Linux NET-3-HOWTO, Réseau sous Linux.

Terry Dawson, VK2KTJ, Alessandro Rubini (mainteneur), alessandro.rubini@linux.it (Traduction et trahison de Jacques.Chion@wanadoo.fr, un grand merci à Jean-Albert Ferrez pour son aide) v1.3, 1 Avril 1998.

Le système Linux possède un support réseau inclus dans le noyau et écrit presque entièrement à partir de zéro. Les performances de l'implémentation tcp/ip des derniers noyaux en font une alternative digne de respect même vis à vis de ses meilleurs concurrents. Le but de ce document est de décrire comment installer et configurer le logiciel de réseau sous Linux, ainsi que les outils nécessaires.

1 Changements depuis la version 1.2

```
Additions:
    Informations sur transproxy
    Petits changements ca et la
Corrections/Mises a jour:
    Nouveau mainteneur
A faire (sans changements, helas):
    Ajouter le metteur en forme de trafic
    Decrire le nouvel algorithme de routage
    Ajouter les options de compilation du noyau sur IPv6
    Decrire les entrees de /proc/sys/net/*
    Pilote de peripherique WanRouter
```

2 Introduction.

Le premier document NET-FAQ fut écrit par Matt Welsh et Terry Dawson pour répondre aux questions fréquemment posées concernant les réseaux sous Linux, en un temps où le LPD (Linux Documentation Project) n'existait pas. Il s'agissait alors des toutes premières versions de développement du noyau réseau sous Linux. Le document NET-2-HOWTO, qui succéda au NET-FAQ, fut l'un des premiers documents du LDP HOWTO et il traitait de ce qui fut appelé version 2, et plus tard version 3, du logiciel réseau du noyau Linux. A son tour ce document prend la suite et ne traite que de la version 3 du noyau réseau Linux.

Les versions précédentes de ce document étaient devenues plutôt énormes en raison du grand nombre de sujets abordés. Pour résoudre ce problème un certain nombre de documents HOWTO ont été créés et traitent de sujets spécifiques. Ce document fait référence à ceux qui sont pertinents et aborde les sujets qui ne sont pas encore couverts par d'autres documents.

En avril 1998, Terry abandonna le maintien du NET-3 en raison de sa grande charge de travail. Je suis novice dans ce type de travail, mais ferai de mon mieux pour y arriver.

2.1 Retour d'informations

J' apprécie toujours les retours d'informations et tout particulièrement les contributions utiles. S'il vous plait adressez tout retour d'informations ou contributions à *l'adresse* (`mailto:rubini@linux.it`).

3 Comment utiliser ce document (NET-3-HOWTO-HOWTO?).

La présentation de ce document diffère des versions précédentes: Nous avons regroupé les chapitres de telle sorte qu'il y ait au début une information sur le sujet traité (ainsi vous pourrez le sauter s'il ne vous intéresse pas), puis le coeur du sujet, que vous devez être sûrs d'avoir assimilé avant d'aller dans les sections suivantes et qui traitent de la technique.

Lire les sections générales

Ces paragraphes s'appliquent à chaque technologie, ou presque, décrite plus tard, il est donc important que vous les ayez compris.

Réfléchissez à votre réseau

Vous devez savoir comment votre réseau est, ou sera, conçu et quels matériels et types de technologies vous utiliserez.

Lisez les sections qui concernent la technologie correspondant plus particulièrement à vos besoins.

Si vous savez ce que vous voulez, vous pouvez pointer sur chaque élément tour à tour. Ces paragraphes traitent seulement de détails concernant une technologie particulière.

Configurez votre réseau

Si vous allez réellement essayer de configurer votre réseau, prenez soigneusement note de tout problème éventuel.

Cherchez de l'aide si nécessaire

Si vous rencontrez des problèmes qui ne sont pas traités dans ce document, reportez-vous au paragraphe donnant les endroits où l'on peut en obtenir ou bien envoyer des reports de bogues.

Amusez-vous!

Le réseau est amusant, profitez-en.

4 Informations générales sur le réseau sous Linux.

4.1 Brève histoire du développement du noyau du réseau Linux.

Développer une nouvelle implémentation noyau de l'ensemble du protocole tcp/ip, de qualité, et qui marcherait aussi bien que les produits existants, n'était pas une tâche facile. La décision de ne pas partir d'une implémentation existante fut prise à un moment où il y avait un doute sur d'éventuelles restrictions sur les

droits de copie, en raison de décisions de justice U.S., et à un moment où il y avait beaucoup d'enthousiasme pour faire différemment et peut-être même mieux que ce qui avait été fait auparavant.

Le premier volontaire pour diriger le développement fut Ross Biro <biro@yggdrasil.com>. Ross produisit une implémentation de routines simple, incomplète, mais parfaitement utilisable, à laquelle fut ajouté un pilote Ethernet pour la carte interface réseau WD-8003. Ce fut suffisant pour que beaucoup de personnes essayent le logiciel et même certains s'arrangèrent pour se connecter, avec cette configuration, sur le réseau Internet en direct. La pression de la communauté Linux qui s'occupait du développement du support réseau augmenta, et pour finir, la convergence de cette pression injuste et de ses propres obligations l'emportèrent sur les avantages que Ross en tirait; il arrêta donc sa tâche de coordinateur de développement. Les efforts de Ross pour faire démarrer le projet, son acceptation de la responsabilité de faire vraiment quelque chose d'utile dans de telles circonstances mouvementées, furent le point de départ de tout le travail ultérieur et donc un élément essentiel du succès du produit actuel.

Orest Zborowski <obz@Kodak.COM> produisit la première interface socket BSD pour le noyau Linux. Ce fut un grand pas en avant et permit à beaucoup d'applications réseau existantes d'être portées sous Linux sans grandes modifications.

A peu près à cette époque Laurence Culhane <loz@holmes.demon.co.uk> développa les premiers pilotes Linux pour supporter le protocole SLIP. Ceci permit à beaucoup de gens qui n'avaient pas accès à un réseau Ethernet d'essayer le logiciel réseau. Puis certains utilisèrent ce pilote pour se connecter sur l'Internet. Cela donna à encore plus de personnes un aperçu de ce qui serait possible si Linux avait un support complet pour le réseau et augmenta le nombre d'utilisateurs utilisant et expérimentant ce logiciel réseau.

L'une des personnes qui a aussi activement travaillé sur la construction du support réseau fut Fred van Kempen <waltje@uwalt.nl.mugnet.org>. Après la période d'incertitude qui suivit le retrait de Ross, Fred offrit son temps et accepta le rôle de conducteur du développement sans rencontrer d'opposition. Fred avait quelques projets ambitieux quant à la direction vers laquelle il voulait porter le logiciel réseau Linux, et il se mit à progresser dans ces directions. Fred produisit une série de code réseau appelée le code noyau 'NET-2' (le code 'NET' étant celui de Ross), qui permit à beaucoup de personnes de l'utiliser avec intérêt. Ensuite Fred mit nombre d'innovations dans la poursuite du développement, telle que l'interface de périphérique dynamique, le support du protocole radio-amateur AX-25 et une implémentation réseau conçue de manière plus modulaire. Le code NET-2 de Fred fut utilisé par un grand nombre d'enthousiastes, ce nombre augmentant au fur et à mesure de l'utilisation du logiciel dans le monde. Le logiciel réseau à ce moment était constitué encore d'un grand nombre de patches qui devaient être appliqués au code noyau et n'était pas inclus dans la distribution normale. Le document NET-FAQ et son successeur NET-2-HOWTO décrivait la procédure assez complexe pour que tout cela fonctionne. Fred se concentra sur le développement d'innovations et cela prenait du temps. La communauté des utilisateurs s'impatientait, car elle voulait avoir quelque chose fonctionnant correctement et qui satisferait 80% des utilisateurs puis, comme avec Ross, la pression sur le responsable du développement augmentait.

Alan Cox <iialan@www.uk.linux.org> proposa une solution pour améliorer la situation. Il proposa de prendre le code NET-2 de Fred, de le déboguer, de le rendre fiable et stable si bien qu'il satisferait l'utilisateur de base impatient, relâchant ainsi la pression sur Fred qui pourrait continuer son oeuvre. Alan se mit au travail avec un certain succès et sa première version du code réseau Linux fut appelée 'Net-2D(ebugged;)'. Le code fonctionnait de manière fiable avec plusieurs configurations typiques et l'utilisateur de base était content. Alan avait vraiment des idées et une compétence à lui pour contribuer au projet et de nombreuses

discussions concernant la direction que devait prendre le code NET-2 furent suivies d'effet. Il se développa alors deux écoles distinctes dans la communauté Linux, l'une ayant pour principe 'que ça marche d'abord, puis on améliorera ensuite' et l'autre 'améliorer d'abord'. Linus arbitra finalement et offrit son aide aux efforts de développement d'Alan et inclut son code dans la distribution standard du noyau. Cela plaçait Fred dans une situation délicate. Tout développement de longue haleine souffrirait de l'absence d'utilisation et d'essais par l'utilisateur de base et cela signifierait que les progrès seraient longs et difficiles. Fred continua à travailler encore quelque temps, puis se retira finalement, et Alan devint le nouveau pilote de développement du code réseau dans le noyau Linux.

Donald Becker <becker@cesdis.gsfc.nasa.gov> révéla rapidement ses talents dans les aspects de bas niveau du réseau et produisit une énorme quantité de pilotes Ethernet, presque tous ceux inclus dans les noyaux actuels étant lui. Il y a d'autres personnes qui ont apporté une contribution significative, mais le travail de Donald est prolifique et mérite donc une mention spéciale.

Alan continua à affiner le code NET-2-D(ebugged) pendant un certain temps, tout en progressant sur certains des sujets qui restaient en suspens dans la liste des 'TODO' (NdT: 'A Faire'). Pendant que les sources du noyau Linux `1.3.*` faisaient leurs premiers pas, le code réseau migra vers la distribution NET-3, sur laquelle les versions actuelles sont basées. Alan travailla sur de multiples aspects du code réseau et, avec l'assistance d'un grand nombre de personnes talentueuses venant de la communauté Linux, développa le code dans toutes sortes de direction. Alan produisit des pilotes de périphériques réseau, le premier standard AX.25 et les implémentations IPX. Alan continua à rafistoler le code, le restructurant petit à petit et l'amenant à son niveau d'aujourd'hui.

Le support PPP fut ajouté par Michael Callahan <callahan@maths.ox.ac.uk> et Al Longyear <longyear@netcom.com>, ce qui fut important pour accroître le nombre de personnes utilisant Linux désireuses d'aller sur le réseau.

Jonathon Naylor <jsn@cs.nott.ac.uk> apporta sa contribution en améliorant le code AX.25 d'Alan et en y ajoutant les protocoles NetRom et Rose. Le support AX.25/NetRom lui-même est tout à fait significatif, car aucun autre système d'exploitation que Linux ne peut se vanter d'avoir un support natif pour ce protocole.

Il y a eu bien sûr des centaines d'autres personnes qui ont apporté une contribution significative à la couche réseau de Linux. Vous en retrouverez certains plus tard dans les paragraphes traitant de technologies spécifiques, d'autres ont collaboré aux modules, pilotes, corrections de bogues, suggestions, rapports d'essais et support moral. Dans tous les cas chacun peut se prévaloir d'avoir joué un rôle et offert ce qu'il pouvait. Le code réseau Linux est un excellent exemple de ce que l'on peut obtenir avec un style Linux de développement anarchique, si cela ne vous a pas encore surpris, et on le voit encore, le développement ne s'est pas arrêté.

4.2 Où obtenir d'autres informations sur la couche réseau de Linux.

Il y a un grand nombre d'endroits où l'on peut trouver de bonnes informations sur le réseau Linux.

Alan Cox, l'actuel mainteneur du code réseau Linux entretient une page web qui contient les points principaux du réseau actuel et les nouveaux développements à l'adresse: *www.uk.linux.org* (`http://www.uk.linux.org/NetNews.html`).

Un autre bon endroit est un livre écrit par Olaf Kirch ayant pour titre `Network Administrators Guide`. C'est une oeuvre du *Linux Documentatation Project* (`http://sunsite.unc.edu/LDP/`) et vous pouvez le lire de manière interactive sur *Network Administrators Guide HTML version* (`http://sunsite.unc.edu/LDP/LDP/nag/nag.html`) ou bien vous pouvez l'obtenir sous différents formats via ftp sur: *sunsite.unc.edu LDP ftp archive* (`ftp://sunsite.unc.edu/pu`

Le livre d'Olaf est très compréhensible et fournit un point de vue de haut niveau sur la configuration réseau sous Linux.

(NdT: ce livre a été traduit en français de manière remarquable par le regretté René Cougnenc)

Il existe un groupe de discussion dédié au réseau et, en ce qui le concerne dans la hiérarchie Linux, c'est: *comp.os.linux.networking* (`news:comp.os.linux.networking`)

Il existe une liste de diffusion à laquelle vous pouvez vous inscrire, et où vous pourrez poser des questions ayant trait au réseau Linux. Pour souscrire vous devez envoyer un message par couurier électronique:

```
To: majordomo@vger.rutgers.edu
Subject: rien du tout
Message:

subscribe linux-net
```

Sur les différents réseaux IRC il y a souvent des canaux `#linux` sur lesquels des personnes sont en mesure de répondre à vos questions concernant le réseau Linux.

Souvenez-vous lorsque vous faites part d'un problème d'y inclure le plus possible de détails nécessaires. Plus spécialement indiquez les versions des logiciels que vous utilisez, en particulier la version du noyau, les versions des outils tels que *pppd* ou *dip*, et la nature exacte des problèmes que vous rencontrez. Cela veut dire prendre note de la syntaxe exacte des messages d'erreurs que vous recevez, et les commandes que vous avez exécutées.

4.3 Où obtenir des informations sur le réseau, non spécifiques de Linux.

si vous désirez des informations générales de base sur tcp/ip, alors je vous recommande de regarder les documents suivants:

introduction à tcp/ip

ce document se trouve à la fois sur *en version texte* (`ftp://athos.rutgers.edu/runet/tcp-ip-intro.doc`) et *en version postscript* (`ftp://athos.rutgers.edu/runet/tcp-ip-intro.ps`).

administration tcp/ip

ce document se trouve à la fois sur *en version texte* (`ftp://athos.rutgers.edu/runet/tcp-ip-admin.doc`) et *en version postscript* (`ftp://athos.rutgers.edu/runet/tcp-ip-admin.ps`).

Si vous recherchez des informations plus détaillées je vous recommande chaudement:

```
"Internetworking with TCP/IP"
par Douglas E. Comer

ISBN 0-13-474321-0
Prentice Hall publications.
```

Si vous voulez apprendre comment écrire des applications réseau dans un environnement compatible Unix, je vous recommande également chaudement:

```
"Unix Network Programming"
par W. Richard Stevens

ISBN 0-13-949876-1
Prentice Hall publications.
```

Vous pouvez essayer aussi le groupe de discussions: *comp.protocols.tcp-ip* (`news:comp.protocols.tcp-ip`).

Une importante source d'informations techniques concernant l'Internet et la suite des protocoles tcp/ip sont les RFC. RFC est l'acronyme de 'Request For Comment' et c'est le moyen habituel de soumettre et de s'informer des normes de protocole Internet. Il y a beauccoup d'endroits où sont stockées ces RFC. Beaucoup de ceux-ci sont des sites ftp, d'autres fournissent des accès WWW avec un moteur de recherche qui cherche les bases de données RFC avec des mots clé particuliers.

Une source possible de RFC est: *Nexor RFC database* (`http://pubweb.nexor.co.uk/public/rfc/index/rfc.html`).

5 Informations générales sur la configuration réseau.

Vous devez connaître et bien comprendre les sous-paragraphes suivants avant d'essayer de configurer votre réseau. Ce sont des principes de base qui s'appliquent, indépendamment de la nature du réseau que vous voulez mettre en place.

5.1 De quoi ai-je besoin pour démarrer?

Avant de commencer à construire ou configurer votre réseau, vous aurez besoin de certaines choses. Les plus importantes sont:

5.1.1 Sources du noyau actuel.

Si le noyau que vous utilisez actuellement ne supporte pas les types de réseau ou les cartes que vous voulez utiliser, vous aurez besoin des sources du noyau pour pouvoir le recompiler avec les options adéquates.

Vous pouvez toujours obtenir les sources du dernier noyau sur: *ftp.funet.fi* (`ftp://ftp.funet.fi/pub/Linux/PEOPLE/Linus/`
(NdT: et bien sûr *ftp.lip6.fr* (`ftp://ftp.lip6.fr/pub/linux/kernel/sources/v2.0`)).

Normalement les sources du noyau doivent être désarchivées dans le répertoire `/usr/src/linux`. Pour savoir comment appliquer les patches et compiler le noyau, lisez le *Kernel-HOWTO* (`Kernel-HOWTO.html`). Pour savoir comment configurer les modules du noyau, lisez le *Module-HOWTO* (`Module-HOWTO.html`).

Sauf indication contraire, je vous recommande de vous en tenir à la version standard du noyau (celle avec un chiffre pair en seconde place dans le numéro de version). Les distributions de développement (avec un chiffre impair en seconde place dans le numéro de version) peuvent avoir une structure ou autre chose qui peut poser problème avec les logiciels de votre système. Si vous n'êtes pas certains de résoudre ce type de problèmes, avec en plus ceux qui existeraient sur d'autres logiciels, ne les utilisez pas.

5.1.2 Outils de réseau actuels.

Ces outils sont les programmes utilisés pour configurer les fichiers de périphériques réseau. Ils vous permettent d'assigner des adresses aux périphériques et de configurer des routes par exemple.

La plupart des distributions Linux modernes sont fournies avec les outils de réseau, aussi si vous avez fait votre installation à partie d'une distribution et que vous n'avez pas encore installé les outils de réseau, vous devez le faire.

Si vous n'avez pas fait l'installation à partir d'une distribution, vous aurez alors besoin des sources pour les compiler vous-mêmes. Ce n'est pas difficile.

Les outils de réseau sont maintenus par Bernd Eckenfels et se trouvent sur: *ftp.inka.de* (`ftp://ftp.inka.de/pub/comp/Linux`, et sont mirrorisés sur: *ftp.linux.uk.org* (`ftp://ftp.uk.linux.org/pub/linux/Networking/base/`).

Soyez sûrs de choisir la version la mieux appropriée à votre noyau et suivez les instructions incluses dans le paquetage.

Pour installer et configurer la version actuelle (au moment où nous écrivons), vous avez besoin de faire :

```
#
# cd /usr/src
# tar xvfz net-tools-1.33.tar.gz
# cd net-tools-1.33
# make config
# make
# make install
#
```

De plus, si vous voulez configurer une protection firewall ou utiliser l'IP masquerade vous aurez besoin de la commande *ipfwadm*. La dernière version peut-être obtenue sur: *ftp.xos.nl* (`ftp:/ftp.xos.nl/pub/linux/ipfwadm`). Encore une fois, de nombreuses versions existent. Soyez sûrs de prendre celle qui s'adapte le mieux à votre noyau.

Pour installer et configurer la version qui a cours en ce moment, vous devrez faire:

```
#
# cd /usr/src
# tar xvfz ipfwadm-2.3.0.tar.gz
# cd ipfwadm-2.3.0
# make
# make install
#
```

5.1.3 Applications réseau.

Les programmes d'application réseau sont des programmes tels que *telnet* et *ftp* et leurs serveurs respectifs. David Holland <`dholland@hcs.harvard.edu`> s'occupe maintenant d'une distribution très répandue. Vous pouvez l'obtenir sur: *ftp.uk.linux.org* (`ftp://ftp.uk.linux.org/pub/linux/Networking/base`).

Pour installer et configurer la version qui existe à l'heure actuelle vous devrez faire:

```
#
# cd /usr/src
# tar xvfz /pub/net/NetKit-B-0.08.tar.gz
# cd NetKit-B-0.08
# more README
# vi MCONFIG
# make
# make install
#
```

5.1.4 Adresses.

Les adresses de protocole Internet (IP) sont composées de quatre octets. La convention d'écriture est appelée 'notation décimale pointée'. Sous cette forme chaque octet est converti en un nombre décimal (0-255), en omettant les zéros de tête (à moins que ce nombre ne soit lui-même un zéro) et chaque octet est séparé par le caractère '.'. Par convention chaque interface d'un hôte ou routeur possède une adresse IP. Il est permis, dans certaines circonstances, que la même adresse IP soit utilisée sur différentes interfaces d'une même machine, mais, en général, chaque interface possède sa propre adresse.

Les réseaux protocole Internet sont des séquences contiguës d'adresses IP. Toutes les adresses d'un même réseau ont des chiffres en commun. La partie d'adresse commune à toutes les adresses d'un réseau s'appelle la 'partie réseau' de l'adresse. Les chiffres restants s'appellent 'partie hôte'. Le nombre de bits qui sont partagés par toutes les adresses d'un même réseau est appelé masque de réseau (netmask) et c'est le rôle du masque de réseau de déterminer quelles adresses appartiennent à 'son' réseau et celles qui ne sont pas concernées. Par exemple:

```
---------------------------------------     ----------------
Adresse hote (host address)                 192.168.110.23
Masque de reseau (network mask)             255.255.255.0
Partie reseau (network portion)             192.168.110.
Partie hote (host portion)                             .23
---------------------------------------     ----------------
Adresse reseau (network address)            192.168.110.0
Adresse de diffusion (broadcast address)    192.168.110.255
---------------------------------------     ----------------
```

Toute adresse qui est 'andée bit à bit' avec son masque de réseau révèlera l'adresse du réseau auquel elle appartient. L'adresse du réseau est par conséquent l'adresse de plus petit nombre dans l'ensemble des adresses et a toujours la partie hôte codée avec des zéros.

L'adresse de diffusion est une adresse spéciale que chaque hôte du réseau écoute en même temps que son adresse personnelle. Cette adresse est celle à laquelle les datagrammes sont envoyés si tous les hôtes du réseau sont en mesure de les recevoir. Certains types de données telles que les informations de routage et

les messages d'alerte sont transmis vers l'adresse de diffusion de telle sorte que tous les hôtes du réseau peuvent les recevoir en même temps. Il y a deux standards utilisés de manière courante pour définir ce que doit être l'adresse de diffusion. Le plus largement utilisé est de prendre l'adresse la plus haute possible du réseau comme adresse de diffusion. Dans l'exemple ci-dessus ce serait `192.168.110.255`. Pour d'autres raisons certains sites ont adopté la convention d'utiliser l'adresse de réseau comme adresse de diffusion. En pratique cela n'a pas beaucoup d'importance, mais vous devez être sûrs que tous les hôtes du réseau sont configurés avec la même adresse de diffusion.

Pour des raisons d'administration, il y a quelque temps, lors du développement du protocole IP, des ensembles d'adresses ont été organisés en réseaux et ces réseaux ont été regroupés en ce que l'on a appellé classes. Ces classes donnent un certain nombre de réseaux de tailles standards auxquels on peut assigner des adresses. Ces classes sont:

Classe de reseau	Masque de reseau	Adresses de reseau
A	255.0.0.0	0.0.0.0 - 127.255.255.255
B	255.255.0.0	128.0.0.0 - 191.255.255.255
C	255.255.255.0	192.0.0.0 - 223.255.255.255
Multicast	240.0.0.0	224.0.0.0 - 239.255.255.255

Le type d'adresse que vous devez utiliser dépend de ce que vous voulez faire exactement. Vous pouvez utiliser une combinaison des actions suivantes pour obtenir l'ensemble des adresses dont vous aurez besoin:

Installer une machine Linux sur un réseau IP existant

Alors vous devez contacter un des administrateurs du réseau et lui demander les informations suivantes:

- Adresse hôte
- Adresse réseau
- Adresse de diffusion
- Masque de réseau
- Adresse de routage
- Adresse du serveur de noms de domaine (DNS)

Vous configurerez alors votre réseau Linux à l'aide de ces données. Vous ne pouvez pas les inventer vous-mêmes et espérer que votre configuration fonctionne.

Construire un réseau tout neuf non connecté à l'Internet

Si vous construisez un réseau privé et que vous n'ayez pas l'intention de vous connecter à l'Internet, vous pouvez alors choisir n'importe quelle adresse. Cependant, pour des raisons de sécurité et de fiablité, il y a quelques adresses de réseau IP réservées à cet usage. Elles sont spécifiées dans la RFC 1597 et sont les suivantes:

ALLOCATIONS POUR RESEAUX PRIVES		
Classe reseau	Masque de reseau	Adresses de reseau
A	255.0.0.0	10.0.0.0 - 10.255.255.255
B	255.255.0.0	172.16.0.0 - 172.31.255.255
C	255.255.255.0	192.168.0.0 - 192.168.255.255

Vous devez d'abord décider de la dimension de votre réseau et choisir ensuite les adresses dont vous avez besoin.

5.2 Où mettre les commandes de configuration ?

Il y a plusieurs possibilités de procédures de démarrage d'un système Linux. Après le démarrage du noyau, celui-ci exécute toujours un programme appelé '*init*'. Ce programme lit le fichier de configuration appelé `/etc/inittab` et commence le processus de démarrage. Il y a quelques variantes de *init* et c'est là que l'on trouve la plus grande différence entre les diverses distributions ou machines.

Habituellement le fichier `/etc/inittab` contient un entrée telle que:

```
si::sysinit:/etc/init.d/boot
```

Cette ligne spécifie le nom du fichier script qui prend en charge réellement la séquence de démarrage. Ce fichier est en quelque sorte équivalent au fichier MS-DOS `AUTOEXEC.BAT`.

Il y a aussi d'autres scripts appelés par le script de démarrage, et souvent le réseau est configuré dans l'un de ceux-ci.

Le tableau suivant peut être utilisé comme guide suivant le système que vous avez:

Distrib.	Interface Config/Routage	Initialisation serveur
Debian	/etc/init.d/network	/etc/init.d/netbase
		/etc/init.d/netstd_init
		/etc/init.d/netstd_nfs
		/etc/init.d/netstd_misc
Slackware	/etc/rc.d/rc.inet1	/etc/rc.d/rc.inet2
RedHat	/etc/sysconfig/network-scripts/ifup-<ifname>	/etc/rc.d/init.d/network

La plupart des distributions récentes incluent un programme qui vous permet de configurer beaucoup de types d'interfaces réseau. Si vous en possédez une, regardez si ce programme vous convient au lieu de tenter une configuration manuelle.

```
----------------------------------------
Distrib   | Programme de configuration reseau
----------------------------------------
RedHat    | /sbin/netcfg
Slackware | /sbin/netconfig
----------------------------------------
```

5.3 Créer vos interfaces réseau.

Sur beaucoup de systèmes Unix les périphériques réseau apparaissent dans le répertoire */dev* . Il n'en est pas de même avec Linux. Les périphériques réseau sont créés dynamiquement dans les logiciels et ne demandent donc pas de fichiers de périphériques.

Dans la majorité des cas les périphériques réseau sont créés automatiquement par le pilote de périphérique pendant son initialisation et lorsqu'il détecte votre matériel. Par exemple le pilote Ethernet crée les interfaces `eth[0..n]` une à une quand il détecte votre matériel Ethernet. La première carte Ethernet trouvée devient `eth0`, la deuxième `eth1` etc.

Cependant dans certains cas, notamment avec *SLIP* et *PPP*, les périphériques réseau sont créés au travers de l'action d'un programme utilisateur. Le même mécanisme séquentiel s'applique sur les périphériques, mais ce n'est pas au moment du démarrage du système. La raison en est que, à l'inverse des dispositifs Ethernet, le nombre de périphériques *SLIP* ou *PPP* actifs peut varier dans le temps. Nous en discuterons plus tard.

5.4 Configurer une interface réseau.

Lorsque vous avez tous les programmes requis, votre adresse et les informations réseau, vous pouvez alors configurer vos interfaces. Lorsque nous parlons de la configuration d'interface, nous faisons allusion au processus d'assignation des adresses du périphérique réseau, et au processus de réglage des paramètres configurables. Le programme le plus utilisé pour ce faire est la commande *ifconfig* (interface configure).

Typiquement vous utilisez une commande comme ci-dessous:

```
# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
```

Dans ce cas je configure l'interface Ethernet '`eth0`' avec l'adresse IP '`192.168.0.1`' et un masque de réseau '`255.255.255.0`'. Le '*up*' qui termine la commande enjoint à l'interface de devenir active.

Le noyau suppose certaines valeurs par défaut lorsque l'on configure les interfaces. Par exemple, vous pouvez indiquer une adresse de réseau et une adresse de diffusion, mais si vous ne le faites pas comme nous venons de le faire dans l'exemple ci-dessus, alors le noyau fera certaines hypothèses basées sur le masque de réseau que vous avez fourni, et si vous ne l'avez pas donnée, sur la classe de l'adresse IP configurée. Dans mon exemple, le noyau assumera que c'est un réseau de classe C et configurera une adresse réseau de '`192.168.0.0`' et une adresse de diffusion de '`192.168.0.255`'.

Il y a de nombreuses autres options pour la commande *ifconfig* . Les plus importantes sont:

up

active une interface.

down

désactive une interface.

(-)arp

active ou désactive le protocole de résolution d'adresses sur cette interface.

(-)allmulti

active ou désactive la réception de tous les paquets multicast matériel (Ndt: Les adresses multicast sont un genre d'adresses de diffusion limitées à un groupe de machine qui n'ont pas nécessairement besoin de se trouver sur le même sous-réseau). Le multicast matériel permet à des groupes d'hôtes de recevoir des paquets adressés vers des destinations spéciales. Ce peut être important si vous utilisez des applications comme la videoconference, mais la plupart du temsp on ne l'utilise pas.

mtu N

ce paramètre permet de régler le *MTU* sur le périphérique.

netmask addr

ce paramètre permet de fixer le masque de réseau.

irq addr

ce paramètre ne marche qu'avec certains types de matériels, mais vous permet d'en fixer l'IRQ.

(-)broadcast *addr*

permet d'activer ou de désactiver l'acceptation de datagrammes destinés à l'adresse de diffusion.

(-)pointopoint *addr*

permet de fixer l'adresse de la machine à l'extrémité d'un lien point-à-point comme pour *SLIP* ou *PPP*.

hw <type> <addr>

permet de fixer l'adresse matérielle de certains périphériques réseau. Ce n'est pas souvent utilisé pour Ethernet, mais utile pour d'autres types de réseau tels que AX.25.

Vous pouvez utiliser la commande *ifconfig* pour toutes les interfaces réseau. Quelques programmes utilisateurs comme *pppd* et *dip* configurent automatiquement les périphériques en même temps qu'ils les créent, dès lors l'utilisation manuelle de *ifconfig* n'est pas nécessaire.

5.5 Configurer votre résolveur de noms.

Le '*Résolveur de Nom*' (Name Resolver) fait partie de la bibliothèque standard de Linux. Sa première fonction est de convertir des noms d'hôtes compréhensibles par l'homme, comme '`ftp.funet.fi`' , en adresses IP compréhensibles par une machine, comme `128.214.248.6`.

5.5.1 Qu'y a-t-il dans un nom ?

Vous êtes probablement familiers avec l'aspect des noms d'hôtes Internet, mais vous ne savez pas comment ils sont composés ou décomposés. Les noms de domaine Internet sont hiérarchisés par nature, c'est à dire qu'ils ont une structure arborescente. Un '*domaine*' est une famille, ou un groupe de noms. Un '*domaine*' peut être subdivisé en '*sous-domaines*'. Un '*domaine de premier niveau*' est un domaine qui n'est pas un sous-domaine. Les Domaines de Premier Niveau sont spécifiés dans la RFC-920. Quelques exemples:

COM

Organisations Commerciales

EDU

Organisations ayant rapport avec l'Education

GOV

Organisations Gouvernementales

MIL

Organisations Militaires

ORG

Autres organisations

NET

Organisations ayant un rapport avec l'internet

Nom de Pays

il existe des codes de deux lettres qui représentent un pays donné.

Chacun de ces domaines de premier niveau possède des sous-domaines. Les domaines de premier niveau basés sur les noms de pays sont divisés ensuite en sous-domaines basés sur les domaines `com`, `edu`, `gov`, `mil` et `org` . Ainsi par exemple, vous finissez par : `com.au` and `gov.au` pour des organisations commerciales ou gouvernementales situées en Australie. Pour des raisons historiques la plupart des domaines appartenant à des domaines qui ne sont pas basés sur des noms de pays sont pour les organisations appartenant aux Etats-Unis, bien que les Etats-Unis aient aussi le code de pays '`.us`'.

Le niveau de division suivant représente habituellement le nom de l'organisation. Ces sous-domaines sont variables, souvent ils sont basés sur la structure en départements de l'organisation mais ils peuvent être basés également sur d'autres critères considérés comme rationnels et compréhensibles par les administrateurs réseau de l'organisation.

La partie tout à fait à gauche du nom est toujours l'unique nom assigné à la machine hôte et est appelée le nom d'hôte '*hostname*', la partie de droite du nom est le nom de domaine '*domainname*' et le nom complet s'appelle le nom de domaine complètement qualifié '*Fully Qualified Domain Name*'.

Si l'on examine mon adresse électronique par exemple, le nom pleinement qualifié est '`perf.no.itg.telstra.com.au`'. Cela veut dire que le nom d'hôte est '`perf`' et le nom de domaine '`no.itg.telstra.com.au`'. Le nom de

domaine est basé sur un domaine de premier niveau basé sur mon pays, l'Australie et comme mon adresse électronique appartient à une organisation commerciale nous avons '`.com`' comme domaine de niveau adjacent. Le nom de la société est (était) '`telstra`' et notre structure interne de noms est basé sur la structure organisationnelle, dans mon cas, ma machine appartient à l'Information Technology Group, Network Operations section.

5.5.2 Quelles sont les informations nécessaires.

Vous devez connaître le domaine auquel votre nom d'hôte appartient. Le résolveur de nom effectue la traduction en faisant appel à un '*Serveur de Nom de Domaine*', aussi vous devez connaître l'adresse IP d'un serveur de nom local que vous pouvez utiliser.

Il y a trois fichiers que vous devez éditer, nous en parlerons chacun à leur tour.

5.5.3 /etc/resolv.conf

Le fichier `/etc/resolv.conf` est le principal fichier de configuration pour le code de résolution de nom. Son format est tràs simple. C'est un fichier texte avec um mot-clé par ligne. Il y a trois mots-clés typiquement utilisés, qui sont:

domain

ce mot-clé indique le nom de domaine local.

search

ce mot-clé spécifie une liste d'autres noms de domaine pour rechercher un nom d'hôte.

nameserver

ce mot-clé, qui peut être utilisé plusieurs fois, spécifie l'adresse IP d'un serveur de nom de domaine pour la résolution de noms.

Un exemple de `/etc/resolv.conf` pourrait ressembler à ceci:

```
domain maths.wu.edu.au
search maths.wu.edu.au wu.edu.au
nameserver 192.168.10.1
nameserver 192.168.12.1
```

Cet exemple spécifie que le nom de domaine par défaut à ajouter aux noms non qualifiés (c'est à dire sans domaine) est `maths.wu.edu.au`, et que si l'hôte n'est pas trouvé dans ce domaine on peut aussi essayer le domaine `wu.edu.au` directement. Deux entrées de serveurs de noms sont fournies, chacune d'elles pouvant être appelée par le résolveur de noms.

5.5.4 /etc/host.conf

Le fichier `/etc/host.conf` sert à configurer certaines choses en vue de modifier le comportement du résolveur de noms. Son format est décrit en détail dans la page de manuel '`resolv+`'. Dans le plupart des cas l'exemple suivant vous conviendra:

```
order hosts,bind
multi on
```

Cette configuration indique au résolveur de nom de vérifier en premier lieu le fichier `/etc/hosts` avant d'essayer un serveur de noms et de renvoyer toutes les adresses valables d'un hôte trouvé dans le fichier `/etc/hosts` au lieu d'en donner simplement la première.

5.5.5 /etc/hosts

Le fichier `/etc/hosts` est l'endroit où vous mettez les noms et les adresses IP des hôtes locaux. Si vous mettez un hôte dans ce fichier, alors vous n'avez pas à interroger le serveur de nom de domaine pour obtenir son adresse IP. L'inconvénient est que vous devez tenir votre fichier à jour si l'adresse de cet hôte a changé. Dans un système bien administré les seuls noms d'hôtes qui apparaissent habituellement sont l'interface loopback, et le nom des hôtes locaux.

```
# /etc/hosts
127.0.0.1       localhost loopback
192.168.0.1     this.host.name
```

Vous pouvez spécifier plus d'un nom d'hôte comme montré dans la première entrée qui est standard pour l'interface loopback.

5.6 Configurer votre interface loopback.

L'interface '`loopback`' est un type spécial d'interface qui permet de vous connecter à vous-même. Il y a plusieurs raisons pour faire ceci, par exemple si vous voulez faire des essais de logiciel réseau sans interférer avec quelqu'un d'autre sur votre réseau. Par convention, l'adresse IP '`127.0.0.1`' lui a été assignée. Aussi quelle que soit la machine où vous êtes, si vous ouvrez une connexion telnet vers `127.0.0.1` vous atteindrez toujours l'hôte local.

Configurer l'interface loopback est simple et vous devez vous assurer de l'avoir fait.

```
# ifconfig lo 127.0.0.1
# route add -host 127.0.0.1 lo
```

Nous en dirons plus sur la commande *route* dans le prochain paragraphe.

5.7 Routage.

Le routage est un vaste sujet. On peut écrire de grandes quantités de textes sur ce sujet. La plupart d'entre vous ont besoin d'un simple routage, et certains même de rien du tout. Je ne parlerai que des principes du routage. Si vous voulez plus d'informations je vous suggère de vous reporter aux références fournies en début du document.

Commençons par une définition. Qu'est-ce que le routage IP ? Voici celle que j'utilise:

> Le routage IP est le processus par lequel un hôte, ayant des connexions réseau multiples, décide du chemin où délivrer les datagrammes IP qu'il a reçus.

Il peut être utile d'illustrer cela par un exemple. Imaginez un routeur dans un bureau: il peut avoir un lien PPP sur l'Internet, un certain nombre de segments Ethernet alimentant les stations de travail et un second lien PPP vers un autre bureau. Lorsque le routeur reçoit un datagramme de l'une de ses connexions, le routage est le mécanisme utilisé pour déterminer vers quelle interface il doit renvoyer ce datagramme. De simples hôtes ont besoin aussi de routage, tous les hôtes Internet ayant deux périphériques réseau, l'un étant l'interface loopback décrite auparavant et l'autre est celui qui est utilisé pour parler avec le reste du monde, soit un lien Ethernet, soit une interface série PPP ou SLIP.

Ok, alors comment marche le routage? Chaque hôte possède une liste spéciale de règles de routage, appelée une table de routage. Cette table contient des colonnes qui contiennent au moins trois champs, le premier étant une adresse de destination, le deuxième étant le nom de l'interface vers lequel le datagramme doit être routé et le troisième, qui est optionnel, l'adresse IP d'une autre machine qui transportera le datagramme vers sa prochaine destination sur le réseau. Sur Linux vous pouvez voir cette table en utilisant la commande suivante:

```
# cat /proc/net/route
```

ou bien en utilsant l'une des commandes suivantes:

```
# /sbin/route -n
# /sbin/netstat -r
```

Le processus de routage est plutôt simple: une datagramme entrant est reçu, l'adresse de destination est examinée et comparée avec chaque entrée de la table. L'entrée qui correspond le mieux à cette adresse est choisie, et le datagramme est renvoyé vers l'interface spécifiée. Si le champ passerelle est rempli, alors le datagramme est renvoyé vers cet hôte via l'interface spécifiée, sauf indication contraire l'adresse de destination est supposée comme étant sur le réseau supporté par l'interface.

Pour manipuler ce tableau, une commande spéciale est utilisée. Cette commande prend des arguments et les convertit en appels du système noyau qui lui demande d'ajouter, de supprimer ou de modifier des entrées dans la table de routage. Cette commande s'appelle '*route*'.

Un simple exemple. Imaginez que vous ayez un réseau Ethernet. On vous a dit que c'est un réseau classe C avec une adresse de `192.168.1.0`. On vous fournit une adresse IP `192.168.1.10` pour votre usage et on vous a dit que `192.168.1.1` est un routeur connecté à l'Internet.

La première étape est de configurer l'interface comme indiqué plus haut. Vous utiliserez la commande:

```
# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
```

Maintenant vous avez besoin d'ajouter une entrée dans la table de routage pour indiquer au noyau que les datagrammes destinés aux hôtes dont les adresses correspondent à `192.168.1.*` doivent être dirigés vers le périphérique Ethernet. Vous utiliserez une commande comme ceci:

```
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

Notez l'utilisation de l'argument '`-net`' pour indiquer au programme route que cette entrée est une route réseau. Un autre choix peut être '`-host`' qui est une route spécifique d'une adresse IP.

Cette route vous permettra d'établir des connexions IP avec tous les hôtes sur votre segment Ethernet. Mais qu'en est-il des hôtes IP qui n'y sont pas?

Ce serait compliqué d'ajouter des routes pour chaque réseau destinataire, aussi il y a une astuce utilisée pour simplifier la tâche. L'astuce est appelée route par '`default`'. La route par `default` s'adapte à toutes les destinations possibles, mais pas très bien, de telle sorte que si il y a une entrée qui correspond à l'adresse requise elle sera utilisée à la place de la route par `default`. L'idée de la route par `default` est simplement de pouvoir dire 'et tout le reste va ici'. Dans l'exemple que j'ai inventé, on utilisera une entrée telle que:

```
# route add default gw 192.168.1.1 eth0
```

L'argument '`gw`' indique à la commande route que le prochain argument est l'adresse IP, ou le nom, d'une passerelle (gateway) ou d'une machine routeur vers qui tous les datagrammes correspondant à cette entrée seront dirigés pour routage ultérieur.

Ainsi votre configuration complète sera:

```
# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
# route add default gw 192.168.1.1 eth0
```

Si vous regardez bien vos fichiers réseau '`rc`' vous en trouverez au moins un très semblable à celui-ci. C'est une configuration courante.

Examinons maintenant une configuration un peu plus compliquée. Imaginons que nous configurions le routeur examiné auparavant, celui qui avait un lien PPP vers l'Internet et des segments LAN alimentant des stations de travail dans le bureau. Supposons que ce routeur ait 3 segments Ethernet et un lien PPP. Notre configuration de routage ressemblerait à ceci:

```
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
# route add -net 192.168.2.0 netmask 255.255.255.0 eth1
# route add -net 192.168.3.0 netmask 255.255.255.0 eth2
# route add default ppp0
```

Chacune des stations de travail utilisera le format plus simple décrit ci-dessus, seul le routeur aura besoin d'indiquer les routes réseau séparément car pour les stations de travail le mécanisme de routage par `default`

les capturera toutes, laissant au routeur le soin de les séparer de manière appropriée. Vous pouvez vous demander pourquoi la route par défaut n'utilise pas '`gw`'. La raison en est très simple: les protocoles de lien série comme PPP et SLIP ont seulement deux hôtes sur leur réseau, un à chaque bout. Spécifier à l'hôte que l'autre bout de la liaison est une passerelle est sans objet et redondant, car il n'a pas d'autre choix, aussi vous n'avez pas à indiquer une passerelle pour ce type de connexions réseau. Les autres types comme Ethernet, arcnet ou token ring ont besoin que l'on indique une passerelle car ces réseaux supportent un grand nombre d'hôtes.

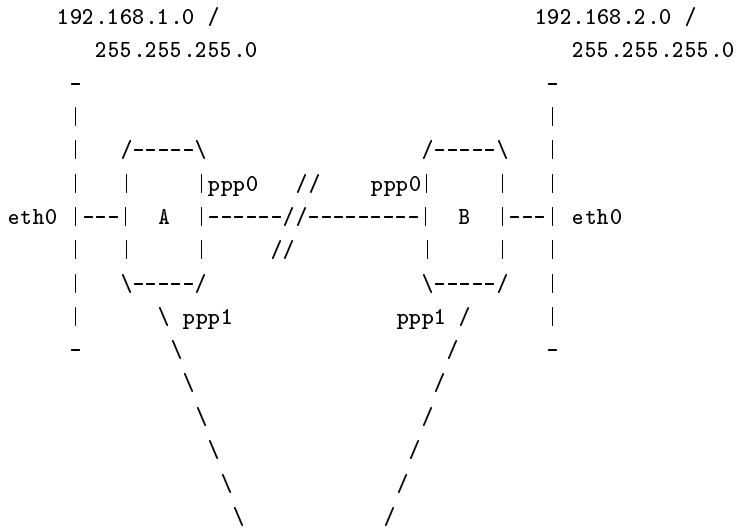
5.7.1 Alors, que fait le programme *routed* ?

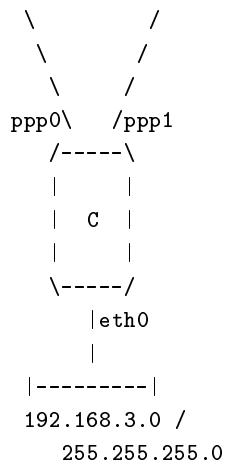
La configuration de routage décrite ci-dessus est bien adaptée aux réseaux simples où il n'y a que des chemins uniques entre les destinations. Lorsque vous avez un réseau plus complexe les choses deviennent plus compliquées. Heureusement pour la plupart d'entre vous, ce ne sera pas le cas.

Le gros problème est qu'avec le 'routage manuel' ou 'routage statique' comme décrit ci-dessus, si une machine ou un lien tombe en panne dans le réseau, alors la seule façon de diriger vos datagrammes vers un autre chemin, s'il existe, est d'intervenir manuellement et d'exécuter les commandes adéquates. Naturellement c'est lourd, lent, peu pratique et source de risques. Des techniques variées ont été développées pour régler automatiquement les tables de routage dans le cas d'incidents sur un réseau où il y a plusieurs routes possibles, toutes ces techniques étant regroupées sous le nom de 'protocoles de routage dynamique'.

Vous avez peut-être entendu parler des plus courants. Ce sont RIP (Routing Information Protocol) et OSPF (Open Shortest Path First Protocol). RIP est très souvent utilisé sur les petits ou moyens réseaux d'entreprise. L'OPSF est plus moderne et plus apte à gérer de grands réseaux et mieux adapté dans le cas où il y a un grand nombre de chemins possibles à travers le réseau. Les implémentations usuelles de ces protocoles sont: '*routed*' - RIP, et '*gated*' - RIP, OSPF et autres. Le programme '*routed*' est normalement fourni avec votre distribution Linux ou est inclus dans la paquetage 'NetKit' décrit auparavant.

Un exemple pour vous montrer comment et où vous pouvez utiliser un protocole de routage dynamique ressemblerait à ceci:





Nous avons trois routeurs A, B et C. Chacun supporte un segment Ethernet avec un réseau IP de classe C (masque de réseau 255.255.255.0). Chaque routeur a également une liaison PPP vers chacun des autres routeurs. Ce réseau forme un triangle.

Il est évident que la table de routage sur le routeur A ressemble à ceci:

```
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
# route add -net 192.168.2.0 netmask 255.255.255.0 ppp0
# route add -net 192.168.3.0 netmask 255.255.255.0 ppp1
```

Cela fonctionnera bien jusqu'à ce que le lien entre A et B tombe en panne. Si cette liaison est défaillante, alors l'entrée de routage montre que les hôtes sur le segment A ne peuvent pas atteindre les hôtes sur le segment B car leurs datagrammes seront dirigés sur le lien ppp0 du routeur A qui est rompu. Ils pourront encore continuer à parler aux hôtes du segment C, et les hôtes du segment C pourront toujours parler à ceux du segment B car la liaison reste intacte.

Mais.., si A peut parler à C et si C peut toujours parler à B, pourquoi A ne routerait-il pas ses datagrammes pour B via C, et laisser ensuite C les envoyer à B? C'est exactement le type de problèmes que les protocoles de routage dynamique comme RIP sont en mesure de résoudre. Si chacun des routeurs A, B et C utilisent un démon de routage alors leurs tables de routage seront automatiquement réglées pour refléter le nouvel état du réseau mêma si l'une des liaisons est défectueuse. Configurer un tel réseau est simple, sur chaque routeur vous devez seulement faire deux choses. Dans ce cas pour le routeur A:

```
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
# /usr/sbin/routed
```

Le démon de routage '*routed*' trouve automatiquement tous les ports actifs vers le réseau quand il démarre et écoute tous les messages sur chacun des périphériques réseau ce qui lui permet de déterminer et de mettre à jour sa table de routage.

C'était une très brève explication du routage dynamique et de son utilisation. Si vous voulez d'avantage d'explications reportez aux références listées en début de document.

Les points importants relatifs au routage dynamique sont:

1. Vous n'avez besoin d'utiliser un démon de routage dynamique que quand votre machine Linux peut choisir entre plusieurs routes pour une destination donnée.
2. Le démon de routage dynamique modifiera automatiquement votre table de routage pour tenir compte des changements survenus dans votre réseau.
3. RIP est adapté aux réseaux de petite et moyenne taille.

5.8 Configurer vos serveurs réseau et les services.

Les serveurs de réseau et les services sont des programmes qui permettent à un utilisateur distant de devenir utilisateur de votre machine Linux. Les programmes serveur sont à l'écoute des ports réseau. Les ports réseau permettent d'adresser un service particulier à un hôte particulier et de faire la différence entre une connexion telnet entrante et une connexion ftp entrante. L'utilisateur distant établit une connexion réseau avec votre machine puis le programme serveur, ou démon de réseau, à l'écoute du port, accepte la connexion et s'exécute. Il y a deux façons d'opérer pour les démons de réseau. Les deux sont couramment utilisés en pratique. Ce sont:

autonome

le programme démon écoute le port réseau désigné et lorsqu'il y a une connexion, il prend lui-même la connexion en charge pour fournir le service.

esclave du serveur *inetd*

le serveur *inetd* est un programme démon spécial spécialisé dans la conduite des connexions réseau. Il possède un fichier de configuration qui indique quel programme doit être utilisé lorsqu'une connexion entrante est reçue. Chacun des ports service doit être configuré soit avec le protocole tcp, soit avec le protocole udp. Les ports sont décrits dans un autre fichier dont nous parlerons plus tard.

Il y deux fichiers importants que vous devez configurer. Ce sont `/etc/services` qui assigne des noms aux numéros de port et `/etc/inetd.conf` qui sert pour la configuration du démon de réseau *inetd* .

5.8.1 `/etc/services`

Le fichier `/etc/services` est une simple base de données qui associe des noms compréhensibles par l'homme à des ports service compréhensibles par la machine. Son format est tout à fait simple. Le fichier est un fichier texte dont chaque ligne représente une entrée de la base de données. Chaque entrée comprend trois champs séparés par des caractères espace ou tabulation. Ces champs sont:

```
nom      port/protocole        alias     # commentaire
```

nom

un simple mot qui représente le service décrit.

port/protocole

ce champ est divisé en deux.

port

un nombre qui spécifie le numéro de port où le service désigné sera disponible. La plupart des services ont des numéros assignés. Ils sont décrits dans la `RFC-1340`.

protocole

c'est soit `tcp` soit `udp`.

Il est important de noter qu'une entrée comme `18/tcp` est très différente de `18/udp` et qu'il n'y a pas de raisons techniques que le même service existe sur les deux. Normalement le bon sens prévaut et c'est vraiment pour un service particulier disponible à la fois sur `tcp` et `udp` que vous verrez une entrée pour les deux..

alias

autre nom qui peut être utilisé pour désigner ce service.

Tout texte apparaissant après le caractère '`#`' est ignoré et traité comme commentaire.

Exemple de fichier `/etc/services`. Toutes les distributions récentes de Linux fournissent un bon fichier `/etc/services`. Juste au cas où vous construiriez tout depuis le départ, voici une copie du fichier `/etc/services` fourni avec la distribution *Debian* (`http://www.debian.org/`) .

```
# /etc/services:
# $Id: services,v 1.3 1996/05/06 21:42:37 tobias Exp $
#
# Network services, Internet style
#
# Notez que c'est la politique actuelle de l'IANA d'assigner un seul numero
# de port a la fois pour TCP et UDP; ainsi, la plupart des ports ont deux
#entrees meme si le protocole ne supporte pas UDP.
# Mis a jour d'apres la RFC 1340, ``Assigned Numbers'' (Juillet 1992).
# Il n'y a pas tous les ports, seulement les plus courants.

tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message send protocol
msp             18/udp                          # message send protocol
```

```
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp                          # SSH Remote Login Protocol
telnet          23/tcp
# 24 - private
smtp            25/tcp          mail
# 26 - non assigne
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/udp          resource        # resource location
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
re-mail-ck      50/tcp                          # Remote Mail Checking Protocol
re-mail-ck      50/udp                          # Remote Mail Checking Protocol
domain          53/tcp          nameserver      # name-domain server
domain          53/udp          nameserver
mtp             57/tcp                          # deprecated
bootps          67/tcp                          # BOOTP server
bootps          67/udp
bootpc          68/tcp                          # BOOTP client
bootpc          68/udp
tftp            69/udp
gopher          70/tcp                          # Internet Gopher
gopher          70/udp
rje             77/tcp          netrjs
finger          79/tcp
www             80/tcp          http            # WorldWideWeb HTTP
www             80/udp                          # HyperText Transfer Protocol
link            87/tcp          ttylink
kerberos        88/tcp          kerberos5 krb5  # Kerberos v5
kerberos        88/udp          kerberos5 krb5  # Kerberos v5
supdup          95/tcp
# 100 - reserve
hostnames       101/tcp         hostname        # usually from sri-nic
iso-tsap        102/tcp         tsap            # part of ISODE.
csnet-ns        105/tcp         cso-ns          # also used by CSO name server
csnet-ns        105/udp         cso-ns
rtelnet         107/tcp                         # Remote Telnet
rtelnet         107/udp
pop-2           109/tcp         postoffice      # POP version 2
pop-2           109/udp
pop-3           110/tcp                         # POP version 3
pop-3           110/udp
sunrpc          111/tcp         portmapper      # RPC 4.0 portmapper TCP
sunrpc          111/udp         portmapper      # RPC 4.0 portmapper UDP
```

```
auth            113/tcp         authentication tap ident
sftp            115/tcp
uucp-path       117/tcp
nntp            119/tcp         readnews untp   # USENET News Transfer Protocol
ntp             123/tcp
ntp             123/udp                         # Network Time Protocol
netbios-ns      137/tcp                         # NETBIOS Name Service
netbios-ns      137/udp
netbios-dgm     138/tcp                         # NETBIOS Datagram Service
netbios-dgm     138/udp
netbios-ssn     139/tcp                         # NETBIOS session service
netbios-ssn     139/udp
imap2           143/tcp                         # Interim Mail Access Proto v2
imap2           143/udp
snmp            161/udp                         # Simple Net Mgmt Proto
snmp-trap       162/udp         snmptrap        # Traps for SNMP
cmip-man        163/tcp                         # ISO mgmt over IP (CMOT)
cmip-man        163/udp
cmip-agent      164/tcp
cmip-agent      164/udp
xdmcp           177/tcp                         # X Display Mgr. Control Proto
xdmcp           177/udp
nextstep        178/tcp         NeXTStep NextStep       # NeXTStep window
nextstep        178/udp         NeXTStep NextStep       # server
bgp             179/tcp                         # Border Gateway Proto.
bgp             179/udp
prospero        191/tcp                         # Cliff Neuman's Prospero
prospero        191/udp
irc             194/tcp                         # Internet Relay Chat
irc             194/udp
smux            199/tcp                         # SNMP Unix Multiplexer
smux            199/udp
at-rtmp         201/tcp                         # AppleTalk routing
at-rtmp         201/udp
at-nbp          202/tcp                         # AppleTalk name binding
at-nbp          202/udp
at-echo         204/tcp                         # AppleTalk echo
at-echo         204/udp
at-zis          206/tcp                         # AppleTalk zone information
at-zis          206/udp
z3950           210/tcp         wais            # NISO Z39.50 database
z3950           210/udp         wais
ipx             213/tcp                         # IPX
ipx             213/udp
imap3           220/tcp                         # Interactive Mail Access
imap3           220/udp                         # Protocol v3
ulistserv       372/tcp                         # UNIX Listserv
ulistserv       372/udp
```

```
#
# services specifiques a UNIX
#
exec            512/tcp
biff            512/udp         comsat
login           513/tcp
who             513/udp         whod
shell           514/tcp         cmd             # no passwords used
syslog          514/udp
printer         515/tcp         spooler         # line printer spooler
talk            517/udp
ntalk           518/udp
route           520/udp         router routed   # RIP
timed           525/udp         timeserver
tempo           526/tcp         newdate
courier         530/tcp         rpc
conference      531/tcp         chat
netnews         532/tcp         readnews
netwall         533/udp                         # -for emergency broadcasts
uucp            540/tcp         uucpd           # uucp daemon
remotefs        556/tcp         rfs_server rfs  # Brunhoff remote filesystem
klogin          543/tcp                         # Kerberized `rlogin' (v5)
kshell          544/tcp         krcmd           # Kerberized `rsh' (v5)
kerberos-adm    749/tcp                         # Kerberos `kadmin' (v5)
#
webster         765/tcp                         # Network dictionary
webster         765/udp
#
# D'apres ``Assigned Numbers'':
#
#> Les Ports Enregistres ne sont pas controles par l'IANA et peuvent etre
#> utilises sur la plupart des systemes par des processus ordinaires
#> ou des programmes executes par des utilisateurs ordinaires.
#
#> Les ports sont utilises dans le TCP [45,106] pour nommer les extremites
#> des connexions logiques qui transportent les conversations de longue
#> duree. Pour offrir des services a des utilisateurs non connus, un port
#> de service pour contact a ete defini. Cette liste specifie le port utilise
#> par le processus serveur ainsi que son port de contact. Comme l'IANA ne peut
#> controler l'usage de ces ports, on donne ici une liste d'utilisation
#> de ces ports pour etre agreable a la communaute.
#
ingreslock      1524/tcp
ingreslock      1524/udp
prospero-np     1525/tcp                # Prospero non-privileged
prospero-np     1525/udp
rfe             5002/tcp                # Radio Free Ethernet
rfe             5002/udp                # Actually uses UDP only
```

```
bbs             7000/tcp                # BBS service
#
#
# services Kerberos (Project Athena/MIT)
# Notez que ceux-ci sont pour Kerberos v4, et ne sont pas officiels. Les sites
# tournant sous v4 doivent utiliser ceux-ci et annuler les entrees v5 ci-dessus.
#
kerberos4       750/udp         kdc     # Kerberos (server) udp
kerberos4       750/tcp         kdc     # Kerberos (server) tcp
kerberos_master 751/udp                 # Kerberos authentication
kerberos_master 751/tcp                 # Kerberos authentication
passwd_server   752/udp                 # Kerberos passwd server
krb_prop        754/tcp                 # Kerberos slave propagation
krbupdate       760/tcp         kreg    # Kerberos registration
kpasswd         761/tcp         kpwd    # Kerberos "passwd"
kpop            1109/tcp                # Pop with Kerberos
knetd           2053/tcp                # Kerberos de-multiplexor
zephyr-srv      2102/udp                # Zephyr server
zephyr-clt      2103/udp                # Zephyr serv-hm connection
zephyr-hm       2104/udp                # Zephyr hostmanager
eklogin         2105/tcp                # Kerberos encrypted rlogin
#
# Services non officiels mais necessaires (pour NetBSD)
#
supfilesrv      871/tcp                 # SUP server
supfiledbg      1127/tcp                # SUP debugging
#
# Services protocole de delivrance de datagrammes
#
rtmp            1/ddp                   # Routing Table Maintenance Protocol
nbp             2/ddp                   # Name Binding Protocol
echo            4/ddp                   # AppleTalk Echo Protocol
zip             6/ddp                   # Zone Information Protocol
#
# Services Debian GNU/Linux
rmtcfg          1236/tcp                # Gracilis Packeten remote config server
xtel            1313/tcp                # french minitel
cfinger         2003/tcp                # GNU Finger
postgres        4321/tcp                # POSTGRES
mandelspawn     9359/udp        mandelbrot      # network mandelbrot

# Services locaux
```

5.8.2 /etc/inetd.conf

Le fichier `/etc/inetd.conf` est le fichier de configuration du serveur démon *inetd* . Il sert à dire à *inetd* ce qu'il doit faire lorsqu'il reçoit une demande de connexion pour un service particulier. Pour les services où

vous acceptez une connexion vous devez dire à *inetd* quel démon serveur de réseau doit tourner, et comment.

Son format est aussi très simple. C'est un fichier texte dont chaque ligne décrit un service que vous voulez fournir. Tout texte suivant un '`#`' est ignoré et considéré comme commentaire. Chaque ligne contient sept champs séparés par un nombre quelconque d'espaces (espace ou tabulation). Le format général est comme suit:

```
service  type_de_socket  proto  drapeaux  utilisateur  chemin  arguments
```

service

est le service applicable à cette configuration, pris dans le fichier `/etc/services`.

type_de_socket

ce champ décrit le type de socket que cette entrée considère comme pertinent, les valeurs permises sont: `stream`, `dgram`, `raw`, `rdm`, ou `seqpacket`. C'est un peu technique par nature, mais par expérience, presque tous les services basés sur `tcp` utilisent `stream` et presque tous les services basés sur `udp` utilisent `dgram`. Il n'y a que quelques types de serveurs démons spéciaux qui utilisent d'autres valeurs.

protocole

le protocole considéré comme valide pour cette entrée. Il doit correspondre à l'entrée appropriée dans le fichier `/etc/services` et sera donc soit `tcp` soit `udp`. Les serveurs basés sur Sun RPC (Remote Procedure Call) utilisent `rpc/tcp` ou `rpc/udp`.

drapeaux

il n'y a en fait que deux valeurs pour ce champ. Celles-ci disent à *inetd* si le programme serveur réseau libère le socket aprés démarrage, et donc si *inetd* peut prendre en compte une des prochaines demandes de connexion, ou bien si *inetd* doit attendre qu'un autre démon serveur tournant déjà prenne en charge la nouvelle demande de connexion. C'est encore compliqué, mais en pratique tous les serveurs `tcp` doivent avoir cette entrée positionnée sur `nowait` et la plupart des serveurs `udp` ont cette entrée positionnée sur `wait`. Attention il y a quelques exceptions notables, laissez vous guider par l'exemple suivant si vous n'êtes pas sûrs.

utilisateur

ce champ décrit quel compte utilisateur extrait de `/etc/passwd` sera considéré comme propriétaire du démon réseau lorsqu'il est lancé. C'est très utile lorsque vous voulez vous protéger contre les trous de sécurité. Vous pouvez mettre `nobody` comme utilisateur pour une entrée si bien que dans le cas où le réseau comporte une brèche, les dommages éventuels seront minimisés. Cependant habituellement ce champ est réglé sur `root`, car beaucoup de serveurs ont besoin des privilèges de root pour tourner correctement.

chemin_de_serveur

ce champ est le chemin réel du programme à exécuter pour cette entrée.

arguments

ce champ correspond au reste de la ligne et est optionnel. Il sert à indiquer les arguments de commande que vous voulez passer au programme serveur au lancement.

Exemple de fichier `/etc/inetd.conf` Comme pour le fichier `/etc/services` toutes les distributions modernes incluent un bon fichier `/etc/inetd.conf` pour pouvoir travailler. Ici, pour être complet , vous trouverez le fichier `/etc/inetd.conf` de la distribution *Debian* (`http://www.debian.org/`).

```
# /etc/inetd.conf:  voir inetd(8) pour d'autres informations.
#
# Base de donnees pour la configuration d'un serveur Internet
#
#
# Modifie pour Debian par Peter Tobias <tobias@et-inf.fho-emden.de>
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
# Services internes
#
#echo           stream  tcp     nowait  root    internal
#echo           dgram   udp     wait    root    internal
discard         stream  tcp     nowait  root    internal
discard         dgram   udp     wait    root    internal
daytime         stream  tcp     nowait  root    internal
daytime         dgram   udp     wait    root    internal
#chargen        stream  tcp     nowait  root    internal
#chargen        dgram   udp     wait    root    internal
time            stream  tcp     nowait  root    internal
time            dgram   udp     wait    root    internal
#
# Services standards.
#
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.telnetd
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.ftpd
#fsp    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.fspd
#
# Shell, login, exec et talk sont des protocoles BSD.
#
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd
login   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind
#exec   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rexecd
talk    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.talkd
ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.ntalkd
#
# Services Mail, news et uucp.
#
smtp    stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.smtpd
#nntp   stream  tcp     nowait  news    /usr/sbin/tcpd  /usr/sbin/in.nntpd
#uucp   stream  tcp     nowait  uucp    /usr/sbin/tcpd  /usr/lib/uucp/uucico
#comsat dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.comsat
#
# Pop et autres
```

```
#
#pop-2  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop2d
#pop-3  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop3d
#
# 'cfinger' est le serveur finger GNU de Debian.  (NOTE: L'implementation
# habituelle du demon 'finger' permet de le faire tourner avec 'root'.)
#
#cfinger stream tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.cfingerd
#finger stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.fingerd
#netstat        stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/netstat
#systat stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/ps -auwwx
#
# Le service Tftp est fourni principalement pour demarrer. La plupart des sites
# l'utilise seulement sur les machines servant de 'serveurs de boot'.
#
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd /boot
#bootps dgram   udp     wait    root    /usr/sbin/bootpd        bootpd -i -t 120
#
# Services Kerberos (ils doivent probablement etre corriges)
#
#klogin         stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind -k
#eklogin        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind -k -x
#kshell         stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd -k
#
# Services tournant UNIQUEMENT sur Kerberos (doivent etre probablement corriges)
#
#krbupdate      stream tcp      nowait  root    /usr/sbin/tcpd  /usr/sbin/registerd
#kpasswd        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/kpasswdd
#
# Services RPC
#
#mountd/1       dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.mountd
#rstatd/1-3     dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rstatd
#rusersd/2-3    dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rusersd
#walld/1        dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rwalld
#
# Fin de inetd.conf.
ident           stream  tcp     nowait  nobody  /usr/sbin/identd       identd -i
```

5.9 Autres fichiers de configuration ayant un rapport avec le réseau.

Il y a beaucoup de fichiers relatifs à la configuration réseau sous Linux susceptibles de vous intéresser. Vous n'aurez jamais à modifier ces fichiers, mais il est utile de les décrire pour que vous sachiez ce qu'ils contiennent et quelle est leur utilité.

5.9.1 /etc/protocols

Le fichier `/etc/protocols` est une base de données qui donne la relation des numéros id de protocole avec leurs noms. Il est utilisé par les programmeurs pour leur permettre de spécifier les protocoles par leur nom dans les programmes et aussi par quelques programmes tels que *tcpdump* pour pouvoir afficher en sortie des noms au lieu de chiffres. La syntaxe générale de ce fichier est:

```
nom du protocole     numero    alias
```

Le fichier `/etc/protocols` fourni avec la distribution *Debian* (`http://www.debian.org/`) est le suivant:

```
# /etc/protocols:
# $Id: protocols,v 1.1 1995/02/24 01:09:41 imurdock Exp $
#
# Protocoles Internet (IP)
#
#       d'apres: @(#)protocols  5.1 (Berkeley) 4/17/89
#
# Mise a jour pour NetBSD basee sur la RFC 1340, Assigned Numbers (July 1992).

ip      0       IP              # internet protocol, pseudo protocol number
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # Internet Group Management
ggp     3       GGP             # gateway-gateway protocol
ipencap 4       IP-ENCAP        # IP encapsulated in IP (officially ``IP'')
st      5       ST              # ST datagram mode
tcp     6       TCP             # transmission control protocol
egp     8       EGP             # exterior gateway protocol
pup     12      PUP             # PARC universal packet protocol
udp     17      UDP             # user datagram protocol
hmp     20      HMP             # host monitoring protocol
xns-idp 22      XNS-IDP         # Xerox NS IDP
rdp     27      RDP             # "reliable datagram" protocol
iso-tp4 29      ISO-TP4         # ISO Transport Protocol class 4
xtp     36      XTP             # Xpress Tranfer Protocol
ddp     37      DDP             # Datagram Delivery Protocol
idpr-cmtp       39      IDPR-CMTP       # IDPR Control Message Transport
rspf    73      RSPF            # Radio Shortest Path First.
vmtp    81      VMTP            # Versatile Message Transport
ospf    89      OSPFIGP         # Open Shortest Path First IGP
ipip    94      IPIP            # Yet Another IP encapsulation
encap   98      ENCAP           # Yet Another IP encapsulation
```

5.9.2 /etc/networks

Le fichier `/etc/networks` a une fonction similaire au fichier `/etc/hosts`. Il fournit une simple base de données de noms de réseau avec des adresses. Son format diffère en ce qu'il n'y a que deux champs par ligne,

et que ces champs sont codés comme ceci:

```
Nom du reseau   adresse de reseau
```

Un exemple:

```
loopnet    127.0.0.0
localnet   192.168.0.0
amprnet    44.0.0.0
```

Lorsque vous utilisez une commande comme *route*, si une destination est un réseau, et que ce réseau a une entrée dans le fichier `/etc/networks` la commande affichera alors le nom du réseau en lieu et place de son adresse.

5.10 Sécurité réseau et contrôle d'accès.

Laissez moi débuter ce paragraphe en vous mettant en garde que sécuriser votre machine et le réseau contre les attaques pernicieuses est un art complexe. Je ne me considère pas du tout comme un expert dans ce domaine, et bien que les mécanismes que je vais décrire, puissent vous aider, si vous êtes préoccupés par la sécurité, alors je vous recommande d'effectuer vous-mêmes des recherches sur le sujet. Il existe beaucoup de bonnes références sur l'Internet qui traitent du sujet.

Une importante règle pratique est: '**N'utilisez pas de serveurs dont vous n'avez pas l'utilité**'. Beaucoup de distributions arrivent configurées avec plein de services configurés et qui démarrent automatiquement. Pour assurer quand même un minimum de sécurité vous devriez aller dans votre fichier `/etc/inetd.conf` et retirez (*placez un '#' au début de la ligne*) toute entrée que vous ne comptez pas utiliser. De bons candidats sont: `shell`, `login`, `exec`, `uucp`, `ftp`, et les services informatifs tels que `finger`, `netstat` and `systat`.

Il y a plein de sortes de sécurité et de mécanismes de contrôle d'accès; je vais décrire les plus élémentaires.

5.10.1 /etc/ftpusers

Le fichier `/etc/ftpusers` est un mécanisme simple qui vous permet d'interdite l'accès de votre machine à certains utilisateurs de ftp. Il est lu par le programme démon (*ftpd*) lorsqu'une connexion ftp est reçue. Le fichier est une simple liste d'utilisateurs qui ne peuvent pas se connecter. Il ressemble à:

```
# /etc/ftpusers - utilisateurs ne pouvant pas se connecter par ftp
root
uucp
bin
mail
```

5.10.2 /etc/securetty

Le fichier `/etc/securetty` vous permet de spécifier sur quels fichiers de périphériques `tty` `root` a le droit de connecter. Le fichier `/etc/securetty` est lu par le programme de connexion (habituellement */bin/login*).

Son format est une liste de fichiers de périphériques autorisés, et sur tous les autres `root` ne peut le faire:

```
# /etc/securetty - consoles ou root peut se connecter
tty1
tty2
tty3
tty4
```

5.10.3 Le mécanisme de contrôle d'accès des hôtes *tcpd*.

Le programme *tcpd* que vous avez vu dans le fichier `/etc/inetd.conf` fournit les mécanismes de contrôle d'accès et de connexion aux services qu'il a pour but de protéger.

Lorsqu'il est invoqué par le programme *inetd*, il lit deux fichiers contenant les règles d'accès et il autorise ou interdit l'accès au serveur qu'il protège.

Il cherche dans ces deux fichiers jusqu'à ce qu'il trouve une correspondance. S'il n'en trouve pas il suppose que l'accès est autorisé. Il recherche dans l'ordre suivant: `/etc/hosts.allow`, `/etc/hosts.deny`. Je décrirai chacun d'eux plus tard. Pour une description complète référez-vous aux pages de manuel appropriées (`hosts_access(5)` est un bon point de départ).

/etc/hosts.allow Le fichier `/etc/hosts.allow` est un fichier de configuration du programme */usr/sbin/tcpd*. Il contient les hôtes dont l'accès est *autorisés (allowed)* et qui peuvent donc utiliser un service de votre machine.

Le format du fichier est très simple:

```
# /etc/hosts.allow
#
# <liste des services>: <liste des hotes> [: commande]
```

`liste des services`

c'est une liste de serveurs, séparés par des virgules, auxquels les règles d'accès s'appliquent. Exemples de serveur: `ftpd`, `telnetd`, et `fingerd`.

`liste des hôtes`

c'est une liste de noms d'hôtes, séparés par des virgules (vous pouvez utiliser également des adresses IP). Vous pouvez en plus spécifier des noms d'hôtes ou des adresses IP avec des jokers pour obtenir des groupes d'hôtes. Des exemples: `gw.vk2ktj.ampr.org` pour un hôte spécifique, `.uts.edu.au` pour tous les hôtes se terminant par cette chaîne , `44.` pour toutes les adresses IP commençant par ces chiffres. Il y a quelques expressions pour simplifier la configuration, parmi lesquelles: `ALL` pour tous les hôtes, `LOCAL` pour tout hôte dont le nom ne contient pas de '.' c'est à dire appartenant au même domaine que votre machine, et `PARANOID` pour tout hôte dont le nom ne correspond pas avec son adresse (tricherie dans le nom). Il y a enfin une expression qui peut être utile. Il s'agit de `EXCEPT` qui vous permet de fournir une liste avec des exceptions. Nous verrons un exemple plus tard.

`commande`

c'est un paramètre optionnel. Ce paramètre est le nom complet d'une commande (avec son répertoire) qui sera exécutée chaque fois qu'il y aura correspondance. Ce peut être par exemple une commande qui essayera d'identifier qui se connecte, ou de générer un message par courrier ou tout message d'alerte pour l'administrateur système avertissant que quelqu'un est en train de se connecter. On peut y inclure des extensions, par exemple: `%h` donnera le nom de l'hôte qui se connecte ou bien son adresse s'il n'a pas de nom , `%d` le programme démon appelé.

Un exemple:

```
# /etc/hosts.allow
#
# Permet a tout le monde d'utiliser le courrier
in.smtpd: ALL
# telnet et ftp pour les hotes de mon domaine et my.host.at.home.
telnetd, ftpd: LOCAL, myhost.athome.org.au
# finger pour tout le monde, mais garde une trace de l'identite.
fingerd: ALL: (finger @%h | mail -s "finger from %h" root)
```

/etc/hosts.deny Le fichier `/etc/hosts.deny` est un fichier de configuration du programme */usr/sbin/tcpd*. Ce fichier contient les hôtes qui *n'ont pas l'autorisation* d'accéder à l'un des services de votre machine.

Un exemple simple ressemblerait à ceci:

```
# /etc/hosts.deny
#
# Interdit l'acces aux hotes ayant des noms suspects
ALL: PARANOID
#
# Interdit l'acces a tous les hotes
ALL: ALL
```

L'entrée `PARANOID` est en fait redondante car l'autre entrée interdit dans tous les cas. L'une ou l'autre entrée devrait convenir, en fonction de vos besoins particuliers.

Mettre `ALL: ALL` par défaut dans le fichier `/etc/hosts.deny` puis autoriser certains services,en liaison avec les hôtes que vous avez choisis, dans le fichier `/etc/hosts.allow`, est la configuration la plus sûre.

5.10.4 /etc/hosts.equiv

Le fichier `hosts.equiv` est utilisé pour concéder à certains hôtes des droits d'accès leur permettant d'avoir un compte sur votre machine sans fournir de mot de passe. Cela est utile dans un environnement sécurisé où vous contrôlez toutes les machines, sinon ce peut être très risqué. Votre machine est aussi sûre que le moins sûr de vos hôtes de confiance. Pour augmenter la sécurité, n'utilisez pas cette possiblité et encouragez vos utilisateurs à utiliser le fichier `.rhosts`.

5.10.5 Configurer votre démon *ftp* correctement.

Beaucoup de sites sont intéressés à avoir un serveur *ftp* anonyme pour permettre aux autres de transférer et de récupérer des fichiers sans avoir besoin d'une identification spéciale. Si vous décidez d'offrir ce service soyez certains de configurer votre démon *ftp* de manière adéquate pour les accès anonymes. La plupart des pages de manuel dédiées à `ftpd(8)` décrivent tous les détails pour y arriver. Vous devez toujours vous assurer que vous avez bien suivi les instructions. Un règle importante est de ne pas utiliser une copie de votre fichier `/etc/passwd` dans le répertoire `/etc` du compte anonyme. Soyez sûrs d'avoir éliminé tous les détails des comptes exceptés ceux qui sont nécessaires , autrement vous serez vulnérables vis à vis de ceux qui maîtrisent les techniques de mise en pièce des mots de passe.

5.10.6 Firewall sur le réseau.

Ne pas permettre aux datagrammes d'atteindre votre machine ou les serveurs est un excellent moyen de sécurisation. Ceci est abordé en profondeur dans *Firewall-HOWTO* (`Firewall-HOWTO.html`).

5.10.7 Autres suggestions.

Voici d'autres suggestions à prendre en considération:

sendmail

en dépit de sa popularité, le démon *sendmail* apparaît avec une effrayante régularité dans les mises en garde concernant la sécurité. Faites comme vous voulez, mais j'ai choisi de ne pas l'utiliser.

NFS et autres services Sun RPC

soyez circonspects avec eux. Il y a toutes sortes d'exploits possibles avec ces services. Il est difficile de trouver une option pour les services tels que NFS, mais si vous les configurez, soyez prudents envers ceux à qui vous accordez des droits.

6 Informations spécifiques sur les technologies réseau.

Les sous-chapitres suivants traitent de technologies réseau particulières. Les informations mentionnées ne s'appliquent pas nécessairement aux autres types.

6.1 ARCNet

Les noms de fichier périphériques de ARCNet sont '`arc0e`', '`arc1e`', '`arc2e`' ... ou bien '`arc0s`', '`arc1s`', '`arc2s`', etc. La première carte détectée par le noyau devient '`arc0e`' ou '`arc0s`' et les autres sont nommées en suivant dans l'ordre de leur détection. La lettre finale dépend de votre choix: soit un format d'encapsulation de paquets Ethernet, soit un format de paquets suivant RFC1051.

Options de compilation du noyau:

```
Network device support  --->
```

```
[*] Network device support
<*> ARCnet support
[ ]   Enable arc0e (ARCnet "Ether-Encap" packet format)
[ ]   Enable arc0s (ARCnet RFC1051 packet format)
```

Si vous avez construit convenablement votre noyau pour supporter votre carte Ethernet, alors la configuration de la carte est facile.

Typiquement vous devriez utiliser quelque chose comme ceci:

```
# ifconfig arc0e 192.168.0.1 netmask 255.255.255.0 up
# route add -net 192.168.0.0 netmask 255.255.255.0 arc0e
```

Merci de vous référer aux documents `/usr/src/linux/Documentation/networking/arcnet.txt` et `/usr/src/linux/Docume` pour d'autres informations.

Le support ARCNet fut développé par Avery Pennarun, `apenwarr@foxnet.net`.

6.2 Appletalk (AF_APPLETALK)

Le support Appletalk ne possède pas de noms de périphériques spécifiques car il utilise les périphériques réseau existants.

Options de compilation noyau:

```
Networking options  --->
    <*> Appletalk DDP
```

Le support Appletalk permet à votre machine Linux de dialoguer avec les réseaux Apple. Son utilisation principale est de pouvoir partager des ressources, comme les imprimantes et les disques, entre vos ordinateurs Linux et Apple. Un logiciel supplémentaire est requis, il s'appelle *netatalk*. Wesley Craig `netatalk@umich.edu` représente une équipe appelée le 'Research Systems Unix Group' à l'université du Michigan. Celle-ci a élaboré le paquetage *netatalk*, qui fournit un logiciel implémentant la pile protocole Appletalk et quelques utilitaires. Soit ce paquetage *netatalk* vous a été fourni avec votre distribution Linux, soit vous pouvez le récupérer par ftp depuis le site *University of Michigan* (`ftp://terminator.rs.itd.umich.edu/unix/netatalk/`)

Pour construire et installer le paquetage, vous faites:

```
# cd /usr/src
# tar xvfz .../netatalk-1.4b2.tar.Z
- Vous pouvez editer le fichier 'Makefile' a ce stade, plus
precisement pour changer la valeur de la variable
 DESTDIR qui definit l'endroit ou les fichiers seront
installes plus tard.
 Le repertoire par defaut,/usr/local/atalk, semble
 tres raisonnable.
# make
- puis, en temps que root:
# make install
```

6.2.1 Configurer le support Appletalk.

La première chose à faire pour que tout fonctionne est de vérifier que les entrées adéquates sont présentes dans le fichier `/etc/services`. Ces entrées sont:

```
rtmp    1/ddp   # Routing Table Maintenance Protocol
nbp     2/ddp   # Name Binding Protocol
echo    4/ddp   # AppleTalk Echo Protocol
zip     6/ddp   # Zone Information Protocol
```

L'étape suivante consiste à créer les fichiers de configuration Appletalk dans le répertoire `/usr/local/atalk/etc` (ou bien à l'endroit où vous avez installé le paquetage).

Le premier fichier à créer est `/usr/local/atalk/etc/atalkd.conf`. Initialement ce fichier ne nécessite qu'une ligne qui indique le périphérique supportant le réseau sur lequel sont vos machines Apple:

```
eth0
```

Le programme démon Appletalk ajoutera d'autres détails quand il tournera.

6.2.2 Exporter un système de fichiers Linux avec Appletalk.

Vous pouvez exporter des systèmes fichiers depuis votre machine Linux vers le réseau en sorte qu'une machine Apple puisse les partager.

Pour cela vous devez configurer le fichier `/usr/local/atalk/etc/AppleVolumes.system`. Il y a une autre fichier de configuration appelé `/usr/local/atalk/etc/AppleVolumes.default` qui a exactement le même format et qui décrit quels systèmes de fichiers les utilisateurs connectés pourront recevoir avec des privilèges d'invités.

Tous les détails, qui vous diront comment configurer ces fichiers et avec quelles options, peuvent être trouvés dans la page de manuel de *afpd*.

Un simple exemple:

```
/tmp Scratch
/home/ftp/pub "Public Area"
```

Ce qui exportera votre système de fichiers `/tmp` comme volume AppleShare 'Scratch' et votre répertoire public ftp comme volume AppleShare 'Public Area'. Les noms de volume ne sont pas obligatoires, le programme démon choisissant pour vous, mais ça ne coûte rien de les spécifier quand même.

6.2.3 Partager votre imprimante Linux avec Appletalk.

Partager votre imprimante Linux avec vos machines Apple est très simple. Vous devez faire tourner le programme *papd* qui est le démon protocole d'accès aux imprimantes de Appletalk. Lorsque vous faites tourner ce programme il acceptera les requêtes émanant de vos machines Apple et spoulera le travail d'impression vers votre démon local d'impression.

Vous devrez éditer le fichier `/usr/local/atalk/etc/papd.conf` pour configurer le démon. La syntaxe de ce fichier est la même que le fichier habituel `/etc/printcap`. Le nom que vous donnez à la définition de l'imprimante doit être conforme au protocole de désignation Appletalk, NBP.

Un exemple de configuration ressemble à ceci:

```
TricWriter:\
   :pr=lp:op=cg:
```

Ce qui fera une imprimante nommée 'TricWriter' disponible pour le réseau Appletalk et tous les travaux acceptés seront imprimés sur l'imprimante linux '`lp`' (telle que définie dans le fichier `/etc/printcap`) utilisant *lpd*. L'entrée '`op=cg`' indique que l'utilisateur linux '`cg`' est l'opérateur de l'imprimante.

6.2.4 Démarrer Appletalk.

Bon, vous devriez être prêts pour essayer cette configuration de base. Le fichier *rc.atalk* fourni avec le paquetage *netatalk* devrait vous convenir, alors vous faites ceci:

```
# /usr/local/atalk/etc/rc.atalk
```

et tout devrait démarrer et tourner sans problémes. Vous ne devriez voir aucun message d'erreurs et le programme devrait vous envoyer des messages sur la console indiquant chaque étape qui démarre.

6.2.5 Tester Appletalk.

Pour tester si le programme fonctionne correctement, allez sur une des machines Apple, faites dérouler le menu Pomme, cliquez sur AppleShare, et votre boîte Linux devrait apparaitre.

6.2.6 Mises en garde sur Appletalk.

- Vous aurez peut-être besoin de démarrer votre support Appletalk avant de configurer votre réseau IP. Si vous avez des problèmes pour démarrer vos programmes Appletalk, ou si après les avoir démarrés vous avez des ennuis avec votre réseau IP, essayez alors de mettre en route votre programme Appletalk avant de faire démarrer `/etc/rc.d/rc.inet1`.
- Le démon *afpd* (Apple Filing Protocol Daemon) secoue sévèrement votre disque dur. Derrière les points de montage il crée deux répertoires appelés `.AppleDesktop` et `Network Trash Folder`. Ensuite, pour chaque répertoire auquel vous accédez il crée un sous-répertoire `.AppleDouble` pour pouvoir stocker des fichiers de ressource, etc. Réfléchissez bien avant d'exporter `/`, vous aurez besoin de beaucoup de temps pour tout nettoyer.
- Le programme *afpd* attend des mots de passe en clair venant des Macs. La sécurité pouvant être un problème, soyez donc attentifs lors de l'utilisation de ce démon sur une machine connectée sur l'Internet et ne vous en prenez qu'à vous-même si quelqu'un de mal intentionné vous fait des misères.
- Les outils de diagnostic existants tels que *netstat* et *ifconfig* ne supportent pas Appletalk. Les informations peuvent être trouvées dans le répertoire `/proc/net/` si vous en avez besoin.

6.2.7 Autres informations

Pour en savoir plus sur la configuration de Appletalk pour Linux, référez vous à la page de Anders Brownworth *Linux Netatalk-HOWTO* sur *thehamptons.com* (`http://thehamptons.com/anders/netatalk/`).

6.3 ATM

Werner Almesberger <`werner.almesberger@lrc.di.epfl.ch`> dirige un projet en vue de fournir un support Mode de Transfert Asynchrone (Asynchronous Transfer Mode) pour Linux. Les informations sur l'état du projet se trouvent sur: *lrcwww.epfl.ch* (`http://lrcwww.epfl.ch/linux-atm/`).

6.4 AX25 (`AF_AX25`)

Les noms de périphériques AX.25 sont '`sl0`', '`sl1`', etc. avec les noyaux `2.0.*` ou '`ax0`', '`ax1`', etc. avec les noyaux `2.1.*`.

Options de compilation du noyau:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
```

Les protocoles AX25, Netrom et Rose sont couverts par le document *AX25-HOWTO* (`AX25-HOWTO.html`). Ces protocoles sont utilisés par les radio-amateurs du monde entier pour l'expérimentation packet-radio.

La plupart du travail d'implémentation de ces protocoles a été réalisé par Jonathon Naylor, `jsn@cs.nott.ac.uk`.

6.5 DECNet

Le support pour DECNet est en cours d'élaboration. Vous devriez le voir apparaitre dans l'un des prochains noyaux `2.1.*`.

6.6 EQL - égaliseur de charge à lignes multiples

Le nom du périphérique EQL est '`eql`'. Avec les sources standards du noyau vous ne pouvez avoir qu'un seul périphérique EQL par machine. EQL permet d'utiliser plusieurs lignes point à point telles que PPP, SLIP ou PLIP comme si c'était un seul lien logique de transport tcp/ip. C'est souvent moins cher d'utiliser plusieurs lignes à faible débit que d'avoir une ligne à haut débit.

Options de compilation du noyau:

```
Network device support --->
    [*] Network device support
    <*> EQL (serial line load balancing) support
```

Pour supporter ce mécanisme la machine à l'autre bout de la ligne doit également supporter EQL. Linux, Livingstone Portmasters et de nouveaux serveurs de ligne supportent des systèmes compatibles.

Pour configurer EQL vous avez besoin des outils eql, disponibles sur: *sunsite.unc.edu* (`ftp://sunsite.unc.edu/pub/linux/s`

La configuration est plutôt directe. Vous commencez par configurer l'interface eql. C'est exactement comme un autre périphérique réseau. Vous configurez l'adresse IP et le mtu en utilisant l'outil *ifconfig* , comme ceci:

```
ifconfig eql 192.168.10.1 mtu 1006
```

Ensuite vous devez initialiser manuellement chacune des lignes que vous allez utiliser. Ce peut être toute combinaison de périphériques réseau point à point. La façon d'initialiser les connexions dépend du type de lien, voyez les paragraphes appropriés pour d'autres informations.

Enfin vous devez associer le lien série et le dispositif EQL, cela s'appelle 'mise en esclavage' (enslaving) et est réalisé avec la commande *eql_enslave* comme suit:

```
eql_enslave eql sl0 28800
eql_enslave eql ppp0 14400
```

Le paramètre '*estimated speed*' que vous fournissez à *eql_enslave* ne fait rien directement. Il est utilisé par le pilote EQL pour déterminer comment les datagrammes vont se répartir sur ce périphérique, aussi vous pouvez régler l'équilibrage des lignes en jouant avec cette valeur.

Pour libérer une ligne d'un périphérique EQL vous utilisez la commande *eql_emancipate* comme ci-dessous:

```
eql_emancipate eql sl0
```

Vous ajoutez le routage comme vous le feriez pour tout lien point à point; sauf que vos routes doivent se rapporter au dispositif `eql` plutôt qu'aux périphériques séries eux-mêmes. Typiquement vous devriez utiliser:

```
route add default eql
```

Le pilote EQL fut développé par Simon Janes, `simon@ncm.com`.

6.7 Ethernet

Les noms de périphériques Ethernet sont '`eth0`', '`eth1`', '`eth2`' etc. La première carte détectée par le noyau devient '`eth0`' et le reste est nommé dans l'ordre de détection.

Pour savoir comment faire marcher votre carte Ethernet sous Linux référez-vous au *Ethernet-HOWTO* (`Ethernet-HOWTO.html`).

Une fois que vous avez compilé convenablement votre noyau pour supporter les cartes Ethernet, la configuration des cartes est aisée.

Typiquement vous faites ainsi:

```
# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
# route add -net 192.168.0.0 netmask 255.255.255.0 eth0
```

La plupart des pilotes Ethernet ont été développés par Donald Becker, `becker@CESDIS.gsfc.nasa.gov`.

6.8 FDDI (Fiber Distributed Data Interface)

Les noms de périphériques FDDI sont '`fddi0`', '`fddi1`', '`fddi2`' etc. La première carte détectée par le noyau s'appelle '`fddi0`' et le reste est nommé dans l'ordre de détection.

Larry Stefani, `lstefani@ultranet.com`, a développé un pilote pour les cartes Digital Equipment Corporation FDDI EISA et PCI.

Options de compilation noyau:

```
Network device support  --->
    [*] FDDI driver support
    [*] Digital DEFEA and DEFPA adapter support
```

Lorsque vous avez construit votre noyau pour supporter le pilote FDDI et qu'il est installé, la configuration de l'interface FDDI est presque identique à celle d'une interface Ethernet. Vous devez spécifier le nom de l'interface FDDI appropriée dans les commandes *ifconfig* et *route*.

6.9 Relais de trame (Frame Relay)

Les noms de périphériques de 'Frame Relay' sont '`dlci00`', '`dlci01`' etc pour les systèmes d'encapsulation DLCI et '`sdla0`', '`sdla1`' etc pour les FRAD(s) (Frame Relay Access Device).

Frame Relay est une nouvelle technologie réseau conçue pour s'adapter au trafic de transmission de données 'par à coups' ou de nature intermittente. Vous vous connectez à un réseau Frame Relay en utilisant un dispositif d'accès Frame Relay (FRAD). Les supports Linux Frame Relay supportent IP par dessus Frame Relay comme décrit dans la RFC-1490.

Options de compilation noyau:

```
Network device support  --->
    <*> Frame relay DLCI support (EXPERIMENTAL)
    (24)   Max open DLCI
    (8)   Max DLCI per device
    <*>   SDLA (Sangoma S502/S508) support
```

Mike McLagan, `mike.mclagan@linux.org`, a développé le support Frame Relay et les outils de configuration.

A l'heure actuelle le seul FRAD supporté est *Sangoma Technologies* (`http://www.sangoma.com/`) `S502A`, `S502E` et `S508`.

Pour configurer les systèmes FRAD et DLCI après avoir reconstruit votre noyau, vous aurez besoin des outils de configuration. Ils sont disponibles sur *ftp.invlogic.com* (`ftp://ftp.invlogic.com/pub/linux/fr/frad-0.15.tgz`). Compiler et installer les outils est très facile, mais le manque de fichier Makefile au premier niveau oblige à le faire à la main:

```
# cd /usr/src
```

```
# tar xvfz .../frad-0.15.tgz
# cd frad-0.15
# for i in common dlci frad; make -C $i clean; make -C $i; done
# mkdir /etc/frad
# install -m 644 -o root -g root bin/*.sfm /etc/frad
# install -m 700 -o root -g root frad/fradcfg /sbin
# install -m 700 -o root -g root dlci/dlcicfg /sbin
```

Après l'installation vous devez créer un fichier `/etc/frad/router.conf` Vous pouvez utiliser cet exemple, qui est une version modifiée de l'un des fichiers donné en exemple:

```
# /etc/frad/router.conf
# C'est un modele de configuration pour relais de trame.
# Tout y est inclus. Les valeurs par defaut sont basees sur le code
# fourni avec les pilotes DOS de la carte Sangoma S502A.
#
# Une ligne avec '#' est un commentaire
# Les blancs sont ignores (vous pouvez utiliser des tabulations aussi).
# Les sections [] inconnues et les entrees inconnues sont ignorees.
#

[Devices]
Count=1                 # number of devices to configure
Dev_1=sdla0             # the name of a device
#Dev_2=sdla1            # the name of a device

# Ce qui est specifie ici s'applique a tous les peripheriques, et peut etre
# mis a jour pour chaque carte individuelle.
#
Access=CPE
Clock=Internal
KBaud=64
Flags=TX
#
# MTU=1500              # Maximum transmit IFrame length, default is 4096
# T391=10               # T391 value    5 - 30, default is 10
# T392=15               # T392 value    5 - 30, default is 15
# N391=6                # N391 value    1 - 255, default is 6
# N392=3                # N392 value    1 - 10, default is 3
# N393=4                # N393 value    1 - 10, default is 4

# On specifie ici les valeurs par defaut pour toutes les cartes
# CIRfwd=16             # CIR forward   1 - 64
# Bc_fwd=16             # Bc forward    1 - 512
# Be_fwd=0              # Be forward    0 - 511
# CIRbak=16             # CIR backward  1 - 64
# Bc_bak=16             # Bc backward   1 - 512
# Be_bak=0              # Be backward   0 - 511
```

```
#
#
# Configurations specifiques
#
#

#
#  Sangoma S502E
#
[sdla0]
Type=Sangoma            # Type of the device to configure, currently only
                        # SANGOMA is recognised
#
# Specifique des types 'Sangoma'
#
# cartes S502A, S502E, S508
Board=S502E
#
# Le nom du logiciel de carte en essai pour Sangoma
# Testware=/usr/src/frad-0.10/bin/sdla_tst.502
#
# Le nom du logiciel de carte FR
# Firmware=/usr/src/frad-0.10/bin/frm_rel.502
#
Port=360                # Port for this particular card
Mem=C8                  # Address of memory window, A0-EE, depending on card
IRQ=5                   # IRQ number, do not supply for S502A
DLCIs=1                 # Number of DLCI's attached to this device
DLCI_1=16               # DLCI #1's number, 16 - 991
# DLCI_2=17
# DLCI_3=18
# DLCI_4=19
# DLCI_5=20
#
# Ce qui est specifie ici s'applique au peripherique seulement,
# et remplace les valeurs par defaut
#
# Access=CPE            # CPE or NODE, default is CPE
# Flags=TXIgnore,RXIgnore,BufferFrames,DropAborted,Stats,MCI,AutoDLCI
# Clock=Internal        # External or Internal, default is Internal
# Baud=128              # Specified baud rate of attached CSU/DSU
# MTU=2048              # Maximum transmit IFrame length, default is 4096
# T391=10               # T391 value    5 - 30, default is 10
# T392=15               # T392 value    5 - 30, default is 15
# N391=6                # N391 value    1 - 255, default is 6
# N392=3                # N392 value    1 - 10, default is 3
```

```
# N393=4                # N393 value    1 - 10, default is 4

#
# Le second peripherique est une autre carte
#
# [sdla1]
# Type=FancyCard        # Type of the device to configure.
# Board=                # Type of Sangoma board
# Key=Value             # values specific to this type of device


#
# Parametres de configuration DLCI par defaut.
# Peuvent etre ecrases par des configurations specifiques
#
CIRfwd=64               # CIR forward   1 - 64
# Bc_fwd=16             # Bc forward    1 - 512
# Be_fwd=0              # Be forward    0 - 511
# CIRbak=16             # CIR backward  1 - 64
# Bc_bak=16             # Bc backward   1 - 512
# Be_bak=0              # Be backward   0 - 511

#
# Configuration DLCI
# Optionnel. La convention d'appellation est
# [DLCI_D<devicenum>_<DLCI_Num>]
#

[DLCI_D1_16]
# IP=
# Net=
# Mask=
# Flags defined by Sangoma: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
# CIRfwd=64
# Bc_fwd=512
# Be_fwd=0
# CIRbak=64
# Bc_bak=512
# Be_bak=0

[DLCI_D2_16]
# IP=
# Net=
# Mask=
# Flags defined by Sangoma: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
# CIRfwd=16
```

```
# Bc_fwd=16
# Be_fwd=0
# CIRbak=16
# Bc_bak=16
# Be_bak=0
```

Lorsque vous avez construit votre fichier `/etc/frad/router.conf` la seule étape restante est de configurer les périphériques eux-mêmes. C'est un tout petit peu plus compliqué que la configuration normale d'un périphérique réseau; vous devez vous souvenir de monter le périphérique FRAD avant les périphériques d'encapsulation DLCI.

```
# Configure le materiel frad et les parametres DLCI
/sbin/fradcfg /etc/frad/router.conf || exit 1
/sbin/dlcicfg file /etc/frad/router.conf
#
# Montage du dispositif FRAD
ifconfig sdla0 up
#
# Configure les interfaces d'encapsulation DLCI et le routage
ifconfig dlci00 192.168.10.1 pointopoint 192.168.10.2 up
route add -net 192.168.10.0 netmask 255.255.255.0 dlci00
#
ifconfig dlci01 192.168.11.1 pointopoint 192.168.11.2 up
route add -net 192.168.11.0 netmask 255.255.255.0 dlci00
#
route add default dev dlci00
#
```

6.10 Enregistrement IP (IP Accounting)

Les possibilités d'enregistrement IP du noyau Linux vous permettent de recueillir et d'analyser les données d'utilisation du réseau. Les données collectées comprennent le nombre de paquets et le nombre d'octets en cumul depuis la dernière remise à zéro. Vous avez à votre disposition une grande variété de règlages pour obtenir les données que vous désirez.

Options de compilation noyau:

```
Networking options  --->
    [*] IP: accounting
```

Après avoir compilé et installé le noyau vous devez utiliser la commande *ipfwadm* pour configurer l'enregistrement IP. Il y a différentes possibilités pour choisir les informations à enregistrer. J'ai pris un exemple simplifié qui pourrait vous être utile; lisez plutôt la page de manuel *ipfwadm* pour plus d'informations.

Scenario: Vous avez un réseau Ethernet qui est relié à l'Internet via une liaison PPP. Sur l'Ethernet vous avez une machine qui offre un grand nombre de services et vous voulez savoir quel trafic est engendré par le trafic telnet, rlogin, ftp et www.

Vous pouvez utiliser une commande qui ressemble à ceci:

```
#
# Donne les reglages d'enregistrement
ipfwadm -A -f
#
# Ajoute des reglages pour le segment Ethernet local
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 20
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 20
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 23
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 23
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 80
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 80
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 513
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 513
ipfwadm -A in -a -P tcp -D 44.136.8.96/29
ipfwadm -A out -a -P tcp -D 44.136.8.96/29
ipfwadm -A in -a -P udp -D 44.136.8.96/29
ipfwadm -A out -a -P udp  -D 44.136.8.96/29
ipfwadm -A in -a -P icmp -D 44.136.8.96/29
ipfwadm -A out -a -P icmp -D 44.136.8.96/29
#
# Reglages par defaut
ipfwadm -A in -a -P tcp -D 0/0 20
ipfwadm -A out -a -P tcp -S 0/0 20
ipfwadm -A in -a -P tcp -D 0/0 23
ipfwadm -A out -a -P tcp -S 0/0 23
ipfwadm -A in -a -P tcp -D 0/0 80
ipfwadm -A out -a -P tcp -S 0/0 80
ipfwadm -A in -a -P tcp -D 0/0 513
ipfwadm -A out -a -P tcp -S 0/0 513
ipfwadm -A in -a -P tcp -D 0/0
ipfwadm -A out -a -P tcp -D 0/0
ipfwadm -A in -a -P udp -D 0/0
ipfwadm -A out -a -P udp  -D 0/0
ipfwadm -A in -a -P icmp -D 0/0
ipfwadm -A out -a -P icmp -D 0/0
#
# Liste les reglages
ipfwadm -A -l -n
#
```

La dernière commande liste chacune des règles d'enregistrement et affiche le total.

Il est important de noter, lorsque l'on analyse les enregistrement IP, que **les totaux sont incrémentés à chaque fois**, donc pour connaitre les différences vous devez exécuter les opérations mathématiques nécessaires. Par exemple si je veux savoir combien de données ne venaient pas de ftp, telnet, rlogin ou www je

dois soustraire les totaux individuels correspondant à chaque port.

```
# ipfwadm -A -l -n
IP accounting rules
 pkts bytes dir prot source               destination          ports
    0     0 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 20
    0     0 out tcp  44.136.8.96/29       0.0.0.0/0            20 -> *
    0     0 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 23
    0     0 out tcp  44.136.8.96/29       0.0.0.0/0            23 -> *
   10  1166 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 80
   10   572 out tcp  44.136.8.96/29       0.0.0.0/0            80 -> *
  242  9777 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 513
  220 18198 out tcp  44.136.8.96/29       0.0.0.0/0            513 -> *
  252 10943 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> *
  231 18831 out tcp  0.0.0.0/0            44.136.8.96/29       * -> *
    0     0 in  udp  0.0.0.0/0            44.136.8.96/29       * -> *
    0     0 out udp  0.0.0.0/0            44.136.8.96/29       * -> *
    0     0 in  icmp 0.0.0.0/0            44.136.8.96/29       *
    0     0 out icmp 0.0.0.0/0            44.136.8.96/29       *
    0     0 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 20
    0     0 out tcp  0.0.0.0/0            0.0.0.0/0            20 -> *
    0     0 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 23
    0     0 out tcp  0.0.0.0/0            0.0.0.0/0            23 -> *
   10  1166 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 80
   10   572 out tcp  0.0.0.0/0            0.0.0.0/0            80 -> *
  243  9817 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 513
  221 18259 out tcp  0.0.0.0/0            0.0.0.0/0            513 -> *
  253 10983 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> *
  231 18831 out tcp  0.0.0.0/0            0.0.0.0/0            * -> *
    0     0 in  udp  0.0.0.0/0            0.0.0.0/0            * -> *
    0     0 out udp  0.0.0.0/0            0.0.0.0/0            * -> *
    0     0 in  icmp 0.0.0.0/0            0.0.0.0/0            *
    0     0 out icmp 0.0.0.0/0            0.0.0.0/0            *
#
```

6.11 IP Aliasing

Il y a des applications où être en mesure d'affecter plusieurs adresses IP à un seul périphérique réseau pourrait être utile. Certains fournisseurs d'accès à l'Internet utilise souvent cette possibilité pour fournir des offres www et ftp 'à la carte' pour leurs clients. Vous pouvez vous référer au mini-HOWTO IP-Aliasing pour plus d'informations.

Options de compilation du noyau:

```
Networking options  --->
    ....
    [*] Network aliasing
    ....
```

```
<*> IP: aliasing support
```

Aprés avoir compilé et installé le noyau avec le support IP_Alias, la configuration est très simple. Les alias sont ajoutés aux périphériques réseau virtuels associés au périphérique réseau réel. Une simple convention de noms s'applique pour périphériques: `<nom de périphérique>:<numéro de périphérique virtuel>`, par ex. `eth0:0`, `ppp0:10` etc. Notez que le pilote de périphérique ifname:number ne peut être configuré *qu'après* le réglage de l'interface principale.

Par exemple, supposons que vous ayez un réseau Ethernet avec simultanément deux sous-réseaux IP et que vous vouliez que votre machine ait un accès direct aux deux, vous pouvez faire quelque chose comme ceci:

```
#
# ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# ifconfig eth0:0 192.168.10.1 netmask 255.255.255.0 up
# route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
#
```

Pour supprimer un alias vous ajoutez simplement un '`-`' au bout de son nom et et vous faites aussi simplement que ça:

```
# ifconfig eth0:0- 0
```

Toutes les routes associées avec cet alias seront enlevées automatiquement.

6.12 IP Firewall

IP Firewall et les publications sur le Firewalling sont traités plus complètement dans le document *Firewall-HOWTO* (`Firewall-HOWTO.html`). Le IP Firewalling vous permet de sécuriser votre machine contre les accès réseau non-autorisés en filtrant, ou acceptant, des datagrammes venant de, ou allant vers, des adresses IP de votre choix. Il y a différentes règles: le filtrage en entrée, le filtrage en sortie, et le filtrage en retransmission. Les règles en entrée s'appliquent aux datagrammes qui sont reçues par un dispositif réseau. Les règles en sortie s'appliquent aux datagrammes qui sont transmis par un dispositif réseau. Les règles en retransmission s'appliquent aux datagrammes qui ne sont pas pour cette machine, c'est à dire les datagrammes qui seront reroutés.

Options de compilation noyau:

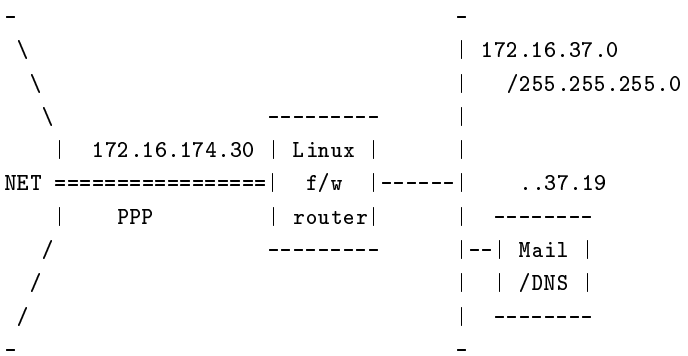
```
Networking options  --->
    [*] Network firewalls
    ....
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: firewalling
    [ ] IP: firewall packet logging
```

La configuration du IP firewall est réalisée en utilisant la commande *ipfwadm*. Comme mentionné plus haut, la sécurité n'est pas ma spécialité, aussi, bien que je vous présente un exemple utilsable par vous-même, faites des recherches et mettez au point vos propres règlages si la sécurité est importante pour vous.

Vraisemblablement l'utilisation la plus courante de l'IP firewall est lorsque vous utilisez votre machine Linux comme routeur et passerelle firewall et que vous voulez protéger votre réseau local contre les accès extérieurs non autorisés.

La configuration suivante est due à Arnt Gulbrandsen, <agulbra@troll.no>.

L'exemple décrit une configuration de firewall pour une machine Linux /firewall/routeur illustrée par ce diagramme:



Les commandes suivantes doivent être normalement placées dans un fichier `rc` de telle sorte qu'elles seront démarrées automatiquement à chaque redémarrage du système. Pour une sécurité maximum, elles devront être effectuées après la configuration des interfaces réseau, mais avant le montage de ces interfaces pour éviter que quelqu'un ne puisse se connecter pendant que la machine firewall reboute.

```
#!/bin/sh

# Flush the 'Forwarding' rules table
# Change the default policy to 'accept'
#
/sbin/ipfwadm -F -f
/sbin/ipfwadm -F -p accept
#
# .. and for 'Incoming'
#
/sbin/ipfwadm -I -f
/sbin/ipfwadm -I -p accept

# First off, seal off the PPP interface
# I'd love to use '-a deny' instead of '-a reject -y' but then it
# would be impossible to originate connections on that interface too.
# The -o causes all rejected datagrams to be logged. This trades
# disk space against knowledge of an attack of configuration error.
#
```

```
/sbin/ipfwadm -I -a reject -y -o -P tcp -S 0/0 -D 172.16.174.30

# Throw away certain kinds of obviously forged packets right away:
# Nothing should come from multicast/anycast/broadcast addresses
#
/sbin/ipfwadm -F -a deny -o -S 224.0/3 -D 172.16.37.0/24
#
# and nothing coming from the loopback network should ever be
# seen on a wire
#
/sbin/ipfwadm -F -a deny -o -S 127.0/8 -D 172.16.37.0/24

# accept incoming SMTP and DNS connections, but only
# to the the Mail/Name Server
#
/sbin/ipfwadm -F -a accept -P tcp -S 0/0 -D 172.16.37.19 25 53
#
# DNS uses UDP as well as TCP, so allow that too
# for questions to our name server
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 -D 172.16.37.19 53
#
# but not "answers" coming to dangerous ports like NFS and
# Larry McVoy's NFS extension.  If you run squid, add its port here.
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 53 \
        -D 172.16.37.0/24 2049 2050

# answers to other user ports are okay
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 53 \
        -D 172.16.37.0/24 53 1024:65535

# Reject incoming connections to identd
# We use 'reject' here so that the connecting host is told
# straight away not to bother continuing, otherwise we'd experience
# delays while ident timed out.
#
/sbin/ipfwadm -F -a reject -o -P tcp -S 0/0 -D 172.16.37.0/24 113

# Accept some common service connections from the 192.168.64 and
# 192.168.65 networks, they are friends that we trust.
#
/sbin/ipfwadm -F -a accept -P tcp -S 192.168.64.0/23 \
        -D 172.16.37.0/24 20:23

# accept and pass through anything originating inside
#
```

```
/sbin/ipfwadm -F -a accept -P tcp -S 172.16.37.0/24 -D 0/0

# deny most other incoming TCP connections, and log them
# (append 1:1023 if you have problems with ftp not working)
#
/sbin/ipfwadm -F -a deny -o -y -P tcp -S 0/0 -D 172.16.37.0/24

# ... for UDP too
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 -D 172.16.37.0/24
```

De bonnes configurations firewall sont dures à faire. Cet exemple peut être un bon point de départ pour vous. La page de manuel *ipfwadm* offre une aide pour savoir comment utiliser cet outil. Si vous voulez configurer un firewall, demandez autour de vous et recueillez des avis venant de sources de confiance et prenez contact avec quelqu'un qui est à l'extérieur pour tester votre configuration et en vérifier la fiabilité.

6.13 Encapsulation IPIP

Pourquoi vouloir encapsuler des paquets IP dans d'autres paquets IP? Cela semble bizarre si vous n'avez jamais vu d'applications auparavant. Il y a deux endroits où c'est utilisé: le Mobile-IP et l'IP-Multicast. C'est dans un environnement qui est peut-être le plus largement utilisé et qui est le moins connu: l'amateurisme-Radio.

Options de compilation du noyau:

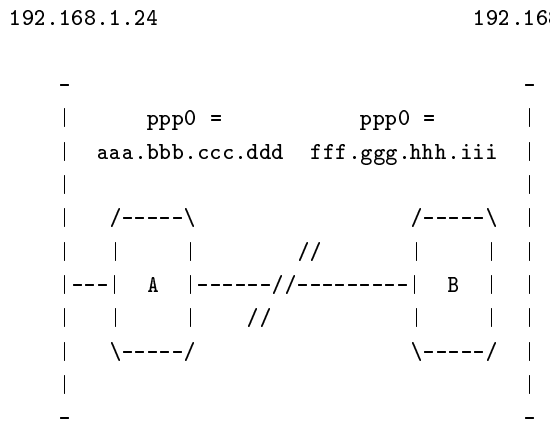
```
Networking options  --->
    [*] TCP/IP Networking
    [*] IP: forwarding/gatewaying
    ....
    <*> IP tunelling
```

Les périphériques IP tunnel s'appellent 'tunl0', 'tunl1', etc..

"Mais pourquoi?" D'accord. D'accord. Les règles de routage classiques spécifient qu'un réseau IP comprend une adresse IP et un masque de réseau. Ceci fournit un ensemble d'adresses contiguës qui peuvent toutes être routées par l'intermédiaire d'une seule entrée de routage. Cela marche, mais signifie que vous ne pouvez utiliser une seule adresse uniquement lorsque vous êtes connecté à un point du réseau auquelle elle appartient. Dans la plupart des cas, il n'y a pas de problèmes, mais si vous êtes en mouvement alors vous ne pouvez pas rester connecté au même endroit tout le temps. L'encapsulation IP/IP (IP tunneling) vous permet de passer outre cette contrainte en permettant aux paquets destinés à votre adresse d'être enveloppés et redirigés vers une autre adresse. Si vous savez que vous allez opérer depuis un autre réseau IP pour quelques temps, vous pouvez régler une machine qui est chez vous pour accepter des paquets destinés à votre adresse IP et les rediriger vers l'adresse que vous allez utiliser provisoirement.

6.13.1 Une configuration de réseau avec tunneling.

Comme toujours, je pense qu'un schéma m'épargnera beaucoup de texte confus, aussi en voilà un:



Ce diagramme montre une autre raison possible d'utiliser l'encapsulation IPIP: le réseau privé virtuel. Cet exemple présuppose que vous ayez deux machines chacune avec une seule connexion Internet. Chaque hôte a une seule adresse IP. Derrière chacune de ces machines se trouve des réseaux privés locaux configurés avec des adresses IP réservées. Supposez que vous vouliez permettre à chacun des hôtes du groupe A de se connecter à n'importe quel hôte du groupe B, comme s'ils étaient vraiment connectés à l'Internet via un routage réseau. L'encapsulation IPIP vous permettra de le faire. A noter que l'encapsulation ne vous permettra pas de faire en sorte que chacun des hôtes des réseaux A et B puissent parler à n'importe qui sur l'Internet, vous aurez toujours besoin de choses comme IP masquerade pour pouvoir le faire. L'encapsulation est normalement accomplie par une machine fonctionnant comme routeur.

Le routeur Linux 'A' sera configuré comme suit:

```
#
PATH=/sbin:/usr/sbin
#
# Ethernet configuration
ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.1.1 up
route add -net 192.168.2.0 netmask 255.255.255.0 gw fff.ggg.hhh.iii tunl0
```

Le routeur Linux 'B' sera configuré comme suit:

```
#
```

```
PATH=/sbin:/usr/sbin
#
# Ethernet configuration
ifconfig eth0 192.168.2.1 netmask 255.255.255.0 up
route add -net 192.168.2.0 netmask 255.255.255.0 eth0
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.2.1 up
route add -net 192.168.1.0 netmask 255.255.255.0 gw aaa.bbb.ccc.ddd tunl0
```

La commande:

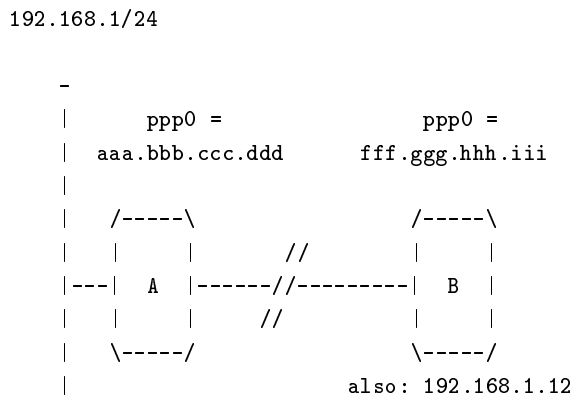
```
route add -net 192.168.1.0 netmask 255.255.255.0 gw aaa.bbb.ccc.ddd tunl0
```

dit: 'Envoyer tous les datagrammes destinés à `192.168.1.0/24` dans un paquet d'encapsulation ayant pour adresses de destination `aaa.bbb.ccc.ddd`'.

Notez que les configurations sont inversées à l'autre bout. Le périphérique tunnel utilise '`gw`' dans la commande route comme *destination* du paquet IP où se trouve le datagramme qu'il doit router. Cette machine doit savoir comment 'désencapsuler' les paquets IPIP, c'est à dire qu'elle doit aussi être configurée comme périphérique tunnel.

6.13.2 Une configuration d'hôte pour l'encapsulation IPIP.

Ce n'est pas tout un réseau que vous aurez à router. Vous pouvez par exemple ne router qu'une seule adresse IP. Dans ce cas vous devrez configurer le périphérique `tunl` sur la machine 'distante' avec sa propre adresse IP et à l'extrémité A n'utiliser qu'une route hôte (avec Proxy Arp) plutôt qu'une route réseau via le périphérique tunnel. Refaisons et modifions notre configuration de manière appropriée. Maintenant nous avons seulement l'hôte '`B`' qui veut agir comme si il était à la fois connecté à l'Internet et également au réseau distant supporté par l'hôte '`A`':



Le routeur Linux 'A' sera configuré comme suit:

```
#
PATH=/sbin:/usr/sbin
#
# Ethernet configuration
ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.1.1 up
route add -host 192.168.1.12 gw fff.ggg.hhh.iii tunl0
#
# Proxy ARP for the remote host
arp -s 192.168.1.12 xx:xx:xx:xx:xx:xx pub
```

L'hôte Linux 'B' sera configuré comme suit:

```
#
PATH=/sbin:/usr/sbin
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.1.12 up
route add -net 192.168.1.0 netmask 255.255.255.0 gw aaa.bbb.ccc.ddd tunl0
```

Ce type de configuration est vraiment typique d'une application IP-Mobile, où un simple hôte veut seulement se balader sur l'Internet et maintenir une adresse IP utilisable tout le temps. Référez-vous au paragraphe Mobile-IP pour avoir plus d'informations et savoir comment faire en pratique.

6.14 IPX (AF_IPX)

Le protocole IPX est la plupart du temps utilisé dans les environnements réseaux locaux Novell NetWare(tm). Linux offre un support pour ce protocole, et peut être configuré pour agir comme extrémité réseau, ou comme routeur pour les environnements réseaux IPX.

Options de compilation du noyau:

```
Networking options  --->
    [*] The IPX protocol
    [ ] Full internal IPX network
```

Le protocole IPX et le NCPFS sont traités en détail dans le document *IPX-HOWTO* (`IPX-HOWTO.html`).

6.15 IPv6

A peine pensez-vous avoir commencé à comprendre comment fonctionne le réseau IP, les règles ont changé! IPv6 est l'abbréviation de version 6 du 'Protocole Internet' (version 6 de IP). Il fut développé initialement pour calmer les inquiétudes de la communauté Internet quant à la pénurie éventuelle d'adresses IP. Les adresses IPv6 ont 16 octets de long (128 bits). IPv6 inclut un certain nombre d'autres changements, la plupart du temps des simplifications, qui rendront les réseaux IPv6 plus facilement gérables que les réseaux IPv4.

Linux a déjà une implémentation IPv6 qui marche, mais pas encore complètement, dans la série des noyaux `2.1.*`.

Si vous voulez essayer cette prochaine génération de technologie Internet, ou si vous voulez un renseignement, lisez le document IPv6-FAQ qui se trouve sur *www.terra.net* (`http://www.terra.net/ipv6/`).

6.16 ISDN

L'Integrated Services Digital Network (ISDN)(NdT: équivalent français RNIS: Réseau Numérique à Intégration de Services) est une série de normes donnant les spécifications d'un réseau de données numériques à usage général. Un 'appel' ISDN crée un service synchrone de données point à point vers la destination. L'ISDN est généralement délivré sur une ligne à haut débit divisée en un certain nombre de canaux discrets.. Il y a deux types de canaux, les 'canaux B' qui transportent effectivement les données utilisateurs, et un canal unique appelé 'canal D' qui est utilisé pour envoyer les informations de contrôle pendant l'échange ISDN en vue d'établir des appels et autres fonctions. En Australie, par exemple, l'ISDN peut être fourni sur une liaison 2 Mps qui est divisée en 30 canaux B discrets de 64 kps et un canal D de 64 kps. N'importe quel nombre de canaux peuvent être utilisés en même temps et ceci dans toutes les combinaisons possibles. Vous pouvez par exemple établir 30 appels différents de 64 kps vers 30 destinations différentes, ou bien 15 appels de 128 kps chacun vers 15 destinations différentes (2 canaux utilisés par appel), ou seulement un petit nombre d'appels, le reste étant inactif. Un canal peut être utilisé pour des appels entrant ou sortant. Le but initial de ISDN était de permettre aux sociétés de Télécommunications de fournir un simple service de données pouvant délivrer soit le téléphone (avec une voix digitalisée) ou bien des services de données vers votre domicile ou votre bureau sans avoir à effectuer de changements pour obtenir une configuration spéciale.

Il y a plusieurs façons de connecter votre ordinateur à un service ISDN. L'une consiste à utiliser un dispositif appelé 'Adaptateur de Terminal' qui se branche sur l'unité de terminal réseau que votre opérateur de télécommunications a installé au moment de l'obtention de votre service ISDN, et qui présente des interfaces séries. L'une de ces interfaces est utilisée pour entrer les commandes pour établir les appels et la configuration, et les autres sont reliées aux périphériques réseau qui utiliseront les circuits de données quand la connexion sera faite. Linux peut travailler avec ce type de configuration sans modifications, vous devez juste traiter le port de l'adaptateur de terminal comme vous traitez tout périphériques série. Une autre façon consiste en ce que le support ISDN du noyau vous permet d'installer une carte ISDN dans votre machine Linux et le logiciel Linux prend en charge les protocoles et fait les appels lui-même.

Options de compilation noyau:

```
ISDN subsystem  --->
        <*> ISDN support
        [ ] Support synchronous PPP
        [ ] Support audio via ISDN
        < > ICN 2B and 4B support
        < > PCBIT-D support
        < > Teles/NICCY1016PC/Creatix support
```

L'implémentation Linux de ISDN supporte différents types de cartes internes ISDN. Il y a celles enumérées dans les options de configuration noyau:

- ICN 2B and 4B
- Octal PCBIT-D
- Teles ISDN-cards and compatibles

Certaines de ces cartes ont besoin de logiciels que l'on doit télécharger pour les rendre opérationnelles. Il y a un utilitaire séparé pour le faire.

Tous les détails pour configurer le support ISDN Linux se trouvent dans le répertoire `/usr/src/linux/Documentation/isdn/` et un document FAQ dédié à *isdn4linux* est disponible sur *www.lrz-muenchen.de* (`http://www.lrz-muenchen.de/ ui161ab/w` (Vous pouvez cliquer sur le drapeau anglais pour obtenir la version anglaise).

Note au sujet de PPP. L'ensemble des protocoles PPP peut travailler sur des lignes série synchrone ou asynchrone. Le démon PPP '*pppd*' couramment distribué pour Linux ne supporte que le mode asynchrone. Si vous désirez utiliser les protocoles PPP avec votre service ISDN vous aurez besoin d'une version spéciale. Les détails pour la trouver se trouvent dans la documentation mentionnée ci-dessus.

6.17 IP Masquerade

Beaucoup de gens ont une simple connexion par téléphone pour aller sur l'Internet. Presque tout le monde ne se voit offrir qu'une seule adresse IP par le founisseur d'accès avec ce type de configuration. Ceci est normalement suffisant pour permettre un accès complet au réseau. IP Masquerade est une astuce intelligente qui vous permet d'avoir plusieurs machines utilisant une seule adresse IP, en faisant croire aux autres hôtes qu'il n'y a que la machine supportant la connexion (NdT: d'où le terme masquerade=duperie, mascarade). Il y a qu'une seule mise en garde, qui est que la fonction 'masquerade' ne travaille pratiquement que dans un seul sens: les hôtes sous 'masquerade' peuvent appeler mais ne peuvent accepter ou recevoir des connexions réseau de la part d'hôtes éloignés. Cela signifie que certains services réseau comme *talk* ne peuvent fonctionner et que d'autres, comme *ftp* doivent être configurés pour fonctionner en mode passif (PASV). Heureusement la plupart des services réseau comme *telnet*, World Wide Web et *irc* fonctionnent correctement.

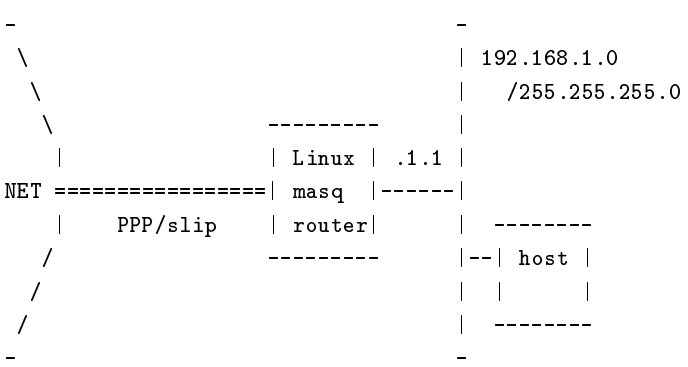
Options de compilation du noyau:

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
```

```
Networking options  --->
    [*] Network firewalls
    ....
    [*] TCP/IP networking
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: masquerading (EXPERIMENTAL)
```

Normalement votre machine Linux supportant un lien SLIP ou PPP se comportera comme si elle était toute seule. De plus elle peut avoir un autre périphérique réseau configuré, par exemple une carte Ethernet, avec des adresses réseau réservée. Les hôtes utilisant 'masquerade' seront ceux du second réseau. Chacun de ces hôtes aura l'adresse IP du port Ethernet réglée comme passerelle ou routeur par défaut.

Une configuration typique ressemble à ceci:



Les commandes adéquates pour cette configuration sont:

```
# Network route for Ethernet
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# Default route to the rest of the internet.
route add default ppp0
#
# Cause all hosts on the 192.168.1/24 network to be masqueraded.
ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0
```

Vous pouvez obtenir plus d'informations sur IP Masquerade avec *IP Masquerade Resource Page* (http://www.hwy401.com/ac

6.18 IP Transparent Proxy

IP transparent proxy est un procédé qui vous permet de rediriger des serveurs ou des services destinés à une autre machine vers les services de votre machine. Typiquement c'est utile lorsque vous avez une machine Linux routeur et qui fournit aussi un serveur proxy. Vous redirigerez toutes les connexions à ce service distant vers le serveur proxy local.

Options de compilation du noyau:

```
Code maturity level options  --->
        [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
        [*] Network firewalls
        ....
        [*] TCP/IP networking
        ....
        [*] IP: firewalling
        ....
        [*] IP: transparent proxy support (EXPERIMENTAL)
```

La configuration du dispositif transparent proxy est réalisé en utilisant la commande *ipfwadm*.

Par exemple:

```
ipfwadm -I -a accept -D 0/0 telnet -r 2323
```

Cet exemple fera en sorte que toutes les tentatives de connexion vers le port `telnet` (23), de n'importe quel hôte, seront redirigées vers le port 2323 de ce même hôte. Si vous utilisez un service sur ce port, vous pouvez forwarder des connexions telnet, les enregistrer ou exécuter tout ce qui bon vous semble.

Un exemple plus intéressant est la redirection de tout le trafic `http` au travers d'un cache local. Cependant, le protocole utlisé par les serveurs proxy diffère du protocole natif de http: quand un client se connecte à `www.server.com:80` et demande `chemin/page`, quand il se connecte au cache local il contacte `proxy.local.domain:8080` et recherche `www.server.com/chemin/page`.

Pour filtrer une demande `http` au travers du proxy local, vous devez pouvoir adapter le protocole en insérant un petit serveur, appelé `transproxy` (vous pouvez le trouver sur la toile). Vous pouvez choisir de faire tourner `transproxy` sur le port 8081, et exécuter la commande:

```
ipfwadm -I -a accept -D 0/0 80 -r 8081
```

Alors le programme `transproxy` recevra toutes les connexions devant aller vers des serveurs externes et les passera au proxy local après avoir corriger les différences de protocole.

6.19 IP Mobile

Le terme "mobilité IP" décrit la possibilité qu'un hôte a de transférer sa connexion réseau d'un point de l'Internet vers un autre sans changer d'adresse IP ou sans perdre la connectivité. Normalement quand un hôte IP change de point de connexion, il change aussi d'adresse IP. La mobilité IP résoud ce problème en allouant une adresse IP fixe à l'hôte qui se déplace et en utilisant une encapsulation IP (tunneling) avec routage automatique pour s'assurer que les datagrammes qui lui sont destinés seront routés vers l'adresse effectivement utilisée à ce moment.

Un projet est en cours en vue de fournir un paquetage complet d'outils Linux pour la mobilité IP. L'état de ce projet et les outils peuvent être obtenus sur: *Linux Mobile IP Home Page* (`http://anchor.cs.binghamton.edu/ mobileip/`)

6.20 Multicast

L'IP Multicast permet de router simultanément des datagrammes IP vers un certain nombre d'hôtes se trouvant sur des réseaux différents. Ce mécanisme est exploité pour fournir sur l'Internet des applications prenant de la bande passante, telles que les transmissions audio et video et autres nouvelles applications.

Options de compilation du noyau:

```
Networking options  --->
        [*] TCP/IP networking
        ....
        [*] IP: multicasting
```

Un ensemble d'outils et quelques modifications de la configuration réseau sont nécessaires. Une source de renseignements pour l'installation et la configuration se trouve sur *www.teksouth.com* (`http://www.teksouth.com/linux/multi`

6.21 NAT - Network Address Translation (Traduction d'adresse réseau)

Le système de traduction d'adresse réseau IP ressemble plutôt au grand frère standardisé du système IP masquerade de Linux. Il est décrit en détail dans la RFC-1631 sur votre archive RFC la plus proche. NET fournit des possibilités que IP Masquerade ne sait pas faire, ce qui le rend plus apte à une utilisation de routeur firewall pour un réseau d'entreprise et des installations de plus grandes dimensions.

Une implémentation alpha de NAT pour le noyau 2.0.29 de Linux a été développée par Michael.Hasenstein, `Michael.Hasenstein@informatik.tu-chemnitz.de`. La documentaion et l'umplémentation de Michael se trouve sur:

Linux IP Network Address Web Page (`http://www.csn.tu-chemnitz.de/HyperNews/get/linux-ip-nat.html`)

Les noyaux 2.1.* récents incluent également quelques fonctions de NAT dans l'algorithme de routage.

6.22 NetRom (`AF_NETROM`)

Les noms de périphériques NetRom sont '`nr0`', '`nr1`', etc.

Options de compilation du noyau:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    [*] Amateur Radio NET/ROM
```

Les protocoles AX25, Netrom et Rose sont décrits dans le document *AX25-HOWTO* (`AX25-HOWTO.html`). Ces protocoles sont utilisés par les radio-amateurs dans le monde entier pour l'expérimentation du packet-radio.

La plupart du travail d'implémentation a été fait par Jonathon Naylor, `jsn@cs.not.ac.uk`.

6.23 PLIP

Les noms de périphériques PLIP sont '`plip0`', '`plip1`, `plip2`.

Options de compilation du noyau:

```
Networking options  --->
    <*> PLIP (parallel port) support
```

PLIP (Parallel Line IP) est, comme SLIP, utilisé pour fournir une connexion réseau *point à point* entre deux machines, sauf qu'il est conçu pour utiliser les ports parallèles de votre machine au lieu des ports séries. Parce qu'il est possible de transmettre plus d'un bit en même temps avec un port parallèle, il est possible d'atteindre de plus hautes vitesses avec l'interface *PLIP* qu'avec une sortie série standard (un schéma de câblage est donné plus tard dans ce document). De plus, même le plus simple des ports parallèles, le port imprimante, peut être utilisé, au lieu d'acheter un UART 16550AFN relativement cher pour vos ports séries. PLIP utilise beaucoup le CPU en comparaison avec une lisaison série et ce n'est sûrement pas un bon choix si vous avez la possibilité d'avoir des cartes ethernet pas chères, mais ça fonctionne lorsque rien d'autre n'est disponible, et ça fonctionne très bien.

Les pilotes PLIP entrent en compétition avec les autres pilotes du matériel branché sur le port parallèle. Si vous voulez utiliser les deux, vous devez alors les compiler en tant que modules pour pouvoir choisir quel port vous voulez utiliser pour PLIP et quels port pour l'imprimante. Voyez *Modules-HOWTO* (`Modules-HOWTO.html`) pour plus d'informations sur la configuration des modules noyau.

Attention, notez que certains portables utilisent des circuits qui ne peuvent pas fonctionner avec PLIP car ils n'autorisent pas certaines combinaisons dont PLIP a besoin et que les imprimantes n'utilisent pas.

L'interface Linux *PLIP* est compatible avec le *Pilote PLIP Crynwyr Packet* et ceci signifie que vous pouvez connecter votre machine Linux avec une machine DOS tournant avec n'importe quel logiciel tcp/ip via *PLIP*.

Dans la série des noyaux 2.0.* les pilotes de périphérique plip sont affectés aux ports i/o et IRQ commesuit:

```
device  i/o addr    IRQ
------  --------    -----
plip0   0x3BC           5
plip1   0x378           7
plip2   0x278           2
```

Si vos ports parallèles ne correspondent pas aux combinaisons précédentes alors vous pouvez changer les IRQ en utilisant la commande *ifconfig* avec le paramètre '`irq`'. N'oubliez pas de valider les IRQ pour vos ports imprimantes dans votre ROM BIOS s'il supporte cette option.

Dans la dernière série des noyaux 2.1.* possédant le support Plug'n'Play les pilotes de périphériques sont alloués de manière séquentielle au fur et à mesure de leur détection, tout comme ceux des cartes ethernet, avec `plip0` alloué pour le premier.

Pour compiler le noyau, il n'y a qu'un seul fichier à regarder pour confgurer *plip*. Il s'agit de `/usr/src/linux/driver/net/CONF` qui contient les temporisations de *plip* en millisecondes. Les valeurs par défaut conviennent dans la plupart des cas. Vous aurez peut-être à les augmenter si vous avez un ordinateur particulièrement lent, auquel cas les temporisations à augmenter se trouvent sur **l'autre** ordinateur. Il existe un programme appelé *plipconfig*

vous permettant de changer ces temporisations sans recompiler le noyau. Il est fourni avec de nombreuses distributions Linux.

Pour configurer une interface *plip*, vous devez **ajouter** les lignes suivantes dans votre fichier réseau `rc`:

```
#
# Attach a PLIP interface
#
#  configure first parallel port as a plip device
/sbin/ifconfig plip0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# End plip
```

Où:

IPA.IPA.IPA.IPA

represente votre adresse IP.

IPR.IPR.IPR.IPR

represente l'adresse l'adresse IP de la machine distante.

Le paramètre *pointopoint* a la même signification que pour SLIP, c'est à dire qu'il spécifie l'adresse de la machine de la machine à l'autre bout de la liaison.

Dans la plupart des cas vous pouvez traiter l'interface *plip* comme si elle était une interface *SLIP*, sauf que ni *dip* ni *slattach* ne doivent, ou ne peuvent, être utilisés.

Plus d'information sur PLIP peut être obtenu avec *PLIP-mini-HOWTO* (`mini/PLIP`)

6.24 PPP

Les noms de périphériques PPP sont '`ppp0`', '`ppp1`, etc. Les noms sont attribués séquentiellement, le premier périphérique étant '`0`'.

Options de compilation du noyau:

```
Networking options  --->
    <*> PPP (point-to-point) support
```

La configuration de PPP est discutée en détail dans le *PPP-HOWTO* (`PPP-HOWTO.html`).

6.24.1 Maintenance d'une connexion permanente avec le réseau à l'aide de *pppd*.

Si vous êtes suffisamment fortunés pour avoir une connexion semi-permanente avec le net et que vous vouliez que votre machine refasse la connexion PPP en cas de déconnexion, alors voici une astuce simple.

Configurer PPP de sorte qu'il soit démarré par l'utilisateur `root` en faisant la commande:

```
# pppd
```

Soyez certains d'avoir l'option '`-detach`' dans le fichier `/etc/ppp/options`. Puis, insérez la ligne suivante dans votre fichier `/etc/inittab`, avec les définitions des *getty*:

```
pd:23:respawn:/usr/sbin/pppd
```

Cela permettra au programme *init* de démarrer et de surveiller le programme *pppd* , et de le redémarrer automatiquement s'il meurt.

6.25 Protocole Rose (`AF_ROSE`)

Les noms de périphériques Rose sont '`rs0`', '`rs1`', etc. . Rose est disponible dans la série des noyaux `2.1.*`.

Options de compilation du noyau:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    <*> Amateur Radio X.25 PLP (Rose)
```

Les protocoles AX25, Netrom et Rose sont expliqués dans le *AX25-HOWTO* (`AX25-HOWTO.html`). Ces protocoles sont utilisés par les opérateurs radio-amateur du monde entier pour l'expérimentation du packet-radio.

La plupart du travail d'implémentation de ces protocoles a été réalisé par Jonathon Naylor, `jsn@cs.not.ac.uk`.

6.26 Support SAMBA - 'NetBEUI', 'NetBios'.

SAMBA est une implémentation du protocole Session Management Block. Samba permet à Microsoft et autres systèmes de monter et d'utiliser vos disques et imprimantes.

SAMBA et sa configuration sont décrits en détail dans le *SMB-HOWTO* (`SMB-HOWTO.html`).

6.27 Client SLIP

Les fichiers de périphériques SLIP sont nommés '`sl0`', '`sl1`' etc. le premier configuré étant '`0`' et les autres s'incrémentant au fur et à mesure de leur configuration.

Options de compilation du noyau:

```
Network device support  --->
    [*] Network device support
    <*> SLIP (serial line) support
    [ ]  CSLIP compressed headers
    [ ]  Keepalive and linefill
    [ ]  Six bit SLIP encapsulation
```

SLIP (Serial Line Internet Protocol) vous permet d'utiliser tcp/ip avec une ligne série, ce peut être un téléphone et un modem, ou tout autre ligne dédiée. Bien sûr pour utiliser SLIP vous devez avoir accès à un

serveur SLIP dans votre entourage. Beaucoup d'universités et de sociétés fournissent des accès SLIP de par le monde.

SLIP utilise les ports séries de votre machine pour transporter les datagrammes IP. Pour cela il doit prendre le contrôle du périphérique série. Les noms de périphériques SLIP sont *sl0*, *sl1* etc. Comment ceux-ci correspondent avec vos périphériques série? Le code réseau utilise ce que l'on nomme un appel *ioctl* (i/o control) pour transformer les périphériques série en périphériques SLIP. Il y a deux programmes qui peuvent faire cela, ce sont *dip* et *slattach*

6.27.1 dip

dip (Dialup IP) est un programme élégant capable de régler la vitesse du dispositif série, de demander à votre modem d'appeler l'autre extrémité de la ligne, de vous connecter automatiquement au serveur distant, de chercher des messages qui vous ont été envoyés par le serveur et d'en extraire des informations telles que votre adresse IP et de faire le *ioctl* nécessaire pour basculer votre port série en mode SLIP. *dip* est très flexible quant à l'utilisation de scripts et grâce à ceci vous pouvez automatiser vos procédures de connexion.

On peut le trouver sur: *sunsite.unc.edu* (`ftp://sunsite.unc.edu/pub/Linux/system/Network/serial/dip/dip337o-uri.`

Pour l'installer faites:

```
#
# cd /usr/src
# gzip -dc dip337o-uri.tgz | tar xvf -
# cd dip-3.3.7o

<editez Makefile>

# make install
#
```

Le fichier `Makefile` suppose l'existence d'un groupe nommé *uucp*, mais vous pouvez le changer en *dip* ou *SLIP*, selon votre configuration.

6.27.2 slattach

slattach au contraire de *dip* est un programme très simple, très facile à utiliser, mais qui n'a pas la sophistication de *dip*. Il n'a pas la possiblité d'accepter des scripts, tout ce qu'il fait étant de configurer votre périphérique série en périphérique SLIP. Il suppose que vous avez toutes les informations nécessaires et que la liaison série est établie avant de l'invoquer. *slattach* est idéal quand vous avez une liaison permanente avec votre serveur, comme un câble physique ou une ligne dédiée.

6.27.3 Quand utiliser quoi ?

Vous devriez utiliser *dip* lorsque votre liaison vers la machine qui est votre serveur SLIP est un modem, ou tout autre lien intermittent. Vous devriez utiliser *slattach* quand vous avez une ligne dédiée, peut-être un

câble, entre votre machine et le serveur et qu'il n'y a pas d'action spéciale nécessaire pour garder la ligne en activité. Voir la section 'Connexion SLIP permanente' pour plus de détails.

Configurer SLIP est analogue à la configuration d'une interface Ethernet (voir la section 'Configurer un périphérique Ethernet' ci-dessus). Cependant, il existe quelques différences.

Tout d'abord, les liens SLIP ne sont pas des réseaux Ethernet en ce sens qu'il n'y a que deux hôtes sur le réseau, un à chaque extrémité de la liaison. A la différence de l'Ethernet qui est disponible dès que vous êtes câblé, avec SLIP, en fonction du type de lien que vous avez, vous pouvez avoir à initialiser votre connexion réseau d'une manière spéciale.

Si vous utilisez *dip*, alors cela ne sera pas fait au moment du démarrage de la machine, mais plus tard, quand vous serez prêt pour utiliser la liaison. Il est possible d'automatiser la procédure. Si vous utilisez *slattach* vous voudrez probablement ajouter une section dans votre fichier *rc.inet1*. Ceci sera décrit bientôt.

Il y a deux types principaux de serveurs SLIP: serveurs avec adressage IP dynamique et serveurs avec adressage IP statique. Presque tous les serveurs SLIP vous demanderont à la connexion d'utiliser un nom d'utilisateur et un mot de passe quand vous composez le numéro. *dip* peut prendre en charge la connexion automatiquement.

6.27.4 Serveur SLIP statique avec une ligne téléphonique et DIP.

Le serveur SLIP statique est celui qui vous fournit une adresse IP qui reste exclusivement la vôtre. A chaque fois que vous vous connectez à ce serveur, vous configurez votre port SLIP avec cette adresse. Le serveur SLIP statique répond à votre appel par modem, vous demande probablement un nom d'utilisateur et un mot de passe, et ensuite dirige tous les datagrammes destinés à votre adresse au travers de cette connexion. Si vous avez un serveur statique, alors vous mettez des entrées pour votre nom d'hôte et votre adresse IP (puisque vous savez ce qu'elle sera) dans votre fichier `/etc/hosts`. Vous avez aussi à configurer d'autres fichiers comme: `rc.inet2`, `host.conf`, `resolv.conf`, `/etc/HOSTNAME` et `rc.local`. N'oubliez pas qu'en configurant `rc.inet1`, vous n'avez pas besoin d'ajouter de commandes spéciales pendant la connexion SLIP puisque c'est *dip* qui fait tout le dur labeur à votre place en configurant votre interface. Vous avez besoin de donner à *dip* les informations adéquates et il configure l'interface pour vous après avoir demandé au modem d'établir l'appel et de vous connecter au serveur.

Si votre serveur SLIP fonctionne comme cela alors vous pouvez directement aller à la section 'Utiliser Dip' pour apprendre à configurer *dip* convenablement.

6.27.5 Serveur SLIP dynamique avec une ligne téléphonique et DIP.

Le serveur SLIP *dynamique* vous alloue une adresse IP de manière aléatoire, à partir d'un groupe d'adresses, à chaque fois que vous vous connectez. Cela signifie qu'il n'y a aucune garantie d'avoir la même adresse à chaque fois, et que celle-ci peut être utilisée par quelqu'un d'autre après la déconnexion. L'administrateur réseau qui a configuré le serveur SLIP a assigné un groupe d'adresses que le serveur SLIP peut utiliser quand il reçoit un appel entrant. Il prend alors la première adresse inutilisée, guide l'appelant au travers du processus de connexion et envoie un message de bienvenue contenant l'adresse IP qu'il a allouée et continue d'utiliser cette adresse tout le temps de l'appel.

Configurer ce type de serveur revient à configurer un serveur statique, sauf que vous devez ajouter une étape pour obtenir l'adresse IP que le allouée par le serveur puis configurer le périphérique SLIP avec celle-ci.

Encore une fois, *dip* fait le sale boulot et les nouvelles versions sont suffisamment élégantes pour non seulement établir la connexion, mais aussi pour lire l'adresse IP inscrite dans le message de bienvenue et la stocker de telle sorte que vous puissiez configurer votre périphérique SLIP avec.

Si votre serveur SLIP fonctionne ainsi, alors vous pouvez aller à la section 'Utiliser DIP' pour savoir comment configurer *dip* de manière adéquate.

6.27.6 Utiliser DIP.

Comme expliqué plus haut, *dip* est un programme puissant qui simplifie et automatise le processus de composition d'un numéro vers un serveur SLIP, se connecte dessus, démarre la connexion et configure les périphériques SLIP à l'aide des commandes *ifconfig* et *route* appropriées.

Essentiellement, pour utiliser *dip* vous écrivez un 'script dip' qui est tout simplement une liste de commandes que *dip* comprend et qui lui dit comment réaliser chacune des actions que vous voulez qu'il fasse. Voyez le fichier `sample.dip` fourni avec *dip* pour avoir une idée de la manière dont il travaille. *dip* est vraiment un programme puissant, avec beaucoup d'options. Au lieu de regarder chacune d'elles, il vaut mieux jeter un coup d'oeil dans la page de manuel, le fichier README et les fichiers d'exemple qui sont fournis avec votre version de *dip*.

Vous pouvez noter que le script `sample.dip` suppose que vous utilisez un serveur SLIP statique, aussi vous connaissez votre adresse IP à l'avance. Pour les serveurs SLIP dynamiques, les nouvelles versions de *dip* incluent une commande que vous pouvez utiliser pour lire et configurer automatiquement votre périphérique SLIP avec l'adresse IP donnée par le serveur dynamique. L'exemple suivant est une version modifiée du fichier `sample.dip` fourni avec *dip337j-uri.tgz* et qui est probablement un bon point de départ pour vous. Vous pouvez le sauvegarder sous le nom de `/etc/dipscript` et l'éditer pour l'adapter à votre configuration:

```
#
# sample.dip    Dialup IP connection support program.
#
#               This file (should show) shows how to use the DIP
#       This file should work for Annex type dynamic servers, if you
#       use a static address server then use the sample.dip file that
#       comes as part of the dip337-uri.tgz package.
#
#
# Version:      @(#)sample.dip  1.40    07/20/93
#
# Author:       Fred N. van Kempen, <waltje@uWalt.NL.Mugnet.ORG>
#

main:
# Next, set up the other side's name and address.
# My dialin machine is called 'xs4all.hacktic.nl' (== 193.78.33.42)
get $remote xs4all.hacktic.nl
```

```
# Set netmask on sl0 to 255.255.255.0
netmask 255.255.255.0
# Set the desired serial port and speed.
port cua02
speed 38400

# Reset the modem and terminal line.
# This seems to cause trouble for some people!
reset

# Note! "Standard" pre-defined "errlevel" values:
#  0 - OK
#  1 - CONNECT
#  2 - ERROR
#
# You can change those grep'ping for "addchat()" in *.c...

# Prepare for dialing.
send ATQ0V1E1X4\r
wait OK 2
if $errlvl != 0 goto modem_trouble
dial 555-1234567
if $errlvl != 1 goto modem_trouble

# We are connected.  Login to the system.
login:
sleep 2
wait ogin: 20
if $errlvl != 0 goto login_trouble
send MYLOGIN\n
wait ord: 20
if $errlvl != 0 goto password_error
send MYPASSWD\n
loggedin:

# We are now logged in.
wait SOMEPROMPT 30
if $errlvl != 0 goto prompt_error

# Command the server into SLIP mode
send SLIP\n
wait SLIP 30
if $errlvl != 0 goto prompt_error

# Get and Set your IP address from the server.
#   Here we assume that after commanding the SLIP server into SLIP
#   mode that it prints your IP address
get $locip remote 30
```

```
if $errlvl != 0 goto prompt_error

# Set up the SLIP operating parameters.
get $mtu 296
# Ensure "route add -net default xs4all.hacktic.nl" will be done
default

# Say hello and fire up!
done:
print CONNECTED $locip ---> $rmtip
mode CSLIP
goto exit

prompt_error:
print TIME-OUT waiting for sliplogin to fire up...
goto error

login_trouble:
print Trouble waiting for the Login: prompt...
goto error

password:error:
print Trouble waiting for the Password: prompt...
goto error

modem_trouble:
print Trouble occurred with the modem...
error:
print CONNECT FAILED to $remote
quit

exit:
exit
```

L'exemple précédent suppose que vous appeliez un serveur SLIP *dynamique*; si vous appelez un serveur SLIP *statique*, alors le fichier `sample.dip` founi avec *dip337j-uri.tgz* devrait vous convenir.

Quand on donne à *dip* la commande *get $local*, il cherche dans le texte venant de l'extrémité de la ligne une chaîne de caractères ressemblant à une adresse IP, c'est à dire des ensembles de nombres séparés par des caractères '.'. Cette modification fut mise en place plus spécialement pour les serveurs SLIP *dynamiques*, afin que le processus de lecture de l'adresse IP fournie par le serveur soit automatisé.

L'exemple ci-dessus crée automatiquement une route par défaut via votre liaison SLIP, et si ce n'est pas ce que vous voulez, car vous avez une connexion Ethernet qui devrait être votre route par défaut, alors enlevez la commande *default* du script. Après que le script ait fini de tourner, tapez la commande *ifconfig*, et vous verrez que vous avez un périphérique *sl0*. C'est votre périphérique SLIP. Si le besoin s'en fait sentir, vous pouvez modifier manuellement sa configuration, après que la commande *dip* soit finie, en utilisant les commandes *ifconfig* et *route*.

Notez que *dip* vous permet de choisir parmi différents protocoles en utilisant la commande `mode`, l'exemple le plus courant étant *cSLIP* pour utiliser SLIP avec compression. Notez encore que les deux extrémités de la liaison doivent être d'accord, aussi assurez-vous que ce que vous avez choisi est en accord avec les réglages du serveur.

L'exemple montré ci-dessus est plutôt robuste et devrait faire face à la plupart des erreurs. Référez-vous à la page de manuel de *dip* pour plus d'informations. Naturellement, vous pouvez, par exemple, modifier le script pour réaliser des choses comme recomposer le numéro vers le serveur si la connexion n'a pas été faite au bout d'un certain temps, ou même essayer une série de serveurs si vous avez accès à plus d'un.

6.27.7 Connexion permanente SLIP utilisant une ligne et slattach.

Si vous avez deux machines reliées par un câble, ou si vous êtes suffisamment riche pour avoir une ligne dédiée, ou un autre type de connexion permanente entre votre machine et une autre, alors vous n'avez pas besoin de vous casser la tête avec *dip* pour régler votre liaison série. *slattach* est un utilitaire très simple à utiliser et vous permet d'avoir les fonctionnalités juste nécessaires pour configurer votre connexion.

Puisque votre connexion est permanente, vous ajoutez quelques commandes dans votre fichier `rc.inet1`. Tout ce dont vous avez besoin pour une connexion permanente est de vous assurer que vous avez configuré votre périphérique série à la bonne vitesse et basculer votre périphérique série en mode SLIP. *slattach* vous permet de faire ceci avec une seule commande. Ajoutez ce qui suit à votre fichier `rc.inet1`:

```
#
# Attach a leased line static SLIP connection
#
#  configure /dev/cua0 for 19.2kbps and cslip
/sbin/slattach -p cslip -s 19200 /dev/cua0 &
/sbin/ifconfig sl0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# End static SLIP.
```

Où:

IPA.IPA.IPA.IPA

représente votre adresse IP.

IPR.IPR.IPR.IPR

représente l'adresse IP de l'hôte distant.

slattach alloue le premier périphérique SLIP disponible au périphérique série spécifié. *slattach* démarre avec *sl0*. Par conséquent la première commande *slattach* relie le périphérique *sl0* au périphérique spécifé, puis *sl1* la fois suivante, etc.

slattach vous permet de configurer un certain nombre de protocoles grâce à l'argument `-p`. Dans votre cas vous utilisez soit *SLIP* soit *cSLIP* suivant que vous voulez utiliser la compression ou non. Note: les deux extrémités doivent être d'accord sur l'utilisation de la compression.

6.28 Serveur SLIP.

Vous avez peut-être une machine connectée au réseau et vous aimeriez que d'autres personnes puisse s'y connecter pour y chercher des services de réseau, alors vous devez configurer votre machine comme serveur. Si vous voulez utiliser SLIP comme protocole de ligne série, vous avez trois possiblités pour configurer votre machine Linux comme serveur SLIP. Ma préférence est la première présentée, *sliplogin*, car elle semble la plus facile à configurer et à comprendre, mais je présenterai un résumé pour chacune, ainsi vous pourrez vous décider vous-mêmes.

6.28.1 Serveur SLIP utilisant *sliplogin*.

sliplogin est un programme que vous pouvez utiliser à la place du shell normal de connexion pour les utilisateurs SLIP, et qui convertit la ligne terminal en ligne SLIP. Il vous permet de configurer votre machine Linux soit en *serveur à adresse statique* (les utilisateurs obtiennent toujours la même adresse à chaque connexion), ou bien en *serveur à adresse dynamique* (les utilisateurs obtiennent une adresse allouée qui n'est pas forcément la même que lors de la connexion précédente).

L'appelant se connecte comme pour une connexion standard, en donnant son nom d'utilisateur et son mot de passe, mais au lieu d'avoir une invite de shell après la connexion, *sliplogin* est exécuté et cherche dans son fichier de configuration une entrée dont le nom correspond à celui de l'hôte appelant. S'il en détecte une, il configure la ligne comme une ligne avec 8 bits de données, et utilise un appel *ioctl* pour convertir la ligne en ligne SLIP. Quand ce processus est fini, la dernière étape de la configuration prend place, *sliplogin* invoquant un script qui configure l'interface SLIP avec l'adresse IP adéquate, ainsi que le masque de réseau et positionne le routage approprié. Ce script est appelé habituellement `/etc/slip.login`, mais tout comme *getty*, si vous avez certains appelants demandant une initialisation spéciale, alors vous pouvez créer des scripts de configuration appelés `/etc/slip.login.loginname` qui seront utilisés à la place du script par défaut.

Il y a trois ou quatre fichiers que vous devez configurer pour que *sliplogin* travaille pour vous. Je décrirai comment et où obtenir les logiciels et comment chacun est configuré. Ces fichiers sont:

- `/etc/passwd`, pour l'acceptation des utilisateurs entrant.
- `/etc/slip.hosts`, qui contient une information unique pour chaque utilisateur entrant.
- `/etc/slip.login`, qui s'occupe de la configuration du routage nécessaire pour l'utilisateur.
- `/etc/slip.tty`, requis uniquement si vous configurez votre serveur avec *allocation d'adresse dynamique*: il contient une table des adresses à allouer.
- `/etc/slip.logout`, qui contient les commandes de 'nettoyage' après la déconnexion de l'utilisateur ou bien une déconnexion intempestive.

Où obtenir *sliplogin* Vous avez peut-être déjà le paquetage dans votre distribution; si ce n'est pas le cas alors *sliplogin* peut être obtenu sur *sunsite.unc.edu* (`ftp://sunsite.unc.edu/pub/linux/system/Network/serial/sliplog`
Le fichier tar contient à la fois les sources, les binaires précompilés et une page de manuel.

Pour s'assurer que seuls les utilisateurs autorisés pourront faire tourner le programme *sliplogin*, vous devez ajouter une entrée dans votre fichier `/etc/group` similaire à la suivante:

```
 ..
slip::13:radio,fred
 ..
```

Lorsque vous installez le paquetage *sliplogin* package, `Makefile` change le groupe du programme *sliplogin* en `slip`, et cela signifie que seuls les utilsateurs qui appartiennent à ce groupe pourront l'exécuter. L'exemple donné ci-dessus ne permet qu'aux utilisateurs `radio` et `fred` de pouvoir faire tourner le programme *sliplogin*.

Pour installer les binaires dans le répertoire `/sbin` et les pages de manuel dans la section 8, faites:

```
# cd /usr/src
# gzip -dc .../sliplogin-2.1.1.tar.gz | tar xvf -
# cd sliplogin-2.1.1
# <..edit the Makefile if you don't use shadow passwords..>
# make install
```

Si vous voulez recompiler les binaires avant de les installer, faites `make clean` avant de faire `make install`. Si vous voulez installer les binaires autre part, vous devez éditer le fichier `Makefile` et le modifier en conséquence.

Lisez les fichiers `README` qui sont inclus dans le paquetage pour plus d'informations.

Configurer `/etc/passwd` pour les hôtes SLIP. Normalement vous devez créer des noms d'hôtes spéciaux, pour ceux qui appellent avec SLIP, dans votre fichier `/etc/passwd`. Une convention souvent suivie est d'utiliser le *nom d'hôte* de l'hôte appelant avec la lettre capitale 'S' comme préfixe. Ainsi, par exemple, si l'hôte appelant s'appelle `radio` alors vous pouvez créer une entrée dans le fichier `/etc/passwd` qui ressemble à ceci:

```
Sradio:FvKurok73:1427:1:radio SLIP login:/tmp:/sbin/sliplogin
```

Le nom du compte n'a pas réellement d'importance, du moment qu'il ait une signification pour vous.

Note: l'appelant n'a pas besoin de répertoire home spécial car il n'utilisera pas de shell le la machine, dès lors `/tmp` est un bon choix. Notez bien que *sliplogin* est utilisé à la place du shell de connexion normal.

Configurer `/etc/slip.hosts` Le fichier `/etc/slip.hosts` est le fichier où *sliplogin* cherche les entrées correspondant au nom de connexion pour obtenir les détails de configuration de cet hôte. C'est le fichier où vous indiquez l'adresse IP et le masque de réseau qui seront assignés à l'appelant et configurés pour leur usage. Des exemples d'entrées pour deux hôtes, une statique pour l'hôte `radio` et l'autre dynamique pour l'hôte `albert` ressemblent à ceci:

```
#
Sradio   44.136.8.99   44.136.8.100  255.255.255.0  normal      -1
Salbert  44.136.8.99   DYNAMIC       255.255.255.0  compressed  60
#
```

Les entrées du fichier `/etc/slip.hosts` sont:

1. Le nom de connexion de l'appelant.
2. L'adesse IP de la machine serveur, donc cette machine.
3. L'adresse IP que l'appelant aura. Si c'est marqué `DYNAMIC` alors l'adresse IP sera allouée suivant les informations contenues dans le fichier `/etc/slip.tty` décrit plus tard. **Note:** vous devez utiliser au moins la version 1.3 de sliplogin pour que cela fonctionne.
4. Le masque de réseau assigné à la machine appelante, en notation décimale, par exemple 255.255.255.0 pour un masque de réseau de classe C.
5. Un réglage du mode SLIP qui active/désactive la compression. Les valeurs autorisées sont "`normal`" et "`compressed`".
6. Un paramètre de délai qui spécifie combien de temps la ligne peut rester inactive (aucun datagramme reçu) avant une déconnexion automatique. Une valeur négative désactive cette possiblité.
7. arguments optionnels.

Note: Vous pouvez mettre soit les noms d'hôtes soit les adresses IP en notation décimale pointée pour les champs 2 et3. Si vous utilisez les noms d'hôtes, alors ces hôtes doivent être résolvables, c'est à dire que votre machine est capable de déterminer une adresse IP pour ces noms d'hôtes, autrement le script échouera pendant l'appel. Vous pouvez le tester en faisant telnet vers un nom d'hôte: si vous obtenez le message '*Trying nnn.nnn.nnn...*' alors votre machine est capable de trouver une adresse ip pour ce nom d'hôte. Si vous obtenez le message '*Unknown host*', alors il n'en a pas. Dans ce cas essayez d'utiliser l'adress ip en notation décimale pointée; ou bien voyez du côté de votre configuration de résolveur de nom (voir la section `Résolution de nom`).

Les modes les plus courants de SLIP sont:

normal

mode SLIP normal non compressé.

compressed

mode avec compression van Jacobsen des en-têtes (cSLIP)

Bien sûr ils sont mutuellement exclusifs, vous devez utiliser l'un ou l'autre. Pour plus d'informations sur les options disponibles, voir les pages de manuels.

Configurer le fichier `/etc/slip.login`. Après que *sliplogin* ait exploré le fichier `/etc/slip.hosts` et ait trouvé une entrée qui convient, il essaye d'exécuter le fichier `/etc/slip.login` pour effectivement configurer l'interface SLIP avec son adresse ip et son masque de réseau.

L'exemple de fichier `/etc/slip.login` fourni avec le paquetage *sliplogin* ressemble à ceci:

```
#!/bin/sh -
```

```
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90
#
# generic login file for a SLIP line.  sliplogin invokes this with
# the parameters:
#     $1       $2       $3    $4, $5, $6 ...
#   SLIPunit ttyspeed   pid   the arguments from the slip.host entry
#
/sbin/ifconfig $1 $5 pointopoint $6 mtu 1500 -trailers up
/sbin/route add $6
arp -s $6 <hw_addr> pub
exit 0
#
```

Notez que ce script utilise seulement les commandes *ifconfig* et *route* pour configurer le périphérique SLIP avec sa propre adresse ip, l'adresse ip de l'hôte distant , le masque de réseau puis crée une route vers l'adresse distante via le périphérique SLIP. C'est à dire la même chose que si vous utilisiez la commande *slattach*.

Notez aussi l'utilisation de *Proxy ARP* pour s'assurer que les hôtes placés sur le même Ethernet que la machine serveur sauront comment atteindre l'hôte qui s'est connecté. Le champ `<hw_addr>` doit être l'adresse matérielle de la carte Ethernet de la machine. Si votre machine serveur n'est pas sur un réseau Ethernet, vous pouvez ignorer cette ligne.

Configurer le fichier `/etc/slip.logout`. Quand la connexion s'est arrêtée, assurez-vous que le périphérique série est revenu à son état normal de telle sorte que les futurs appelants puissent se connecter correctement. Ceci est accompli en utilisant le fichier `/etc/slip.logout`. Il est de format très simple et est appelé avec le même argument que le fichier `/etc/slip.login`.

```
#!/bin/sh -
#
#               slip.logout
#
/sbin/ifconfig $1 down
arp -d $6
exit 0
#
```

Tout ce qu'il fait est de 'mettre à zéro' l'interface qui supprimera la route précédemment créée. Il utilise aussi la commande *arp* pour supprimer tout arp proxy en place, encore une fois vous n'avez pas besoin de la commande *arp* dans le script si votre machine serveur ne possède pas de port Ethernet.

Configurer le fichier `/etc/slip.tty`. Si vous utilisez une allocation d'adresse ip dynamique (tous les hôtes configurés avec le mot-clé `DYNAMIC` dans le fichier `/etc/slip.hosts`) alors vous devez configurer le fichier `/etc/slip.tty` pour lister les adresses qui seront assignées aux ports. Vous n'aurez besoin de ce fichier que si vous voulez que votre serveur alloue des adresses aux utilisateurs de manière dynamique.

Ce fichier est un tableau qui liste les périphériques *tty* supportant les connexions SLIP entrantes et l'adresse ip qui sera assignée aux utilisateurs se connectant à ce port.

Son format est le suivant:

```
# slip.tty    tty -> IP address mappings for dynamic SLIP
# format: /dev/tty?? xxx.xxx.xxx.xxx
#
/dev/ttyS0      192.168.0.100
/dev/ttyS1      192.168.0.101
#
```

Ce que dit ce tableau est que les appelants qui se connectent sur le port `/dev/ttyS0` et qui ont leur champ d'adresse dans le fichier `/etc/slip.hosts` réglé sur `DYNAMIC` auront l'adresse `192.168.0.100`.

De cette manière vous n'avez besoin d'allouer qu'une seule adresse par port pour tous les utilisateurs n'ayant pas besoin d'adresse fixe. Ceci vous permet d'avoir le nombre minimum d'adresses nécessaires pour éviter du gaspillage.

6.28.2 Serveur Slip utilisant *dip*.

Tout d'abord laissez-moi dire que certaines informations ci-dessous viennent des pages de manuel de *dip*, où la manière de faire tourner Linux comme serveur SLIP est brièvement décrite. Faites attention aussi que ce qui suit est basé sur le paquetage *dip337o-uri.tgz* et ne s'applique vraisemblablement pas à d'autres versions de *dip*.

dip possède un mode de travail des données d'entrée qui permet de localiser automatiquement un utilisateur entrant et qui configure la ligne série comme lien SLIP suivant les informations trouvées dans le fichier `/etc/diphosts`. Ce mode est activé en invoquant *dip* avec *diplogin*. Voilà donc comment utiliser *dip* comme serveur SLIP, en créant des comptes spéciaux où *diplogin* est utilisé comme shell de connexion.

La première chose à faire est de créer un lien symbolique comme suit:

```
# ln -sf /usr/sbin/dip /usr/sbin/diplogin
```

Ensuite vous devez ajouter des entrées à la fois dans vos fichiers `/etc/passwd` et `/etc/diphosts`. Les entrées que vous devez y mettre sont formatées comme suit:

Pour configurer Linux comme serveur SLIP avec *dip*, vous devez créer quelques comptes SLIP spéciaux pour les utilisateurs, où *dip* (en mode d'entrée) est utilsé comme shell de connexion. Une convention suggérée est d'avoir tous les comptes SLIP commençant avec la lettre 'S' majuscule, par exemple 'Sfredm'.

Un exemple d'entrée dans `/etc/passwd` pour un utilisateur SLIP ressemble à ceci:

```
Sfredm:ij/SMxiTlGVCo:1004:10:Fred:/tmp:/usr/sbin/diplogin
^^          ^^          ^^  ^^   ^^    ^^    ^^
|           |           |   |    |     |     \__ diplogin comme shell de connexion
|           |           |   |    |     \_______ Repertoire personnel
|           |           |   |    \_____________ Nom complet d'utilisateur
```

```
|          |          |   \_________________ GID
|          |          \_____________________ UID
|          \_______________________________ Mot de passe crypte
\_________________________________________ Nom de connexion Slip
```

Après la connexion de l'utilisateur, le programme *login* (s'il trouve et accepte l'utilisateur) exécute la commande *diplogin*. *dip*, lorsqu'il est invoqué en tant que *diplogin* sait qu'il sera automatiquement utilisé comme shell de connexion. Quand il est démarré comme *diplogin* la première chose qu'il fait est d'utiliser l'appel de la fonction *getuid()* pour obtenir l'identificateur de l'utilisateur appelant. Il regarde ensuite dans le fichier `/etc/diphosts` pour trouver la première entrée qui correspond soit à l'identificateur soit au nom du périphérique *tty* où l'appel est entré et se configure lui-même de manière appropriée. Par un choix judicieux: soit de donner à l'utilisateur une entrée dans le fichier `diphosts`, ou soit de laisser à l'utilisateur la configuration par défaut, vous pouvez construire votre serveur de telle manière que vous puissiez avoir un mélange d'utilisateurs ayant des adresses allouées statiquement ou dynamiquement.

dip ajoutera automatiquement une entrée 'Proxy-ARP' si elle est invoquée en mode d'entrée, aussi vous n'avez pas à vous soucier d'ajouter de telles entrées manuellement.

Configurer `/etc/diphosts` `/etc/diphosts` est utilisé par *dip* pour examiner des configurations préétablies concernant des hôtes éloignés. Ceux-ci peuvent être des hôtes se connectant sur votre machine, ou bien des machines sur lesquelles vous vous connectez.

Le format général de `/etc/diphosts` est:

```
 ..
Suwalt::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
ttyS1::145.71.34.3:145.71.34.2:255.255.255.0:Dynamic ttyS1:CSLIP,296
 ..
```

Les champs sont:

1. `nom de connexion`: comme retourné par getpwuid(getuid()) ou bien le nom de tty.
2. `inutilisé`: pour compatibilité avec passwd
3. `Adresse distante`: adresse IP de l'appelant, soit numérique soit nominative
4. `Adresse locale`: adresse IP de cette machine, soit numérique soit nominative.
5. `Masque de réseau`: en notation décimale pointée
6. `Commentaires`: vous y mettez ce que vous voulez.
7. `protocole`: Slip, CSlip etc.
8. `MTU`: nombre décimal

Un exemple d'entrée `/etc/net/diphosts` pour un hôte distant peut être:

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:SLIP,296
```

qui spécifie une liaison SLIP avec une adresse distante de 145.71.34.1 et un MTU de 296, ou:

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
```

qui spécifie une liaison compatible cSLIP avec une adresse distante de 145.71.34.1 et un MTU de 1006.

Dès lors, tous les utilisateurs à qui vous permettez d'avoir une connexion avec allocation d'adresse IP statique auront une entrée dans `/etc/diphosts`. Si vous voulez que des utilisateurs qui appellent sur un port particulier aient leur adresse allouée dynamiquement, vous devez avoir une entrée pour le périphérique `tty`, mais pas d'entrée pour l'utilisateur lui-même. Vous devez vous souvenir de configurer au moins une entrée pour chaque périphérique `tty` que vos utilisateurs entrants utiliseront pour être sûrs qu'une configuration adéquate soit disponible, indépendamment du modem sur lequel ils se connectent.

Quand un utilisateur se connecte, il recevra une invite normal de login et une demande de mot de passe, pour lesquels il devra entrer son identificateur SLIP et son mot de passe. Si tout est correct, l'utilisateur ne verra pas de message spécial, il devra juste basculer en mode SLIP chez lui et ensuite il pourra se connecter et être configuré avec les paramètres contenus dans le fichier `diphosts`.

6.28.3 Serveur SLIP utilisant l'ensemble *dSLIP*.

Matt Dillon <`dillon@apollo.west.oic.com`> a écrit un paquetage qui permet des liaisons SLIP non seulement entrantes mais aussi sortantes. Le paquetage de Matt est une combinaison de petits programmes et de scripts qui prennent en charge les connexions à votre place. Vous aurez besoin de *tcsh* car au moins l'un des scripts en a besoin. Matt fournit une copie binaire de l'utilitaire *expect* car il est aussi nécessaire pour l'un des scripts. Il serait préférable d'avoir une certaine expérience de *expect* pour que ce paquetage soit utile pour vous, mais que cela ne vous décourage pas.

Matt a écrit une bonne procédure d'installation dans le fichier README, aussi je ne me fatiguerai pas à la répéter.

Vous pouvez récupérer le paquetage *dSLIP* sur son site d'origine:

apollo.west.oic.com

```
/pub/linux/dillon_src/dSLIP203.tgz
```

ou bien sur:

sunsite.unc.edu

```
/pub/Linux/system/Network/serial/dSLIP203.tgz
```

Lisez le fichier `README` et créez les entrées `/etc/passwd` et `/etc/group` **avant** de faire `make install`.

6.29 Support STRIP (Starmode Radio IP)

Les noms de périphériques STRIP sont '`st0`', '`st1`', etc.

Options de compilation du noyau:

```
Network device support  --->
        [*] Network device support
        ....
        [*] Radio network interfaces
        < > STRIP (Metricom starmode radio IP)
```

STRIP est un protocole conçu spécialement pour un certain type de modems radio Metricom dans le cadre d'un projet de recherche conduit par l'Université de Stanford appelé *MosquitoNet Project* (`http://mosquitonet.Stanford.El`
Il y a un tas de choses intéressantes à lire, même si vous n'êtes pas directement concerné par le projet.

Les radios Metricom se connectent sur un port série et emploient la technologie à large bande spectrale et peuvent aller jusqu'à 100kbps. Des informations sur ceux-ci sont disponibles sur: *Le serveur web de Metricom* (`http://www.metricom.com/`).

A l'heure actuelle, les outils réseau habituels ne supportent pas le pilote STRIP, vous devez donc télécharger des outils personnalisés à partir du serveur web MosquitoNet. Pour avoir des détails sur les logiciels à utiliser allez voir: *MosquitoNet STRIP Page* (`http://mosquitonet.Stanford.EDU/strip.html`).

En résumé la configuration consiste à utiliser un programme *slattach* modifié pour régler la discipline de ligne d'un périphérique série pour SLIP, puis à configurer le périphérique '`st[0-9]`' résultant comme vous le feriez pour Ethernet avec une exception importante: pour des raisons techniques STRIP ne supporte pas le protocole ARP , vous devez alors configurer manuellement les entrées ARP pour chacun des hôtes de votre sous-réseau. Cela ne devrait pas être trop contraignant.

6.30 Token Ring

Le noms de périphériques Token ring sont '`tr0`', '`tr1`' etc. Token Ring est un protocole LAN standard IBM en vue d'éviter les collisions en fournissant un mécanisme qui n'autorise qu'une seule station du LAN à transmettre à un moment donné. Un 'jeton' est détenu par une station à un moment donné, et celle-ci est la seule autorisée à émettre. Lorque c'est fait elle passe le jeton à la station suivante. Le jeton fait le tour de toutes les stations actives, d'où le nom de 'Token Ring' (anneau à jeton).

Options de compilation du noyau:

```
Network device support  --->
        [*] Network device support
        ....
        [*] Token Ring driver support
        < > IBM Tropic chipset based adaptor support
```

La configuration de token ring est identique à celle de l'Ethernet à l'exception du nom de périphérique réseau devant être configuré.

6.31 X.25

X.25 est un protocole de circuit basé sur la commutation de paquets défini par le C.C.I.T.T. (un groupe de normalisation reconnu par les compagnies de télécommunications dans la plupart du monde). Une implémentation de X.25 et LAPB est en cours dans les noyaux récents 2.1.*.

Jonathon Naylor jsn@cs.nott.ac.uk dirige le développement et une liste de diffusion a été créée pour discuter des affaires relatives à X.25 pour Linux. Pour y souscrire, envoyez un message à: majordomo@vger.rutgers.edu avec le texte "subscribe linux-x25" dans le corps du message.

Les dernières versions des outils de configuration peuvent être obtenues sur le site ftp de Jonathon à *ftp.cs.nott.ac.uk* (ftp://ftp.cs.nott.ac.uk/jsn/).

6.32 Carte WaveLan

Les noms de périphériques Wavelan sont 'eth0', 'eth1', etc.

Options de compilation du noyau:

```
Network device support  --->
        [*] Network device support
        ....
        [*] Radio network interfaces
        ....
        <*> WaveLAN support
```

La carte WaveLAN est une carte LAN sans-fil à large bande. Elle ressemble beaucoup en pratique à une carte Ethernet et se configure presque de la même manière.

Vous pouvez avoir des informations sur la carte Wavelan sur *Wavelan.com* (http://www.wavelan.com/).

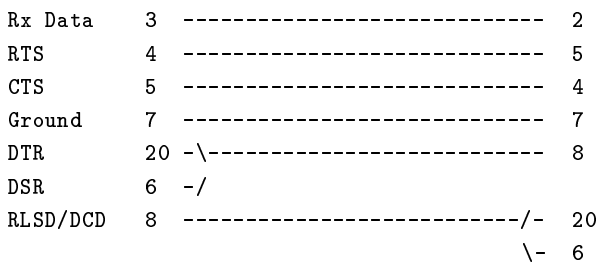
7 Câbles et câblages

Ceux qui sont habiles du fer à souder peuvent vouloir fabriquer leurs propres câbles pour relier deux machines Linux. Les schémas de câblage suivants pourront les y aider.

7.1 Câble série NULL Modem

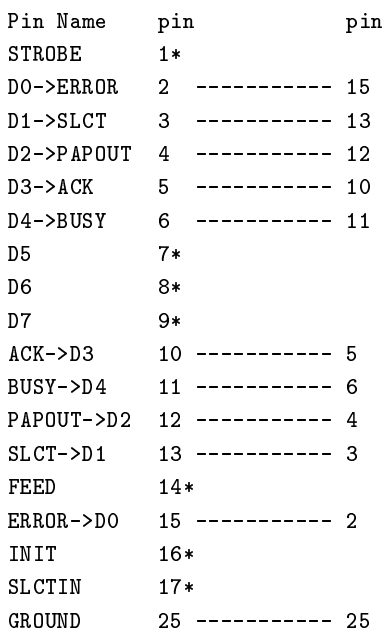
Tous les câbles NULL modem ne se ressemblent pas. Beaucoup ne font que faire croire à votre ordinateur que tous les signaux appropriés sont présents et échangent les données de transmission et de réception. C'est bien, mais cela signifie que vous devez utiliser le contrôle de flux logiciel (XON/XOFF) qui est moins efficace que le contrôle de flux matériel. Le câble suivant donne la meilleure transmission de signal entre les deux machines et vous permet d'utiliser le contrôle de flux matériel (RTS/CTS).

```
Pin Name  Pin                              Pin
Tx Data    2  ------------------------------  3
```



7.2 Câble port parallèle (câble PLIP)

Si vous avez l'intention d'utiliser le protocole PLIP entre deux machines alors ce câble vous conviendra indépendamment du type de port parallèle installé.



Notes:

- Ne pas connecter les broches marquées avec un astérisque '*'.
- Les masses supplémentaires sont 18,19,20,21,22,23 et 24.
- Si le câble que vous utilisez possède un blindage, il doit être connecté à une seule des prises DB-25 et **une seule**.

Attention: un câble PLIP mal branché peut détruire votre carte contrôleur.. Soyez attentifs et vérifiez chaque connexion deux fois pour être sûrs de ne pas vous créer de travail inutile ou de gros ennuis. Bien que l'on puisse utiliser des câbles PLIP sur des longues distances, évitez-le si possible. Les spécifications du câble permettent d'avoir une longueur d'environ 1 mètre. Faites attention si vous utilisez de grandes

longueurs, car les sources de champs magnétiques élevés comme la foudre, les lignes de puissance et les émetteurs radio peuvent interférer et parfois endommager votre carte contrôleur. Si vous voulez vraiment connecter deux de vos ordinateurs sur une grande distance, utilisez plutôt des cartes Ethernet et un câble coaxial.

7.3 Câblage Ethernet 10base2 (coaxial fin)

10base2 est un standard de câblage Ethernet spécifiant l'utilisation d'un câble coaxial 52 ohms avec un diamètre d'environ 5 mm. Il faut se souvenir d'un nombre important de régles quand on relie deux machines avec un câblage 10base2. La première est que vous devez utiliser des terminaisons **à chaque extrémité** du câble. Un terminateur est une résistance de 52 ohms qui sert à s'assurer que le signal est absorbé et non réfléchi à l'extrémité du câble. Sans terminaison à chaque extrémité vous pourriez trouver que l'Ethernet n'est pas fiable ou ne marche pas du tout. Normalement vous utilisez des 'T' pour interconnecter les machines, en sorte que vous finirez par avoir quelque chose qui ressemble à ceci:

```
|==========T=============T=============T==========T==========|
           |             |             |          |
           |             |             |          |
         -----         -----         -----      -----
         |   |         |   |         |   |      |   |
         -----         -----         -----      -----
```

Les '|' à chaque extrémité représentent une terminaison, les '======' représentent une longueur de câble coaxial avec des prises BNC en bout et les 'T' représentent un connecteur en 'T'. Gardez la longueur de câble entre les connecteurs en 'T' et les cartes Ethernet aussi courte que possible, l'idéal étant que ces connecteurs soient branchés directement sur la carte Ethernet.

7.4 Câblage Ethernet à paires torsadées

Si vous n'avez que deux cartes Ethernet avec paires torsadées et que vous voulez les relier, vous n'avez pas besoin de répartiteur. Vous pouvez câbler les deux cartes directement ensemble. Un schéma montrant comment faire est inclus dans le document *Ethernet-HOWTO* (`Ethernet-HOWTO.html`)

8 Glossaire des termes utilisés dans ce document.

Ci-dessous une liste des termes les plus importants utilisés dans ce document.

ARP

C'est l'acronyme de *Address Resolution Protocol* (protocole de résolution d'adresses), permettant à une machine du réseau d'associer une adresse IP à une adresse matérielle.

ATM

C'est l'acronyme de *Asynchronous Transfer Mode* (mode de transfert asynchrone). Un réseau ATM enveloppe les données en blocs de taille standard pour pouvoir les convoyer efficacement d'un point à un autre. ATM est une technologie réseau à commutation de paquets.

client

C'est habituellement le morceau de logiciel à l'extrémité d'un système où se trouve l'utilisateur. Il y a des exceptions, par exemple, dans le système de fenêtres X11 c'est en fait le serveur qui est avec l'utilisateur et le client qui est sur la machine distante. Le client est le programme ou l'extrémité d'un système qui utilise le service fourni par un serveur. Dans le cas de systèmes *d'égal à égal* tels que *slip* ou *ppp* le client se trouve à l'extrémité qui a initialisé la connexion, l'autre extrémité, étant considérée comme le serveur.

datagramme

Un datagramme est un paquet discret de données qui contient les adresses, et qui est l'unité de base de transmission sur un réseau IP. On peut aussi l'appeler 'paquet'.

DLCI

DLCI veut dire 'Data Link Connection Identifier'(identifieur de connexion de liaison données), et est utilisé pour identifier une liaison virtuelle unique point à point via un réseau à relais de trames (Frame Relay). Les DLCI sont normalement assignés par le fournisseur de réseau à relais de trames.

Relais de trames

Frame Relay (Relais de trames) est une technologie réseau idéale lorsque l'on a un trafic de nature cahotique ou sporadique. Les coûts peuvent être réduits quand on a de nombreux clients Frame Relay partageant la même capacité réseau et on compte sur le fait que les clients utilisent le réseau à des instants diifférents.

Adresse matérielle

C'est un nombre qui identifie de manière unique un hôte sur un réseau physique au niveau de la couche accès. Par exemple: *Adresses Ethernet* et *Adresses AX.25*.

ISDN

C'est l'acronyme de *Integrated Services Digital Network*(Réseau Numérique à Intégration de Services=RNIS). ISDN fournit des moyens standardisés avec lesquels les compagnies de télécommunications peuvent délivrer soit de la voix soit des informations vers des clients. Techniquement ISDN est un réseau de données à commutation de paquets.

ISP

C'est l'acronyme de 'Internet Service Provider' (fournisseur d'accès à l'Internet). Ce sont des organisations ou des sociétés qui fournissent aux gens une connexion réseau à l'Internet.

Adresse IP

C'est un nombre qui identifie de manière unique un hôte TCP/IP sur le réseau. Cette adresse est codée sur 4 octets et se présente habituellement sous la forme appelée "notation décimale pointée", où chaque octet est sous forme décimale, avec des points '.' entre chaque.

MSS

Le Maximum Segment Size (*MSS*) (Taille Maximum de Segment) est la plus grande quantité de données qui peut être transmise en une seule fois. Si vous voulez éviter des fragmentations MSS doit être égal à l'en-tête MTU-IP.

MTU

Le Maximum Transmission Unit (*MTU*) (taille maximum de l'unité de transfert) est un paramètre qui détermine le plus long datagramme pouvant être transmis par une interface IP sans avoir besoin d'être fragmenté en unités plus petites. Le MTU doit être plus grand que le datagramme le plus grand que vous voulez transmettre sans être fragmenté. Note: ceci protège de la fragmentation uniquement de manière locale, d'autres liens sur le chemin peuvent avoir un MTU plus petit et les datagrammes seront fragmentés à cet endroit. Les valeurs typiques sont de 1500 octets pour une interface Ethernet, ou de 576 octets pour une interface SLIP.

route

La *route* est le chemin que les datagrammes suivent à travers le réseau pour atteindre leur destination.

serveur

C'est habituellement le morceau de logiciel ou l'extrémité d'un système éloigné de l'utilisateur. Le serveur fournit un service vers un ou plusieurs clients. Des exemples de serveurs sont *ftp*, *Networked File System* (NFS), ou *Domain Name Server* (DNS). Dans le cas de systèmes *égal à égal* comme *SLIP* ou *PPP* le serveur est considéré comme étant l'extrémité de la liaison qui est appelée et l'extrémité appeleante est le client.

fenêtre

La *fenêtre* (window) est la plus grande quantité de données que l'extrémité réceptrice peut accepter à un certain moment.

9 Linux pour un fournisseur d'accès à l'Internet?

Si vous êtes intéressés par l'utilisation de Linux à des fins de fourniture d'accès Internet, je vous recommande de consulter sur *Linux ISP homepage* (`http://www.anime.net/linuxisp/`) pour une bonne liste de pointeurs vers les informations dont vous pourriez avoir besoin.

10 Remerciements

Je voudrais remercier les personnes suivantes pour leur contribution à ce document (sans ordre particulier): Terry Dawson, Axel Boldt, Arnt Gulbrandsen, Gary Allpike, Cees de Groot, Alan Cox, Jonathon Naylor, Claes Ensson, Ron Nessim, John Minack, Jean-Pierre Cocatrix, Erez Strauss.

11 Copyright.

Le document NET-3-HOWTO donne des informations concernant l'installation et la configuration du support réseau pour Linux. Copyright (c) 1997 Terry Dawson.

Celui-ci est libre; vous pouvez le redistribuer et/ou le modifier selon les termes de la GNU General Public License telle que publiée par la Free Software Foundation; soit avec la version 2 de la license, soit (à votre guise) avec une version ultérieure.

Ce document est distribué avec l'espoir qu'il sera utile, mais SANS AUCUNE GARANTIE; ni même la garantie implicite de COMMERCIALISATION ou D'ADAPTATION DANS UN BUT PARTICULIER. Voir la GNU General Public License pour plus de détails.

Vous devriez recevoir une copie de la GNU General Public License avec ce document; si ce n'est pas le cas, écrivez à:

Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.