM_CAR,0}, {TIOCM_CTS,0},
{0,0}}
```

8.6 Systel

Une autre entreprise israélienne. Je ne leur ai jamais acheté d'onduleur, mais il m'ont for aimablement fourni une documentation détaillée sur leur port de communication. Il devrait être assez facile de contrôler leur onduleur. Leur numéro de téléphone est :

972-8-409-019 (fax 972-8-407-216).

8.7 Deltec Power, Fiskars Power Systems et Exide

Fiskars

est une holding finnoise, anciennement propriétaire de *Deltec Power* . En mars 1996, Fiskars a vendu Deltec Power à *Exide* . A cette date, Deltec Power était l'un des plus gros constructeurs d'onduleurs.

Avec Fiskars, Deltec fabriquait les PowerServers 10, 20, 30 et 40. La page web de Deltec Power en mentionne d'autres.

Exide joint maintenant un logiciel de contrôle avec ses onduleurs qui fonctionne sous Linux. Ils vendent aussi celui-ci séparément et affirment qu'il fonctionne avec d'autres onduleurs aussi.

J'aimerais avoir des nouvelles de gens qui utilisent ce logiciel.

Voici l'annonce qu'ils m'ont envoyée par e-mail :

Exide Electronics annonce Lansafe III, logiciel de gestion d'onduleurs sous Linux.

Lansafe III est une application de gestion d'onduleurs. Elle permet l'arrêt automatique du système en cas de coupure de courant de longue durée qui dépasserait l'autonomie de la batterie de l'onduleur.

Lansafe III permet les messages "broadcast" et l'envoi de courriers électroniques en fonction des réglages utilisateur. La séquence d'arrêt peut aussi être paramétrée.

Lansafe III fonctionne avec la plus grande partie des onduleurs Exide Electronics. Il permet aussi l'arrêt automatique simple avec des onduleurs d'autres constructeurs.

Lansafe III pour Linux fonctionne sur les systèmes Linux à base Intel. Deux interfaces sont fournies : mode caractères et X11/Motif.

Lansafe III fonctionne sur toutes les plateformes majeures de systèmes d'exploitation : Linux, IBM AIX, HP UX, Digital Unix, SCO Unix, Solaris, SunOS, AT&T Unix, toutes les plateformes Windows, OS/2, Novell et Macintosh en particulier.

Lansafe III est fourni avec les onduleurs Exide suivant :

- OneUPS Plus ;
- NetUPS ;
- PowerWare Prestige ;
- PowerWare Profile ;
- PowerWare Plus 5xx.

Il est aussi fourni avec les onduleurs FPS Power Systems :

- PowerRite Plus ;
- PowerRite Max ;

- PowerWorks A30 ;
- PowerWorks A40 ;
- séries 9 000 ;
- séries 10 000.

Il est aussi possible d'acquérir une licence logicielle séparée pour l'utilisation d'un onduleur plus ancien ou d'un autre constructeur. Les licences simples sont à USD 149, des licences site sont disponibles.

Pour tout détail, visitez nos sites web : *www.exide.com* ,

www.fiskarsUPS.com

et *www.deltecpower.com*

Accessoirement, lorsque j'ai tenté de me connecter à www.fiskarsUPS.com, il m'a été demandé une identification et un mot de passe.

8.8 Onduleur Beaver modèle UB500

Dan Fandrich

écrit :

Je pense avoir réussi à faire fonctionner mon vieil onduleur Beaver modèle UB500 avec genpower. L'interface utilise des niveaux de tension compatibles RS-232, donc l'installation est simple. Ily a un connecteur DB-9 femelle à l'arrière qui se connecte directement dans un port série DB-9 de PC à l'aide d'un câble droit.

Les interrupteurs DIP permettent quelques ajustements. Pour émuler le type d'onduleurs apc1-nt de genpower, ils doivent être positionnés comme suit :

- 1 on (CTS = coupure de courant) ;
- 2 off (CTS = batterie faible) ;
- 3 off (DSR = coupure de courant) ;
- 4 off (DSR = batterie faible) ;
- 5 off (CD = coupure de courant) ;
- 6 on (CD = batterie faible) ;
- 7 off (RI = coupure de courant) ;
- 8 off (RI = batterie faible) ;
- 9 on (DTR = extinction) ;
- 10 off (RTS = extinction).

Les interrupteurs forment des groupes de paires adjacentes pour chaque broche de sortie. Ils sont exclusifs mutuellement - ne tentez pas de positionner ON les 5 et 6 ensemble, par exemple, ou vous ferez un court-circuit entre les signaux coupure de courant et batterie faible.

C'est tout ce qu'il y a à dire. Vous pouvez ajouter cela à votre documentation.

8.9 Sendom

Charli

écrit :

J'ai connecté un onduleur Seldom avec powerd. Peut-être que ce qui suit sera utile avec d'autres onduleurs.

J'ai utilisé le diagramme de la page de man de *powerd* :

```
        9 broches  25 broches

DTR        4          20        ----------
                                 |       >
DSR        6           6        --       < 10k
                                         >
DCD        1           8        -------------------
                                                relais
GND        5           7        -------------------
```

En fait, l'onduleur seldom n'utilise pas de relais mais quelque chose d'autre et fonctionne dans un sens, *mais pas dans l'autre*. Si donc le câble ne fonctionne pas, il faut essayer d'inverser les broches sur le “relais”.

8.10 Best

L'information sur les onduleurs Best est disponible sur le site web de *Best Power* . Leur site contient un paquetage `checkup.tar` (section 4 (Logiciels)) de communication avec leurs onduleurs, aussi bien en modes intelligent que bâte, fourni en sources, donc compilable sous Linux.

Best Fortress avec le logiciel de Best

Mini-Howto des onduleurs Best Power

par *Michael Stutz*

et *http://dsl.org/m* .

Version 1.0, 14 août 1997

8.10.1 Avertissement

Copyright 1997 Michael Stutz[5] ; cette information est libre ; elle peut être redistribuée et/ou modifiée selon les termes de la Licence Publique Générale GNU (GPL) version 2 ou (à votre préférence) ultérieure, pour autant que la présente phrase soit conservée ; cette information est fournie SANS AUCUNE GARANTIE ; sans même de garantie implicite d'adaptation à un besoin particulier ; se reporter à la GPL de GNU pour plus de détails.

8.10.2 Introduction

Best Power

est constructeur d'onduleurs de haute qualité, et leur série Fortress est particulièrement bien adaptée à des utilisateurs habituels de Linux. Bien que ses produits ne soient actuellement pas aussi bon marché que

[5]NdT : la traduction de ce paragraphe est fournie à titre indicatif au lecteur. Se reporter à la version originale pour les termes exacts.

certains autres (comme ceux d'APC), Best Power fournit le code source de son logiciel et a été très réactif quant aux questions posées par des utilisateurs de Linux. De plus, son matériel semble choisi souvent par les consommateurs, ce qui en fait un bon choix pour les utilisateurs de Linux.

Ce document décrit l'installation d'un onduleur Best Power Fortress (le modèle utilisé est un 660a de 1996 accompagné de son CD-ROM) sur une machine Linux.

8.10.3 Installation

Matériel Installez l'onduleur comme indiqué par les instructions. Les séries *Fortress* de Best Power sont fournies avec un câble RS-232 destiné à être connecté à un port série libre à l'arière de l'ordinateur.

Logiciel Voici ce qui diffère du manuel, qui ne contient pas actuellement d'instructions spécifiques pour Linux. En revanche, le CD-ROM d'accompagnement conient avec le code source du logiciel de l'onduleur, ce qui en rend la mise en oeuvre triviale.

Pour réaliser celle-ci, suivez les étapes ci-dessous, et utilisez le manuel comme référence pour avoir une vue d'ensemble sur le fonctionnement général du logiciel. J'ai pris la liberté de faire quelques modifications dans ce HOWTO sur la configuration du logiciel *Fortress* pour Unix d'une manière qui me semble plus adaptée à un système Linux. Par exemple, j'ai éliminé la nécessité d'un répertoire `/etc/best`, et placé les exécutables dans `/usr/local/bin` qui me semble un endroit plus approprié.

- D'abord, créez le script “upsdown” destiné à être exécuté lors d'un arrêt secteur. Celui-ci va arrêter le système :

```
cat > /etc/upsdown <<EOF
#!/bin/sh
shutdown -h now < /dev/console &
EOF
```

- Maintenant, créez les répertoires pour la documentation et le code source :

```
mkdir /usr/doc/best
mkdir /usr/local/src/best
```

- Montez le CD-ROM, et désarchivez le fichier `unix/checkups.tar` dans un répertoire temporaire :

```
cd /tmp
tar /cdrom/unix/checkups.tar
```

- Allez dans le répertoire `etc/best/advanced` qui doit avoir été créé dans le répertoire temporaire à partir de l'archive ;
- Copiez la documentation et les fichiers script à leurs emplacements idoines dans le système :

```
cp README /usr/doc/best
cp manual.txt /usr/doc/best
cp bestsend /etc
cp source/*.c /usr/local/src/best
```

- Nettoyez le chantier dans le répertoire temporaire et compilez le logiciel :

```
cd /usr/local/src/best
rm -R /tmp/etc
gcc -o checkups checkups.c
```

```
gcc -o mftalk mftalk.c
mv checkups /usr/local/sbin
mv mftalk /usr/local/sbin
```

- Testez l'onduleur. Remplacez `ttySx` par le port série de votre choix. Si votre connexion est bonne, vous devriez voir une ligne de caracères s'imprimer à l'écran :

```
mftalk /dev/ttySx
```

- Rendez le programme checkups exécutable au démarrage pour le tester. Cela peut être réalisé d'un certain nombre de manières différents (décrites dans le manuel). Celle que j'ai utilisée consiste à ajouter une ligne dans `/etc/inittab` :

```
ups:234:once:/usr/local/sbin/checkups -c500 /dev/ttyS1
```

- Testez le tout. Supprimez l'alimentation secteur de l'onduleur en retirant le fusible de l'onduleur, et attendez quelques minutes. Le logiciel affiche un message d'alerte puis arrête le système après quelques instants.
- Si cela fonctionne, retirez l'option `-c500` de la ligne de votre `inittab` (qui en gros implique d'arrêter le système systématiquement au lieu de ne le faire que lorsque le courant est coupé), et vous pouvez rouler !

Conclusion Toute suggestion permettant d'améliorer ce document ou les techniques qui y sont décrites est la bienvenue. A l'instant où j'écris ces lignes, *Best Power* semblait intéressé par l'inclusion de la présente information ou d'une autre dans la sienne afin d'aider les utilisateurs de Linux par rapport à ses produits, il s'agit donc réellement d'une entreprise à promouvoir. Vous pouvez lui transmettre vos impressions à *sales@bestpower.com*

et *support@bestpower.com* .

Best Fortress LI-950

Quelques commentaires sur le *Best Fortress*, de *Leonard N. Zubkoff* , message du 25 mai 1995 dans comp.os.linux.hardware[6] :

Citation de nautix@community.net :

D'accord avec ce que dit Craig. APC a été très peu coopératif, mais je n'ai que de bonnes choses à dire sur Best. J'utilise son modèle LI 660 ; 660 VA, des tas d'indications sur le panneau frontal, etc. Le logiciel CheckUPS est en option payante et nécessite quelques bidouillages pour entrer dans mons système de fichiers FSSTND-isé (NdT : File System STaNDard, le standard de répartition des éléments dans les répertoires préconisé pour Linux) (les répertoires et noms de fichiers sont en dur pour SunOS 4.1.x). Je serai heureux de vous envoyer mes diffs, si vous les voulez (j'adore quand un constructeur fournit le source en tant que pratique commerciale normale !!).

Le logiciel CheckUPS est limité, cependant, à réaliser des arrêts automagiques (NdT : `automagic` dans le texte). L'onduleur peut fournir des tas d'informations sur son état ; CheckUPS ne contrôle que "Si le courant est coupé, combien de temps reste-t'il d'autonomie à la batterie ?".

Best suit aussi ses questionnaires de satisfaction clients. J'ai indiqué ma déception que CheckUPS ne dispose pas de plus de fonctions d'interrogation (comme le voltage en entrée, en sortie, le pourcentage de charge, etc.) qui sont disponibles en entrée. J'ai demandé les spécifications de l'interface ; ils ont dit : "bien sûr" et me l'ont envoyé en 2 jours, gracieusement. Un contrôleur d'état de l'onduleur complet est dans ma casserole de derrière. Quelqu'un voit-il une utilité à ce genre d'utilitaire ?

[6](NdT : le texte qui suit a été reformaté. Le document d'origine comporte une copie de message de forum)

Réponse de Leonard N. Zubkoff :

Laissez-moi ajouter une autre recommandation pour Best Power. Je viens d'acheter un Fortress LI-950, mais j'ai décliné leur offre logicielle pour CheckUPS. Contrairement à certaines autres gammes, un simple câble trois fils suffit à connecter le Fortress à un port série – pas besoin de montage “pull-up” à faire dans le câble. Quelques minutes de bidouillage et j'avais un programme qui fait à la fois daemon d'arrêt système et qui coupe le courant de sortie ensuite lorsque le système est arrêté durant une période sur batterie.

Je pourrais éventuellement utiliser le mode de communications série plus intelligent plutôt que le simple mode de contact, et j'ai donc demandé la documentation au support technique de Best, et il est arrivé aujourd'hui, une semaine après mon appel. Après avoir étudié celle-ci, je déciderai si une interface plus intelligente est réellement intéressante, en particulier puisque dans certains cas j'aurais besoin d'arrêter deux machines en réseau partageant l'onduleur.

Leonard.

Best Ferrups

En complément à la documentation et au logiciel sur le site web de *Best*, vous pouvez aussi utiliser le paquetage `bestups-0.9.tar.gz` (section 4 (Logiciel)). Nous avons juste commencé à le tester avec notre *Ferrups* 5 kVA.

L'idée de base est qu'il y a deux modules. L'un qui reçoit des demandes d'information du port réseau, les relaie à l'onduleur, et renvoit les résultats. Le second module parle au premier, interprète les résultats, et répond OK ou FAIL.

C'est suffisant pour que le paquetage `powerd-2.0.tar.gz` (section 4 (Logiciel)) fasse le reste.

Les détails se trouvent dans le paquetage lui-même.

Par ailleurs, notre *Ferrups* 5 kVA a fonctionné sans histoire pour nos 10 ordinateurs et 30 écrans.

8.11 GPS1000 d'ACCODATA

```
>From hennus@sky.nl.mugnet.org Thu Mar 10 15:10:22 1994
Newsgroups: comp.os.linux.help
Subject: Re: shutdown automatique avec un onduleur
From: hennus@sky.nl.mugnet.org (Hennus Bergman)
Date: Tue, 1 Mar 1994 22:17:45 GMT
Distribution: world
Organization: The Organization For Removal Of On-Screen Logos

Dans l'article <CRAFFERT.94Feb28125452@nostril.lehman.com>,
Colin Owen Rafferty <craffert@nostril.lehman.com> ecrit :
>Je suis prêt à acheter un onduleur pour ma machine, et j'en
>voudrais un qui sache faire un "auto-shutdown".
>
Je viens d'en acheter un vraiment pas cher :-)
C'est un GPS1000 d'ACCODATA. Tout le monde connaît la bonne qualite
de leur production (je n'ai pas d'actions chez eux :-() ?

>Je suppose que tous ont une sorte de connexion serie qui previent le
>systeme de cela.
>
Je l'ai pris à part pour trouver comment il fonctionnait. Il y avait
trois optocoupleurs (deux sorties, une entree) connectes sur un connecteur
à 9 broches à l'arriere. L'un s'allume lorsque le courant est coupe, et
```

```
s'eteint lorsque ce dernier revient. Durant ce temps, on peut utiliser
l'"entree" pour arreter la batterie (il relache le relais de puissance).
Le troisieme est une sorte d'acquittement de la commande d'arret. Je
pense que l'interface de mon onduleur a ete concue pour etre connectee
a des niveaux TTL, mais avec quelques resistances il peut etre connecte a
un port serie. Il est branche de telle sorte qu'avec un port RS-232 on
ne puisse utiliser les deux optocoupleurs de sortie; mais l'acquittement
de la commande d'arret n'est pas vraiment necessaire. On peut se conten-
ter de celui qui est important (Notez qu'il est possible de faire fumer
la partie transistor des optocoupleurs avec des niveaux RS-232 si on
le branche mal). ;-)

J'esperais etre capable de le connecter a mon port de jeux inutilise,
mais ce dernier n'a pas de sortie, n'est-ce pas ?
Je vais probablement finir par mettre un port parallele supplementaire
pour ca.

Tous les onduleurs n'utilisent pas d'optocoupleurs, certains se contentent
de simple relais, qui sont moins difficiles a connecter, mais bien sur,
pas aussi `elegants'.

>Quelqu'un a-t-il ecrit un paquetage qui surveille l'onduleur et effectue
>un shutdown (ou similaire) lorsque le courant s'arrete ?
SysVinit-2.4 (et probablement 2.5 aussi bien) a un demon `powerd' qui
surveille le port serie en continu et previent init quand CD (Detection
de porteuse) tombe. Init active ensuite un shutdown avec un delai. Si le
courant revient apres quelques minutes, le shutdown est arrete. Tres beau.
Le seul probleme que j'aie eu avec est qu'il ne dit pas a l'onduleur de
s'arreter lorsque le shutdown est fini. Il attend simplement la avec une
invite root. Je vais probablement ecrire un petit programme pour l'arreter
>depuis /etc/brc. RSN.

>    Colin Rafferty, Lehman Brothers <craffert@lehman.com>

Hennus Bergman
```

8.12 TrippLite BC750LAN (Standby UPS)

Tom Webster , l'auteur du paquetage *genpower*, m'a envoyé des informations sur le *TrippLite BC750LAN*. Si vous avez l'un d'entre eux, c'est probablement le meilleur paquetage pour commencer.

Mais pour être exhaustif, voici le diagramme de brochage du câble (réalisé par tâtonnements, et sans documentation) :

```
Onduleur              Systeme
  DB-25               DB-25
    1 <--------------> 1       Masse

    2 <--------------> 4       Coupure de secteur
    8 <--------------> 8       Circuit de detection

    3 <--------------> 2       Inverseur d'arret
   20 <--------------> 22      Circuit
```

8.13 APC

Si la pléthore de paquetages pour APC cités plus haut ne vous permettent pas de démarrer, il est possible que la section qui suit soit d'une certaine utilité.

Backup-UPS

Il semble qu'il y ait une certaine controverse sur la fiabilité des informations indiquées ici sur les APC Back-UPS, donc, soyez prudent. Je préface cette section avec un message d'avertissement que j'ai recu. Il peut ne pas prendre tout son sens tant que le reste de la section n'est pas lu, mais ainsi, au moins vous aves plus de chances de le voir. Et, à nouveau, comme je n'ai aucun onduleur APC, je ne peux vérifier la fiabilité d'aucun de ces messages.

Un message d'avertissement

Message de *Marek Michalkiewicz*

sur le BUPS-HOWTO[7] :

Si vous voulez connecter un onduleur APC Back-UPS sur votre machine Linux, ce qui suit peut vous intéresser.

Il y a un bon BUPS-HOWTO qui décrit comment le faire. Mais il comporte un “bug”.

Le signal RTS du port série est utilisé pour arrêter l'onduleur. Celui-ci ne s'arrêtera que s'il travaille sur batterie. Le manuel indique que le le signal d'arrêt doit durer au moins 0,5ms. Mais un temps inférieur est suffisant, au moins pour mon propre APC Back-UPS 600.

L'utilisation de RTS peut être dangereuse, car ce dernier est monté à l'ouverture du périphérique. Le programme backupsd le redescend ensuite, mais il reste haut un moment. Cela coupe le courant lors du premier lancement de backupsd s'il y a une coupure secteur à ce moment précis. Cela peut arriver par exemple si l'onduleur est éteint, et que le courant revienne seulement pour un moment.

Soit il faut lancer backupsd avant de monter les systèmes de fichiers en lecture/écriture, soit (de préférence) utiliser TX (broche 3) plutôt que RTS (broche 7) pour éteindre l'onduleur (la numérotation est pour un DB-9). On peut utiliser `ioctl(fd, TCSBRKP, 10);` pour monter TX pendant une seconde, par exemple. L'utilisation de TX doit etre plus sûre. Je posterai peut-être les diff si le temps me le permet...

BUPS-HOWTO

Luminated Software Group Présente

HOWTO utilisation d'onduleurs *Back-UPS* (d'*APC*) (pour protéger votre système Linux)

Version: 1.01 BETA

Document de : *Christian G. Holtje*

Information sur le câblage et aide : *Ben Galliart*

Adaptation française : *Bernard Choppy*

Ce document est placé dans le Domaine Public à une condition. Celle-ci est que ce qui appartient a César revienne à César. Modifiez ceci autant que vous voulez, rappelez juste que nous avons travaillé dessus.

Attention !

Ni moi, ni aucun de ceux qui on écrit ou aidé à ce document, ne garantissons quoi que ce soit concernant ces textes/sources/indications. Si quoi que ce soit est endommagé, nous n'y sommes POUR RIEN !

[7](NdT : le texte qui suit a été reformaté. Le document d'origine comporte une copie de courrier électronique)

Cela fonctionne POUR AUTANT QUE NOUS LE SACHIONS, mais nous pouvons avoir fait des erreurs. Donc, soyez prudent !

[8]

Bien, vous venez juste d'acheter (ou vous allez le faire) un *Back-UPS* d'*APC* (d'autres modèles pourront peut-être bénéficier de ces informations, avec peu ou pas de modifications, mais nous ne savons pas). Vous avez jeté un coup d'oeil au prix du couple logiciel/câble *Power-Chute*, et n'êtes pas sûr que le jeu en vaille la chandelle. Bien, j'ai fait mon propre câble, et mon propre logiciel et je les utilise pour arrêter automatiquement mon système Linux lors d'une coupure secteur. Vous savez quoi ? Vous pouvez aussi !

*** Le Câble ***

C'était la partie la plus difficile à imaginer (je m'y connais peu en hardware, donc Ben a fait le plus gros du travail). Pour en fabriquer un, vous devez acheter ce qui suit chez votre marchand d'électronique du coin :

- 1 connecteur à souder subminiature mâle DB-9 ;
- 1 connecteur à souder subminiature femelle DB-9 ;
- 2 boîtiers pour les connecteurs ci-dessus (vendus séparement en général) ;
- Du câble multi-brins (pas du mono-brin).

Il vous faut aussi, mais vous pourrez peut-etre l'emprunter :

- un fer à souder ;
- de la soudure.

Ok... Voici comment connecter le tout !

Ces diagrammes montrent le côté ARRIERE (celui où vous soudez les câbles sur les broches). Les lettres V, R et B représentent les couleurs des câbles que j'ai utilisés, et facilitent la distinction des lignes[9] . Le manuel de l'*APC* utilise une numérotation différente. Ignorez-la ! Utilisez la nôtre... Je l'ai déjà changée pour vous !).

```
        ---------------------     Cote Male (vers l'onduleur)
         \  B   R  *  *  * /
           \  *  *  *  V  /
             ------------


        ---------------------     Cote femelle (vers le port COM)
         \  R   *  *  *  V /
           \  *  B  *  *  /
             ------------
```

Pour ceux qui préfèrent les chiffres :

```
          Male         Femelle
   --------------------------------------
            1             7         Bleu
            2             1         Rouge
            9             5         Vert
```

[8]NdT : Le document d'origine contient des références de pièces détachees Radio-Shack, qui etaient distribuées par le réseau Tandy en France. Ce reseau n'existe plus, et les références ont donc été supprimées de la version française. Le lecteur néammoins intéressé pourra se reporter à la version anglaise du présent document.

[9](Note : j'utilise la numérotation standard RS-232 (pour autant qu'on puisse dire)

—- Complément : Utilisation des broches RS-232 ! —- Puisque nous avons eu à trouver cette information :

Depuis l'ARRIERE (côté soudure), les broches sont numérotées ainsi :

```
---------------------
 \  1   2  3  4  5 /
   \  6  7  8  9  /
      ------------
```

Les broches signifient

```
Numero  Nom                     Abreviation (parfois prefixee par D)
1       Detection de porteuse           CD
2       Reception de donnees            RD
3       Transmission de donnees         TD(?)
4       Terminal de donnees pret        DTR
5       Masse de signal                 Gnd
6       Jeu de donnees pret             DSR
7       Demande pour envoyer            RTS(?)
8       Pret a envoyer                  CS
9       Indicateur de sonnerie          RI
```

Ce que nous avons fait était la connexion de la ligne RS-232 de l'onduleur "Fail Output" sur CD, le châssis de l'onduleur sur Gnd, et l'entrée "Shut Down" sur RTS. Facile, maintenant qu'on vous le dit, non ?

Je n'ai aucune idée du comportement du logiciel ci-dessous, si vous achetez le câble d'APC. Il peut fonctionner, ou non.

*** Le Logiciel ***

J'utilise le paquetage *SysVInit* de Miquel van Smoorenburg pour Linux (voir à la fin pour emplacements, remerciements, adresses E-mail, etc.). Je ne sais ce qui doit être changé pour utiliser l'init de quelqu'un d'autre, mais je sais que ce code (qui suit) fonctionne avec celui de Miquel. Simplement ainsi je remercie comme je le dois. J'ai regardé dans le code de Miquel pour comprendre comment `ioctl()` fonctionnait. Si je n'avais pas eu cet exemple, j'aurais eu des problèmes. J'ai aussi utilisé la routine `powerfail()` (telle quelle, je crois), puisqu'elle doit interagir avec init, j'ai pensé qu'il devait savoir ça mieux que moi. Le fichier `.c` est à la fin de ce document, et nécessite seulement d'être copié/collé. Pour cela, supprimez simplement tout ce qui n'est pas du code. Ce document doit se terminer par la ligne /* Fin de Fichier */... Coupez le reste.

Ce programme peut, soit être lancé comme daemon pour contrôler l'état de l'onduleur et l'indiquer à init, soit être lancé pour envoyer la commande `kill-power` (coupure d'alimentation) à l'onduleur. L'alimentation ne sera coupée que s'il y a un problème secteur et que l'onduleur est sur batteries. Une fois le courant revenu, il se rallume.

Pour le lancer comme démon, entrez simplement :

```
backupsd /dev/backups
```

`/dev/backups` est un lien vers `/dev/cua0` (COM 1, pour les DOSseurs) actuellement. La beauté du lien est que je n'ai qu'à le refaire si je passe sur COM 2 ou COM 3.

Ensuite, si le secteur s'arrête, init lancera les commandes de `powerwait`. Un exemple (qui vient de mon `/etc/inittab`) :

```
#Voici les actions de coupure de courant
pf::powerwait:/etc/rc.d/rc.power start
po::powerokwait:/etc/rc.d/rc.power stop
```

Powerwait sera lancé si le courant baisse, et powerokwait s'il revient.

Voici mon `rc.power` complet :

```
#! /bin/sh
#
# rc.power      Ce fichier est execute par init en cas de coupure de courant
#
# Version :     @(#)/etc/rc.d/rc.power   1.50    1994-08-10
#
# Auteur :      Christian Holtje, <docwhat@uiuc.edu>
#

  # Definit le chemin
  PATH=/sbin:/etc:/bin:/usr/bin:/sbin/dangerous

  # Regarde comment nous avons ete appele
  case "$1" in
        start)
                echo "Attention - probleme d'alimentation secteur." | wall
                # Sauvegarde le niveau de fonctionnement actuel
                ps | gawk '{ if (($5 == "init") && ($1 == "1")) print $6 }' \
                         | cut -f2 -d[ | cut -f1 -d] \
                         > /tmp/run.level.power
                /sbin/shutdown -h +1m
                ;;
        stop)
                echo "Alimentation secteur revenue." | wall
                echo "Tentative d'arret du shutdown." | wall
                shutdown -c
                ;;
        *)
                echo "Usage:  $0 [start|stop]"
                exit 1
                ;;
  esac
```

Pas mal, non ? En fait, il y a un petit problème, là... Je n'ai pas eu le temps de le trouver... S'il y a un gourou “sh” par ici...

J'ai laissé un petit détail de côté, c'est de faire couper l'alimentation par l'onduleur si le PC est arrêté courant coupé. Cela est réalisé en ajoutant la ligne suivante à la fin de votre script halt :

```
/sbin/backupsd /dev/backups killpower
```

Cela va simplement couper l'alimentation si le secteur est coupé.

*** Tester le tout ***

C'est juste une petite section pour vous dire :

SOYEZ PRUDENT !

Je vous recommande la sauvegarde de vos partitions Linux, avec plusieurs `sync` avant de tester, et d'être prudent en général. Evidemment, je ne fais que vous le recommander. Je n'ai pas été prudent du tout, et j'ai eu à nettoyer ma partition plusieurs fois pendant les tests de ma configuration. Mais celle-ci fonctionne. :-)

*** Où l'obtenir ***

Le SysVInit de Miquel van Smoorenburg's peut se trouver sur : *SysVinit-2.50.tgz* et une correction pour certains shell bash se trouve juste à côté : *SysVinit-2.50.patch1*

Pour ce qui est d'obtenir ce HOWTO, vous pouvez m'envoyer un E-mail, docwhat@uiuc.edu avec pour sujet :'request' et le mot-clef 'backups' dans le corps du message : *Demande du HOWTO original*

(il est possible que j'automatise cela, et d'autres choses).

*** Section des remerciements qui sont dûs ***

Merci à :

- *Miquel van Smoorenburg*

  pour son superbe paquetage *SysVInit* et son `powerd.c` qui m'ont beaucoup aidés ;
- *Christian Holtje*

  Documentation `backupsd.c` (ce qui n'est pas de Miquel) `rc.power` ;
- *Ben Galliart*

  Le câble, informations sur le standard RS-232 et astuces bruyantes (non rapportées ici).

```
/*  backupsd.c -- Simple daemon pour lire les signaux de coupure de
 *                courant d'un onduleur Back-UPS (d'APC).
 *
 *  Certaines parties proviennent du powerd.c de Miquel van Smoorenburg
 *  D'autres sont originales de Christian Holtje <docwhat@uiuc.edu>
 *  Je crois qu'on peut dire que c'est dans le Domaine Public, simplement
 *  n'oubliez pas de citer les auteurs originaux, la ou c'est necessaire.
 *
 *  Avertissement : Nous ne garantissons RIEN de ce logiciel, ni
 *                  n'assumons aucune responsabilité concernant son
 *                  utilisation, bonne ou mauvaise.
 */

#include <sys/types.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <errno.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <signal.h>

/* C'est le fichier necessaire pour SysVInit */
#define PWRSTAT         "/etc/powerstatus"

void powerfail(int fail);

/* Programme principal */
int main(int argc, char **argv)
{
  int fd;
  int killpwr_bit = TIOCM_RTS;
  int flags;
  int status, oldstat = -1;
  int count = 0;

  if (argc < 2) {
        fprintf(stderr, "Usage: %s <peripherique> [killpower]\n", argv[0]);
        exit(1);
```

```
    }

    /* Ouverture du port */
    if ((fd = open(argv[1], O_RDWR | O_NDELAY)) < 0) {
          fprintf(stderr, "%s : %s : %s\n", argv[0], argv[1], sys_errlist[errno]);
          exit(1);
    }

    if ( argc >= 3  && (strcmp(argv[2], "killpower")==0) )
        {
            /* Coupons l'alimentation */
            fprintf(stderr, "%s : Tentative de coupure d'alimentation !\n",
                    argv[0] );
            ioctl(fd, TIOCMBIS, &killpwr_bit);
            /* Hmmm... Si vous avez une coupure d'alimentation, */
            /* ce code ne sera jamais execute */
            exit(0);
        }
    else
        /* Puisqu'il ne faut pas couper l'alimentation, il faut restaurer */
        /* RTS (killpwr_bit) */
        ioctl(fd, TIOCMBIC, &killpwr_bit);

  /* Passe en demon. */
    switch(fork()) {
    case 0: /* Je suis le fils */
                setsid();
                break;
    case -1: /* Passage demon manque */
                fprintf(stderr, "%s : fork impossible.\n", argv[0]);
                exit(1);
    default: /* Je suis le pere */
                exit(0);
    }


    /* Maintenant, on scrute la ligne DCD */
    while(1) {
        ioctl(fd, TIOCMGET, &flags);
        status = (flags & TIOCM_CD);
        /* Si DCD est monte, le secteur est coupe */
        if (oldstat == 0 && status != 0) {
            count++;
            if (count > 3) powerfail(0);
            else { sleep(1); continue; }
        }
        /* Si DCD est redescendu, le secteur est revenu */
        if (oldstat > 0 && status == 0) {
            count++;
            if (count > 3) powerfail(1);
            else { sleep(1); continue; }
        }
        /* Reinit du compteur, sauvegarde de l'etat et sleep 2 secondes */
        count = 0;
        oldstat = status;
```

```
        sleep(2);
    }
    /* Erreur ! (ne doit pas arriver) */
    return(1);
  }


  /* Signale a init que le courant est coupe ou revenu */
  void powerfail(ok)
  int ok;
  {
    int fd;

    /* Cree le fichier necessaire a init pour shutdown/abandon */
    unlink(PWRSTAT);
    if ((fd = open(PWRSTAT, O_CREAT|O_WRONLY, 0644)) >= 0) {
          if (ok)
                  write(fd, "OK\n", 3);
          else
                  write(fd, "FAIL\n", 5);
          close(fd);
    }
    kill(1, SIGPWR);
  }

  /* Fin de Fichier */
```

Autres informations

[10]

Message de *Jim Ockers*

du 12 janvier 1995 dans `comp.os.linux.hardware` :

Selon la base de connaissances (*KnowledgeBase*) de *Microsoft*, il semble que la broche 5 du connecteur des onduleurs *APC Back-UPS* et *Smart-UPS* (testé avec un *Back-UPS 400* sous *Windows NT*) monte un signal "batterie faible" deux minutes au moins avant l'épuisement de la batterie.

Ce signal est au niveau "TTL collecteur ouvert", et peut être ramené aux niveaux RS-232 selon le schéma suivant :

```
Broche 5                 Broche 8
           +---------+
   o------| 10 kOhm |-------o
           +---------+
```

Par ailleurs, le manuel de l'onduleur stipule que la broche commune à utiliser est la 4 (et non la 9, même si celles-ci sont branchées ensemble).

Message de *Peter Kammer*

du 7 octobre 1996 :

Les schémas de brochage sont inversés en ce qui concerne les connecteurs mâles : en effet, les broches sont numérotées de manière inverse sur les connecteurs mâles et femelles (puisque leurs sens s'opposent

[10](NdT : Le document original comporte de nombreuses copies de courriers électroniques à ce point. Le traducteur s'est permis d'en réaliser une synthèse plus courte et, il l'espère, plus facile à utiliser)

lors du branchement). Il faut donc considérer que les schémas pour les connecteurs mâles sont vus côté extérieur et non côté intérieur (soudure), contrairement à ce qui est indiqué.

Par ailleurs, il existe un document de référence technique pour les onduleurs *Back-UPS* qui se trouve sur le *site web*

d'*APC*.

Message de *Troy Muller*

du 6 avril 1997 :

L'onduleur *Back-UPS Pro 650* fonctionne avec le câble standard d'*APC*. La référence du câble est `940-023A` et le logicel est *Enhanced_APC_BackUPS*. Ce logiciel envoit des messages globaux toutes les deux secondes, mais un eu de bidouillage de `dowalll.c` permet de limiter cette fonction.

APC Smart-UPS

De nombreuses personnes ont un APC Smart UPS. Il semble qu'il existe des paquetages pour utiliser ceux-ci en mode "intelligent" (voir les paquetages mentionnés plus haut `Enhanced_APC_UPSD-v1.4.tar.gz`, `apcd-0.5.tar.gz` et `smupsd-0.7-1.i386.rpm` décrits dans la section 4 (Logiciels)). Je ne sais pas ce que vaut le support pour chacun d'eux. Il semble qu'*APC* **continue** à refuser de publier son protocole pour le mode "intelligent" sans un accord de non-diffusion, ainsi tout le monde a dû faire de la rétro-ingéniérie dessus.

Le consensus général est d'investr dans une gamme qui publie cette information, comme *Best*.

Une autre possibilité est d'utiliser la version du logiciel de contrôle d'onduleurs *Powerchute* d'*APC* pour *SCO Unix* via le paquetage de compatibilité *iBCS*. *Clive A. Stubbings* me dit que cela fonctionne bien après quelques ajustements du script d'installation. Il dit que le seul problème est que "l'interface graphique semble avoir des problèmes à contrôler des onduleurs à-travers le réseau".

Si vous possédez un *APC Smart-UPS* et que vous n'arriviez pas à le faire fonctionner en mode intelligent avec aucun de ces logiciels, vous pouvez malgré tout encore l'utiliser en mode bête. Les sections qui suivent détaillent cette procédure. J'ai reçu, en particulier, des messages concernant les modèles *600*, *700* et *1400*. Il vous faudra probablement bidouiller `powerd.c` comme indiqué dans la section 6.2 (Analyse de câbles et modification de powerd.c).

Message de *Lam Dang*

du 19 août 1994 dans comp.os.lnux.misc :

Réalisation du câble pour un *APC Smart-UPS modèle 600*.

Le cable est a realiser entre un connecteur DB-9 femelle sur l'onduleur et un DB-25 male sur l'ordinateur. Le boitier du DB-25 est assez grand pour contenir un regulateur de tension et deux resistances. L'interface entre le connecteur de l'onduleur et celui du PC est ainsi :

```
Onduleur (DB-9)         PC (DB-25)

1 (Extinction)           20 (DTR)
3 (Coupure de secteur)    5 (CTS)
4 (Commun)                7 (GND)
5 (Batterie faible)       8 (DCD)
9 (Masse chassis)         1 (Chassis Ground)
```

Vous pouvez utiliser la broche 6 de l'onduleur au lieu de la broche 3 (elles sont inverses l'une de l'autre). La complication est de monter les broches collecteur ouvert 3 (ou 6) et 5 de l'onduleur.

Ce modèle APC fournit une sortie non regulée de 24 V continu sur la broche 8. La tension de sortie est disponible tout le temps (au moins un peu après que le signal de batterie faible soit monté). L'intensite est limitee a 40mA. Pour monter, la broche 8 est l'alimentation d'un régulateur de tension de +5V. La sortie de ce régulateur passe dans deux resistances de 4,7kohm. L'autre bout d'une resistance connecte les broches 3 (Coupure de courant) de l'onduleur et 5 du PC (CTS). Celle de l'autre resistance connecte les broches 5 de l'onduleur (Batterie faible) et 8 du PC (DCD). Les deux resistances consomment environ 2 mA lorsqu'elles sont a la masse.

Lorsque l'onduleur est alimenté, les broches 5 (CTS) et 8 (DCD) côté PC doivent être très proches de 5V, et monter la broche 20 pendant 5 secondes ne doit avoir aucun effet. Lorsque l'onduleur passe sur batteries, la broche 5 (CTS) doit tomber à 0V, la broche 8 (DCD) doit rester à l'identique à 5V, et monter la broche 20 (DTR) en court-circuitant les broches 8 et 20, par exemple, doit éteindre l'onduleur après environ 15 secondes.

Lorsque la diode "Low Battery" du panneau frontal s'allume, la broche 8 (DCD) doit descendre à 0V aussi.

Les tensions de l'interface onduleur sont NEGATIVES pour la coupure de secteur (sur la broche 3 de l'onduleur) et la batterie faible, et POSITIVE pour l'arrêt à distance. Les paramètres de ligne série comme la vitesse n'ont aucune importance.

Liste du materiel necessaire :

- un boîtier DB-9 ;
- un connecteur sub-DB-25 femelle ;
- un régulateur de tension 7805 +5Vdc ;
- deux résistances de 4,7kohm ;
- un carte à composants perforée ;
- un câble avec au moins un connecteur 9 broches mâle.

Et de plus :

- un multimètre ;
- un fer à souder;
- quelques heures...

APC Smart-UPS 700

Voici quelques détails sur le fonctionnement du modèle 700 en mode bête, qui présente une utilisation futée d'un transistor placé dans le câble qui éteint l'onduleur lorsque l'ordinateur est éteint.

```
From: Markus Eiden <Markus@eiden.de>
Sender: eiden@eiden.de
To: "Harvey J. Stein" <abel@netvision.net.il>
Subject: Re: APC Smart-UPS
Date: Sun, 30 Mar 1997 16:21:05 +0200
```

J'utilise un APC Smart-UPS 700 pour mon système Linux sur une carte ASUS.

Pour utiliser quelques possibilites de l'onduleur, il faut quatre choses :

1) faire un câble RS-232 avec une petite interface ; 2) le source du powerd du paquetage sysvinit (j'utilise la version 2.6 de Miquel van Smoorenburg). Il faut ensuite modifier ce powerd ; 3) changer /etc/inittab ; 4) faire un script qui lance certaines commandes si le courant est coupé ou si la batterie est faible.

Quelques possibilités :

Lorsque le secteur est coupé, un script est lancé et une entrée est faite dans syslog.

Si la batterie est faible, un autre script est lancé (qui arête l'ordinateur, évidemment) et une entrée est faite dans syslog.

Si l'ordinateur est arrêté et que le courant l'est aussi, l'onduleur sera arrêté à son tour.

1) D'abord le câble :

Si l'on jette un coup d'oeil à l'arrière de l'onduleur, on y trouve un connecteur femelle comme celui-ci :

```
             8           1: Eteint l'onduleur lorsque le courant est coupe
                            et que la broche 1 est haute.
       X   X   X   X     3: Descend en cas de coupure de curant.
     X   X   X   X   X   4: Masse
                         5: Descend en cas de baisse de la batterie.
     1       3   4   5   8: +24V
```

D'un autre côté, l'arrière du PC présente un connecteur mâle comme celui-ci :

```
         8       6       1: DCD
       X   X   X   X     4: DTR
     X   X   X   X   X   5: GND
     5   4           1   6: DSR
                         8: CTS
```

Il faut réaliser l'interface suivant entre ces connecteurs :

```
PC                                                            UPS



                                               #------------------  (8)
                                               |
                                             470 Ohm
                                               |
                #-----#-----#-----#-----#-----#----- ca. 9-12V
                |     |     |     |     |     |
                47    3.3   3.3   3.3   1     470
                kOhm  kOhm  kOhm  kOhm  kOhm  Ohm
                |     |     |     |     |     |
(8) -----------------------#     |     |
                |     |     |           |     |
(6) ------------#------------------------------------------  (5)
                |           |           |     |
(1) ------------------#------------------------------------  (3)
                |                       |     |
                |                      C#------------------------  (1)
                |                      -|     |
                |                    B/       |
(4) ------#-----12kOhm---------|              |
                                \>E           |
                                 |            |
(5)----------------------------#-------#-------------------  (4)
```

- j'utilise un transistor "BC140", mais à peut près n'importe quel transistor NPN devrait faire l'affaire ;-)
- Le transistor fonctionne comme "inverseur". Si l'on éteint l'ordinateur ET que le courant est coupé, la broche 4 du PC descend et la broche 1 de l'onduleur monte. Cela éteint l'nduleur pour économiser la batterie.

2) Le source de *powerd*

J'ai juste retouché très peu le source (donc c'est en fait celui de Miquel).

(a) Emet une "alerte" vers syslogd si la broche 8 du PC (`DCD`) est basse (c'est qu'alors, le câble n'est pas connecté) ;

(b) `DCD` descendu à zéro -> le courant est coupé -> appel de `powerfail(0)` -> envoi de `INIT_CMD_POWERFAIL` au processus `init` ;

(c) `DCD` remonté -> le courant est revenu -> appel de `powerfail(1)` -> envoi de `INIT_CMD_POWEROK` au processus `init` ;

(d) `DSR` et `DCD` descendus à zéro -> le courant est coupé et la batterie est faible > appel de `powerfail(2)` -> envoi de `INIT_CMD_POWERFAILNOW` au processus `init`.

Et voilà.

```
/*
 * powerd       Surveille la ligne DCD d'un port serie connecte a un
 *              onduleur. Si le courant est coupe, previent init.
 *              Si le courant revient, previent init encore.
 *              Tant que le courant est la, DCD doit etre "haut". Lorsque
 *              le courant est coupe, DCD doit descendre.
 *              Powerd maintient DTR haut, donc en branchant une resistance
 *              de 10 kOhm entre DCD et DTR, l'onduleur ou un simple relais
 *              peuvent descendre DCD à la masse.
 *              Il faut aussi brancher DSR et DTR ensemble. Ainsi, powerd
 *              peut controler ici et la que DSR soit haut, et il sait donc
 *              que l'onduleur est connecte !!
 *
 * Usage:       powerd /dev/cua4 (ou tout autre port serie).
 *
 * Auteur:      Miquel van Smoorenburg, <miquels@drinkel.cistron.nl>.
 *              Quelques changements mineurs de Markus Eiden, <Markus@Eiden.de>
 *              pour APC-Smart-UPS-powerd.
 *
 * Version:     1.31,  29-Feb-1996.
 *
 * Traduction:  Bernard Choppy (choppy@imaginet.fr)
 *
 *              Ce programme fut developpe initialement pour mon employeur
 *                      ** Cistron Electronics **
 *              qui a autorise la distribution de celui-ci pour un usage
 *              generalise.
 *
 *              Copyright 1991-1996 Cistron Electronics.
 *
 *
 *              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.
 *
 *              Ce programme est un logiciel libre ; vous pouvez le diffuser
 *              et/ou modifier selon les termes de la GPL (GNU Public License)
 *              de la Free Software Foundation; au choix dans la version 2 de
 *              cette licence, ou (a votre choix) toute autre version.
 *
```

```
 *              Modifications mineures pour APC-powerd par Markus Eiden
 *              Markus@Eiden.de
 */

/* Utilisation de la nouvelle methode de communication avec init */
#define NEWINIT

#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <errno.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <signal.h>
#include <syslog.h>
#include <string.h>
#include "paths.h"
#ifdef NEWINIT
#include "initreq.h"
#endif

#ifndef SIGPWR
#  define SIGPWR SIGUSR1
#endif

#ifdef NEWINIT
void alrm_handler()
{
}
#endif

/* Avise init du changement d'etat du courant */
void powerfail(ok)
int ok;
{
  int fd;
#ifdef NEWINIT
  struct init_request req;

  /* Remplissage de la structure de requete */
  memset(&req, 0, sizeof(req));
  req.magic = INIT_MAGIC;


  /* INIT_CMD_* sont definis dans initreq.h                *
   * Jetez un coup d'oeil a init.c et /etc/inittab         *
   *                                                       *
   * ok=0 -> INIT_CMD_POWERFAIL      -> powerwait          *
   * ok=1 -> INIT_CMD_POWEROK        -> powerokwait        *
   * ok=2 -> INIT_CMD_POWERFAILNOW   -> powerfailnow       */


  switch (ok) {
```

```
      case 0 : req.cmd = INIT_CMD_POWERFAIL;
               /* Coupure -> alerte */
               break;
      case 1 : req.cmd = INIT_CMD_POWEROK;
               /* Retour du courant -> arrete l'alerte */
               break;
      case 2 : req.cmd = INIT_CMD_POWERFAILNOW;
               /* Coupure et batterie faible -> arret systeme */
               break;
               }

  /* Ouvre le fifo (avec timeout) */
  signal(SIGALRM, alrm_handler);
  alarm(3);
  if ((fd = open(INIT_FIFO, O_WRONLY)) >= 0
              && write(fd, &req, sizeof(req)) == sizeof(req)) {
        close(fd);
        return;
  }
  /* On en revient a l'ancienne methode... */
#endif

  /* Creaton d'un fichier info pour init */
  unlink(PWRSTAT);
  if ((fd = open(PWRSTAT, O_CREAT|O_WRONLY, 0644)) >= 0) {
        if (ok)
                write(fd, "OK\n", 3);
        else
                write(fd, "FAIL\n", 5);
        close(fd);
  }
  kill(1, SIGPWR);
}

/* Programme principal */
int main(int argc, char **argv)
{
  int fd;
  int dtr_bit = TIOCM_DTR;
  int flags;
  int status, oldstat = -1;
  int count = 0;
  int tries = 0;
  int powerfailed = 0;
  int rebootnow   = 0;

  if (argc < 2) {
        fprintf(stderr, "Usage: powerd <port>\n");
        exit(1);
  }

  /* Lancement de syslog */
  openlog("powerd", LOG_CONS|LOG_PERROR, LOG_DAEMON);

  /* Ouverture du port a surveiller */
```

```
if ((fd = open(argv[1], O_RDWR | O_NDELAY)) < 0) {
      syslog(LOG_ERR, "%s: %s", argv[1], sys_errlist[errno]);
      closelog();
      exit(1);
}

/* Port ouvert, DTR doit etre haut. On le force tout de meme...*/

/* Fonctionnement : Batterie faible -> Arret -> DTR descend -> *
 * transistor ouvert -> La broche d'arret onduleur monte ->    *
 * l'onduleur s'arrete apres 20 s environ. S'il y a une coupu- *
 * re et que l'ordinateur est eteint, l'onduleur s'arrete.     *
 * Si le courant revient, l'onduleur s'allume, l'ordinateur    *
 * demarre, et powerd est lance.                               *
 *                                                             */

ioctl(fd, TIOCMBIS, &dtr_bit);




/* Passe en daemon. */
switch(fork()) {
      case 0: /* Fils */
              closelog();
              setsid();
              break;
      case -1: /* Erreur */
              syslog(LOG_ERR, "impossible de forker.");
              closelog();
              exit(1);
      default: /* Pere */
              closelog();
              exit(0);
}

/* Relance syslog. */
openlog("powerd", LOG_CONS, LOG_DAEMON);

 syslog(LOG_INFO, "APCpowerd demarre...");

/* On surveille DCD */
while(1) {
      /* Lecture de l'etat. */
      ioctl(fd, TIOCMGET, &flags);

      /* Controle de connexion : CTS doit etre haut */
      tries = 0;
      /* TIOCM_*- Se reporter a  .../ams/termios.h */
      while((flags & TIOCM_CTS) == 0) {
              /* On continue a essayer, et alerte toutes les 2 minutes */
              if ((tries % 60) == 0)
                  syslog(LOG_ALERT, "Onduleur non connecte");
              sleep(2);
              tries++;
```

```
                ioctl(fd, TIOCMGET, &flags);
        }
        if (tries > 0)
                syslog(LOG_ALERT, "Onduleur reconnecte");

        /* Calcule l'etat en cours */
        status = (flags & TIOCM_CAR);

        /* Si DCD est passe a zero, le courant a ete coupe */
        if (oldstat != 0 && status == 0) {
                count++;
                if (count > 3) {
                        powerfailed = 1;
                        powerfail(0);
                        }
                else {
                        sleep(1);
                        continue;
                }
        }
        /* Si DCD remonte, le courant est revenu. */
        if (oldstat == 0 && status > 0) {
                count++;
                if (count > 3) {
                        powerfailed = 0;

                        /* eigentlich unnoetig: */
                        rebootnow = 0;

                        powerfail(1);
                        }
                else {
                        sleep(1);
                        continue;
                }
        }

        /* Batterie faible et courant coupe ? */
        if (rebootnow==0)
        if (powerfailed==1)
        if ((flags & TIOCM_DSR) == 0)
        {
           rebootnow=1;
           powerfail(2);

        }

        /* Reinitialisation, stockage de l'etat et attente 2s. */
        count = 0;
        oldstat = status;
        sleep(2);
  }
  /* N'arrive jamais */
  return(0);
}
```

3) Modifier inittab

`init` reçoit les commandes `INIT_CMD` et lance les scripts idoines :

```
pf::powerwait:/sbin/init.d/powerfail    start
pn::powerfailnow:/sbin/init.d/powerfail now
po::powerokwait:/sbin/init.d/powerfail  stop
```

Ce qui signifie, par exemple : si le courant est coupé (`powerwait`, lancer le script `/sbin/init.d/powerfail` avec le paramètre "`start`".

4) Le script `powerfail`

```
#! /bin/sh
# Copyright (c) 1997 Markus Eiden, Markus@Eiden.de
#

case "$1" in
    start)
        echo "LE COURANT EST COUPE !" | wall
        logger "Coupure de courant"
        ;;
    now)
        echo "BATTERIE FAIBLE ! Arret systeme dans une minute" | wall
        logger "Batterie faible, arret systeme dans une minute"
        sync
        /sbin/shutdown -r -t 5 +1
        ;;
    stop)
        echo "LE COURANT EST REVENU !!" | wall
        logger "Courant retabli"

        /sbin/shutdown -c >/dev/null 2>/dev/null

        ;;
  *)
        echo "Usage: $0 {start|now|stop}"
        exit 1
        ;;
esac

exit 0
```

Eh bien, cela devrait être simple ;-)

Vous voilà prêt maintenant, mais restez prudent : cela fonctionne pour moi, mais je ne peux évidemment pas garantir que quoi que ce soit de cela fonctionne pour vous.

Un petit conseil pour finir : si `/sbin/init.d/powerfail` arrête votre PC, DTR descend, donc la broche d'arrêt (côté onduleur) monte. Dès cet instant, il faut entre 20 et 30 secondes à l'onduleur pour s'arrêter. C'est de votre responsabilité d'empêcher votre machine Linux de redémarrer durant ces 20 secondes (en particulier, de monter les volumes disque). Cela ne fut pas un problème pour mon système.

Quatre méthodes simples permettent d'empêcher Linux de démarrer rapidement :

1. le BIOS doit réaliser certaines routines (comme identifier le nombre de pistes de votre lecteur de disquettes si vous en avez un) ;

2. LILO peut être configuré pour attendre s'il est installé ;
3. il peut ne rien y avoir à faire (comme dans mons cas) ;
4. il est possible d'acheter plus de mémoire afin que le décompte dure 30 secondes. Cela doit correspondre environ à 1 024 Mo ;-).

APC Smart-UPS 1 400

Autre jour, autre APC. Voici pour le *Smart-UPS 1 400*, en mode bête.

```
From: "Slavik Terletsky" <ts@polynet.lviv.ua>
To: hjstein@math.huji.ac.il
Subject: my contribution to UPS HOWTO
Date: Mon, 27 Jan 1997 21:10:16 +0000
```

Daemon d'onduleur pour FreeBSD (2.1.5 - testé).

Schéma de branchement :

```
Onduleur (broche, nom)          PC (broche, nom)
----------------------          ---------------------
1 Arret                 >----------->   4 Terminal pret
2 Courant Coupe         >----------->   8 Pret a emettre
4 Commun                >----------->   5 Masse
5 Batterie faible       >----------+>   1 Detection de porteuse
8 Batterie (+24V)       >-|10kOhm|-+
```

Description

```
Usage: upsd <device> [wait [script]]

device  - device name upsd interacts thru (e.g. /dev/cuaa1)
wait    - time (secs) to wait before running script, (default value 0 sec).
script  - system shutdown script (default /etc/rc.shutdown).
```

Fonctionnement :

`upsd` enregistre tous les changements d'état de l'onduleur (courant présent ou absent, batterie faible ou bonne). Lorsque le courant est absent et que la batterie est faible, `upsd` active le signal d'arrêt de l'onduleur, attend le nombre de secondes indiqué sur la ligne de commande, et lance le script d'arrêt.

Exemple de script :

```
#!/bin/sh
# Le script est execute lorsque le systeme s'arrete

PATH=/sbin:/bin:/usr/sbin:/usr/bin

echo "ARRET IMMEDIAT DU SYSTEME" | wall

reboot
```

Source d'`upsd` :

```
/* daemon d'onduleur
 * Copyright 1997 Slavik Terletsky. All rights reserved.
 * Auteur: Slavik Terletsky <ts@polynet.lviv.ua>
 * Systeme: FreeBSD
 * Traduction: Bernard Choppy <choppy@imaginet.fr>
```

```
 */
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <syslog.h>
#include <unistd.h>
#include <varargs.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/uio.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/ttycom.h>

int status;
int wait = 0;
FILE *fd;
char *scr = "/etc/rc.shutdown";
char *idf = "/var/run/upsd.pid";

void upsterm();
void upsdown(int);

int main(int argc, char *argv[]) {
 int pd;
 int zero = 0;
 char d5, d6, d7;
 char low = 0;
 char pow = 1;

 /* controle des arguments */
 switch(argc) {
 case  4:
 scr = argv[3];
 case  3:
 wait = atoi(argv[2]);
 case  2:
 break;
 default:
 fprintf(stderr, "usage: %s <port> [temporisation [script]]\n", argv[0]);
 exit(1);
 }

 /* controle de l'existence du script */
 if(!(fd = fopen(scr, "r"))) {
 fprintf(stderr, "fopen: %s: %s\n", scr, sys_errlist[errno]);
 exit(1);
 }
 fclose(fd);

 /* controle si upsd s'execute deja */
 if(fd = fopen(idf, "r")) {
 fprintf(stderr, "fopen: le fichier %s existe deja\n", idf);
 exit(1);
 }
```

```
/* passe en daemon */
switch(fork()) {
case -1:       /* erreur */
fprintf(stderr, "fork: %s\n", sys_errlist[errno]);
exit(1);
case  0:       /* fils */
break;
default:       /* pere */
exit(0);
}

/* sauvegarde du pid */
if(!(fd = fopen(idf, "w"))) {
fprintf(stderr, "fopen: %s: %s\n", idf, sys_errlist[errno]);
exit(1);
}
fprintf(fd, "%d\n", (int)getpid());
fclose(fd);

/* ouverture du port a surveiller */
if((pd = open(argv[1], O_RDWR | O_NDELAY)) < 0) {
fprintf(stderr, "open: %s: %s\n", argv[1], sys_errlist[errno]);
exit(1);
}

/* le daemon fonctionne */
openlog("upsd", LOG_PID, LOG_DAEMON);
syslog(LOG_INFO, "daemon demarre");

/* reaction au signal */
(void)signal(SIGTERM, upsterm);

/* surveillance du port */
while(1) {
/* reinitialisation des bits */
if(ioctl(pd, TIOCMSET, &zero) < 0) {
 fprintf(stderr, "ioctl: %s\n", sys_errlist[errno]);
 exit(1);
}

/* lecture de l'etat du port */
if(ioctl(pd, TIOCMGET, &status) < 0) {
 fprintf(stderr, "ioctl: %s\n", sys_errlist[errno]);
 exit(1);
}

/* determination de l'etat */
d5 = status & 0x20;
d6 = status & 0x40;
d7 = status & 0x80;

/* courant present */
if(!(d7 + d5)) {
 if(!pow) {
```

```
  syslog(LOG_CRIT, "courant present");
  pow = 1;
 }
/* courant coupe */
} else {
 if(pow) {
  syslog(LOG_CRIT, "courant coupe");
  pow = 0;
 }
}

/* batterie faible */
if(!d6 && !low) {
 syslog(LOG_ALERT, "batterie faible");
 low = 1;

 /* arret onduleur */
 if(!pow) {
  upsdown(pd);
 }
}

/* batterie ok */
if(d6 && low) {
 syslog(LOG_CRIT, "batterie ok");
 low = 0;
}

sleep(1);
}

/* jamais atteint */
return 0;

}

void upsterm() {
/* message de trace de fin */
syslog(LOG_INFO, "arret du daemon");

/* effacement du fichier de pid */
unlink(idf);

exit(0);
}

void upsdown(int pd) {
/* message de trace d'arret */
syslog(LOG_ALERT, "arret du systeme");

/* effacement du fichier de pid */
unlink(idf);

/* mesure de securite : vidange des tampons d'ecriture */
system("/bin/sync");
```

```
 system("/bin/sync");
 system("/bin/sync");

 /* arret de l'onduleur */
 status = TIOCM_DTR;
 if(ioctl(pd, TIOCMSET, &status) < 0) {
 fprintf(stderr, "ioctl: %s\n", sys_errlist[errno]);
 exit(1);
 }

 /* attente puis lancement du script */
 sleep(wait);
 system(scr);
}
# Slavik Terletsky      # University "Lvivska Poytechnika" #
# Network Administrator # mailto:ts@polynet.lviv.ua        #
```

9 Comment eteindre d'autres machines sur le même onduleur ?

Certaines personnes (y compris moi-même), ont plusieurs PC Linux connectés sur un onduleur. Un PC contrôle l'onduleur et doit éteindre les autres PC lorsque le secteur est coupé.

Nous supposons que les PC peuvent communiquer sur un réseau. Appelons le PC qui surveille l'onduleur le maître, et les autres PC, les esclaves.

Dans les temps anciens, cela nécessitait une amusante programmation.

Maintenant, la meilleure chose à faire semble être de trouver l'un des paquetages `powerd-2.0.tar.gz` ou `upsd-1.0.tgz` cités à la section 4 (Logiciels) et de suivre les instructions. Les deux sont capables de fonctionner sur les esclaves dans un mode qui les connecte à un processus `powerd` ou `upsd` s'exécutant sur le maître pour lui demander l'état de l'onduleur. Certains des paquetages spécifiques pour APC semblent disposer aussi de cette fonctionnalité.

Néammoins, si votre réseau n'est pas sûr, vous pouvez être amené à souhaiter plus de sécurité dans ce montage, puisqu'il est possible de "pirater" un `powerd` esclave pour lui faire croire que le courant est coupé.

Une autre possibilité est d'utiliser le protocole SNMP (Simple Network Management Protocol - protocole simplifié d'administration de réseau). Le détail de l'utilisation de SNMP dépasse le cadre de ce document, pour ne pas dire que cela me dépasse tout court actuellement.

9.1 Méthode de l'état du port

Configurez un port sur le maître qui, lorsqu'on y est connecté, envoie soit "OK", soit "FAIL", soit "BATLOW" lorsque le courant est là, qu'il est coupé, ou que la batterie est faible, respectivement. Montez cela sur le port 13 (le port time) sur lequel vous pouvez faire un telnet et recevoir l'heure locale.

Montez sur les esclaves une version de *powerd* qui lit ce port plutôt que de contrôler une ligne série.

Je pense que c'est probablement la meilleure méthode, et j'ai l'intention d'upgrader mes systèmes pour l'utiliser.

9.2 Méthode d'émission en l'air

Identique à la section 9.1 (Méthode de l'état du port), mais par émission d'un message broadcast Ethernet signifiant l'événement.

Cela peut avoir des implications de sécurité, puisqu'il peut être {{spoofed}}

9.3 Méthode du pseudo-login

Configurez les pseudo-login sur les esclaves avec les noms `powerok` et `powerfail`, tous les deux avec le même UID. Faites de `/etc/powerokscript` le shell du user powerok, et de `/etc/powerfailscript` celui du user powerfail. Sur le maître, faites en sorte que le script `/etc/powerokscript` fasse un rlogin sur chaque esclave en tant que user powerok et que le script `/etc/powerfailscript` fasse un rlogin en tant que powerfail sur chaque esclave. Placez un fichier `.rhosts` sur chaque esclave dans le répertoire par défaut de powerok et powerfail pour autoriser le root du master à entrer comme users powerok et powerfail sur chaque esclave.

C'est le système que j'utilise actuellement. Malheureusement, j'ai quelques difficultés à faire que les login distants s'exécutent et rendent la main sans se bloquer. Il faudrait probablement que le script `/etc/powerfailscript` fasse les rsh sur les esclaves en tâche de fond pour lui éviter de bloquer. Néammoins, je n'ai jamais obtenu de login correct en tâche de fond. J'ai même essayé des combinaisons complexes comme faire se loger `toto` sur l'esclave. Tout ce que j'ai utilisé avait des problèmes et se trouvait bloqué par un entrée tty (ou sortie, je ne m'en rappelle plus).

En plus, cela peut créer des trous de sécurité.