Mise en place de diald pour Linux - un exemple

GNU Copyright par Harish Pillay, *h.pillay@ieee.org* version française par Xavier Serpaggi 1996

Ce qui suit est un exemple de configuration que j'ai faite fonctionner sur plusieurs machines et qui autorise l'installation automatique de diald au moment du démarrage. Quand une demande de connexion se présente, il téléphone automatiquement à mon FAI (Fournisseur d'Accès Internet) préféré, Pacific Internet ici à Singapour.

Table des matières

1 Fichiers de configuration

Merci beaucoup à Eric Schenk pour le super outil qu'est diald.

1.1 Fichier /etc/rc.d/rc.local

```
#! /bin/sh
# Mettez ici toutes les commandes de configuration locale
# On lance gpm
echo "Lancement de gpm..."
gpm -t ms &

# démarrage d'innd
/etc/rc.d/rc.news

# démarrage du démon httpd du CERN
echo "Démarrage de CERN httpd avec proxy et cache."
/usr/local/bin/httpd

# chargement des modules nécessaires
/etc/rc.d/rc.modules

# démarrage de diald
echo "Démarrage du démon diald pour Pacific Internet ..."
cd /usr/lib/ppp
```

```
/usr/lib/ppp/diald.pacific.internet
```

```
/usr/lib/ppp/diald.pacific.internet


/usr/sbin/diald /dev/ttyS1 /dev/ttyS1 lock debug 20 -m ppp local 127.0.0.2 \
  remote 127.0.0.3 defaultroute modem crtscts \
  connect "chat -v -f /usr/lib/ppp/pppchat.pi" \
  dynamic -- debug noipdefault
```

```
/usr/lib/ppp/pppchat.pi


ABORT "NO CARRIER"
ABORT BUSY
"" ATZ OK
ATm1s50=255s111=0DT1-800-555-1212
CONNECT ""
login MONNOMDELOGIN
ssword MOMMOTDEPASSE
```

1.2 Script PPP à l'aveuglette

- Amorçage de minicom.
- Connexion au fournisseur d'accès
- Invoquer PPP de l'autre côté
- Suspendre minicom (alt-A-J).
- Invoquer le script suivant

```
#!/bin/sh
#       Mise en place d'une liaison PPP à l'aveuglette - script appelé blind.ppp

DEVICE=ttyS1
(
        stty 38400 -tostop
        pppd -detach debug noipdefault defaultroute mru 1500 /dev/$DEVICE &
        exit 0
)  /dev/$DEVICE
```

1.3 Fichier /etc/diald.conf

Mon fichier /etc/diald.conf (inchangé par rapport à la distribution originale) :

```
# Voici un ensemble de règles de filtrages assez compliquées
# (Ce sont les règles que j'utilise.)
#
# J'ai décomposé les règles en quatre sections.
# Les trames TCP, les trames UDP, les trames ICMP et enfin une règle
# générale pour tout le reste.
#------------------------------------------------------------------------------
```

```
# Règles pour les trames TCP.
#-------------------------------------------------------------------------------
# Commentaires généraux sur cet ensemble de règles :
#
# En général nous n'aimerions traiter que les données d'une liaison TCP
# ayant un sens pour le temps de déconnexion. Cependant, nous essayons
# d'ignorer les trames sans donnée.
# Puisque la taille minimale d'un en-tête dans une trame TCP est de 40 octets,
# toutes les trames d'une longueur de 40 ne devraient pas contenir de donnée.
# De cette manière il est possible de manquer des trames vides (des informations
# optionnelles de routage et d'autres choses supplémentaires peuvent être
# présentes dans un en-tête IP), mais nous devrions en capter la majorité.
# Remarquez que nous ne voulons pas rejetter les trames avec un champ tcp.live
# vide puisque nous les utiliserons plus tard pour accélérer la déconnexion de
# certaines liaisons TCP.
#
# Nous voulons également nous assurer que les trames WWW restent en vie même
# si la socket TCP est arrétée. Nous faisons cela parce que WWW ne garde pas la
# connexion une fois que les données ont-été transférées, et il serait
# génant d'avoir la liaison qui se créee et se coupe à chaque document.
#
# En dehors de WWW l'utilisation la plus courante de TCP est pour les
# connexions de longues durées dont la coupure signifie que vous n'allez plus
# avoir besoin du réseau.
# Nous ne voulons pas nécessairement avoir à attendre 10 minutes que la
# connexion se termine alors que nous n'avons ni telnet ni rlogin en cours, donc
# nous voulons accélérer le délai de déconnexion sur les liaisons TCP qui sont
# terminées. Ceci est réalisé en mettant dans un cache les trames qui n'ont pas l'indicateur
# live positionné.

# --- début proprement dit de l'ensemble de règles ---

# Quand on débute une connexion on ne donne tout d'abord au lien que 15
# secondes. L'idée ici est de pouvoir éventuellement se rendre compte que le
# réseau de l'autre côté de la connexion n'est pas accessible. Dans ce cas
# il n'est pas nécessaire de donner un temps de vie de 10 minutes au lien.
# Avec la règle ci-dessous nous ne lui donnons initialement que 15 secondes.
# Si le réseau est accessible alors nous devrions normalement recevoir une
# réponse contenant des données dans les 15 secondes. Si cela pose un problème
# parce que vous avez des réponses lentes de la part de certains sites que vous
# visitez régulièrement, vous pouvez augmenter le temps avant déconnexion ou
# bien supprimer cette règle.
accept tcp 15 tcp.syn

# Empécher named de garder la connexion active.
ignore tcp tcp.dest=tcp.domain
ignore tcp tcp.source=tcp.domain

# (Argh! Le telnet de SCO commence par envoyer des SNY vides et n'initie la
# connexion que s'il obtient une réponse. Pfuuutt...
```

```
accept tcp 5 ip.tot_len=40,tcp.syn

# Empécher les trames vides de maintenir le lien actif (autres que les trames
# vides SNY).
ignore tcp ip.tot_len=40,tcp.live

# On s'assure que le transfert http maintient la ligne active pendant 2
# minutes, même après que ce soit terminé.
# REMARQUE : votre fichier /etc/services ne devrait pas définir le service tcp
# www, auquel cas vous devez commenter les deux lignes suivantes et vous
# procurer un fichier /etc/services plus récent. Lisez la FAQ pour savoir
# comment obtenir un nouveau fichier /etc/services.
accept tcp 120 tcp.dest=tcp.www
accept tcp 120 tcp.source=tcp.www

# Une fois que le lien n'est plus actif, nous tentons de stopper la connexion
# rapidement. Remarquez que si le lien est déjà arrété, un changement d'état
# ne le ramènera pas à l'état actif.
keepup tcp 5 !tcp.live
ignore tcp !tcp.live

# une donnée ftp ou une connexion ftp peut être attendue pour rendre compte
# du trafic relativement fréquent.
accept tcp 120 tcp.dest=tcp.ftp
accept tcp 120 tcp.source=tcp.ftp

# REMARQUE : les données ftp ne sont pas définies dans le fichier /etc/services
# distribué dans les dernières versions de NETKIT, donc j'ai commenté ce
# passage.
# Si vous désirez le définir, ajoutez la ligne suivante à votre fichier
# /etc/services :
# ftp-data        20/tcp
# et décommentez les deux règles suivantes
#accept tcp 120 tcp.dest=tcp.ftp-data
#accept tcp 120 tcp.source=tcp.ftp-data

# Si nous n'avons pas réussi à l'avoir avec les règles au dessus, donnons au
# lien 10 minutes de plus.
accept tcp 600 any

# Règles pour les trames UDP.
#
# Nous donnons dès à présent un temps limite aux requettes de domaine puisque
# nous voulons juste qu'elles établissent le lien, pas qu'elles le maintiennent
# pour un long moment.
# Ceci parce que le réseau sera généralement établi par un appel de la
# bibliothèque de résolution de nom (à moins que vous n'ayez mis toutes les
# adresses que vous utilisez fréquemment dans /etc/hosts, auquel cas vous
# découvrirez d'autres problèmes.)
# Remarquez que vous ne devez pas donner une valeur de temps limite de
```

```
# déconnexion plus courte que le temps supposé que va mettre votre DNS pour
# répondre. Sinon quand le lien original s'établit, il risque d'y avoir
# une attente supérieure à celle qu'il y a entre les séries de trames initiales
# avant qu'une trame destinée à maintenir le lien passe par ce dernier.

# Ne pas activer le lien pour rwho.
ignore udp udp.dest=udp.who
ignore udp udp.source=udp.who
# Ne pas activer le lien pour RIP.
ignore udp udp.dest=udp.route
ignore udp udp.source=udp.route
# Ne pas activer le lien pour NTP ou pour timed.
ignore udp udp.dest=udp.ntp
ignore udp udp.source=udp.ntp
ignore udp udp.dest=udp.timed
ignore udp udp.source=udp.timed
# Ne pas activer les requètes de nom de domaine entre deux exécutions de named.
ignore udp udp.dest=udp.domain,udp.source=udp.domain
# Activer le réseau pour les requètes de domaine qui ne viennent pas de
# named.
accept udp 30 udp.dest=udp.domain
accept udp 30 udp.source=udp.domain
# Faire la même chose pour les diffusions de netbios-ns
# REMARQUE : votre fichier /etc/services peut ne pas définir le service
# netbios-ns, auquel cas vous devez commenter les trois lignes suivantes.
ignore udp udp.source=udp.netbios-ns,udp.dest=udp.netbios-ns
accept udp 30 udp.dest=udp.netbios-ns
accept udp 30 udp.source=udp.netbios-ns
# empécher les transferts de routed et gated de maintenir le lien actif
ignore udp tcp.dest=udp.route
ignore udp tcp.source=udp.route
# Le reste a droit à 2 minutes.
accept udp 120 any

# Récupérer toutes trames que nous n'avons pas traitées auparavant et donner 30
# secondes de durée de vie à la connexion.
accept any 30 any
```

1.4 Fichier diald.defs

Mon fichier diald.defs (inchangé par rapport à la distribution originale) :

```
# Définition des règles de protocole par défaut
prule tcp tcp 9:12:13:14:15:16:17:18:19:+0:+1:+2:+3:9:9:9
prule udp udp 9:12:13:14:15:16:17:18:19:+0:+1:+2:+3:9:9:9
prule icmp icmp 9:12:13:14:15:16:17:18:19:9:9:9:9:9:9:9
prule any any 9:12:13:14:15:16:17:18:19:9:9:9:9:9:9:9
# Définition des en-têtes des trames internet.
var ip.ihl 0(24)&0xf
var ip.version 0(28)&0xf
```

```
var ip.tos 1(24)&0xff
var ip.tot_len 2(16)&0xffff
var ip.id 4(16)&0xffff
var ip.frag_off 6(16)&0x3fff
var ip.ttl 8(24)&0xff
var ip.protocol 9(24)&0xff
var ip.check 10(16)&0xffff
var ip.saddr 12
var ip.daddr 16
# Définition des en-têtes des trames TCP.
var tcp.source +0(16)&0xffff
var tcp.dest +2(16)&0xffff
var tcp.seq +4
var tcp.ack_seq +8
var tcp.doff +12(28)&0xf
var tcp.fin +13(24)&0x1
var tcp.syn +13(25)&0x1
var tcp.rst +13(26)&0x1
var tcp.psh +13(27)&0x1
var tcp.ack +13(28)&0x1
var tcp.urg +13(29)&0x1
var tcp.live +127
# Définition de en-têtes des trames UDP.
var udp.source +0(16)&0xffff
var udp.dest +2(16)&0xffff
var udp.len +4(16)&0xffff
var udp.check +6(16)&0xffff
# Définition des en-têtes des trames ICMP.
var icmp.type +0(24)&0xff
var icmp.code +1(24)&0xff
var icmp.checksum +2(16)&0xffff
var icmp.echo.id +4(16)&0xffff
var icmp.echo.sequence +6(16)&0xffff
var icmp.gateway +4
```

2 Ma configuration

- Un 486/66 avec 20 Mo de ram et plein d'espace disque :-)
- des UART 16550 en abondance
- diald 0.11, pppd 2.2.0d, noyau 1.3.95 avec modules.

3 Remarques

- Changez le nom du port série /dev/ttyS1 1.1 (ci-dessus) de manière à ce qu'il corresponde à vos attentes.
- Assurez vous que vous pouvez vous connecter avec PPP en utilisant le script blind.ppp.

Si vous avez des questions écrivez à Harish Pillay à *h.pillay@ieee.org*