Linux Term HOWTO v1.1, version française.

par Patrick Reijnen, patrickr@cs.kun.nl
12 Mai 1995

Ce document expose les bases de l'installation et du fonctionnement de `term`, programme grâce auquel il est possible de multiplexer une liaison série entre deux machines dont l'une est connectée à un réseau. On peut ainsi mettre en place une connexion au réseau sur la seconde machine.

Table des matières

1 Informations légales

1.1 Version française

1.1.1 Copyright

Ce document peut être librement distribué, sous quelque forme que ce soit, s'il l'est gratuitement et dans son intégralité. Des extraits peuvent en être diffusés, sous réserve que les présentes informations de *copyright* soient incluses et que le lecteur soit informé du fait qu'il ne s'agit pas de la version intégrale, dont on indiquera comment se la procurer. Ce document pourra être inclus dans des distributions commerciales sans consentement explicite préalable. L'auteur souhaite cependant être averti de telles utilisations.

Ce `HOWTO` pourra être traduit dans toute langue, à condition que les informations de *copyright* et de limitation de garantie soient fournies sous leur forme originelle et que l'identité du traducteur soit indiquée dans la traduction[1]. Le traducteur souhaite de plus être informé de toute utilisation de ce document dans une distribution commerciale.

1.1.2 Limitation de garantie

Bien que l'auteur ait essayé de fournir les informations les plus exactes et récentes possibles, il ne peut garantir que l'utilisation de ce document ne provoquera pas la perte de données. Il ne fournit donc **AUCUNE**

1. Le traducteur est en l'occurence Arnaud Ruch (ruch@labri.u-bordeaux.fr).

GARANTIE sur les informations contenues dans ce HOWTO et ne pourra en aucun cas être tenu pour responsable de quelconques dommages résultant de son utilisation [2].

1.2 Version originale

1.2.1 Copyright statement

1.2.2 DISCLAIMER

While I have tried to include the most correct and up-to-date information available, I cannot guarantee that usage of the information in this document does not result in loss of data. I provide **NO WARRANTY** about the information in this HOWTO and I cannot be made liable for any consequences for any damage resulting from using information in this HOWTO.

2 Introduction

2.1 À propos de ce document

Ce HOWTO tente de dissiper la confusion que peut engendrer l'utilisation de `term`, remarquable programme de Michael O'Reilly grâce auquel il est possible de multiplexer une liaison série, afin de mettre en place une connexion à un réseau. La documentation fournie avec `term` étant de relativement bonne qualité, ce document ne vise en aucun cas à la remplacer. Son intention est d'expliquer sur quelles bases fonctionne `term` et de détailler les étapes nécessaires à la mise en place de quelques-uns des services réseau les plus courants. L'auteur insiste sur le fait que ce document ne couvre pas l'ensemble de ce qu'il y a à connaître sur `term`. Après l'avoir parcouru, reportez-vous aux pages du manuel qui contiennent de nombreuses informations complémentaires.

2.2 Qu'est-ce que `term` ?

`term` est un programme écrit par Michael O'Reilly (`michael@iinet.com.au`) et maintenu par Bill Riemers (`bcr@physics.purdue.edu`), qui permet à plusieurs connexions d'opérer simultanément sur une même liaison série. Il est par exemple possible de télécharger un fichier tout en travaillant sur un site distant différent, et ce

2. Il en va bien entendu de même du traducteur.

grâce à la même liaison modem. `term` permet également de lancer des clients `X Windows` par l'intermédiaire de cette même liaison. Grâce aux utilitaires `tredir` et `tudpredir`, `term` est en mesure de fournir la plupart des services réseau `TCP/IP` et `UDP` traditionnels : *mail*, *news*, `ftp`, `telnet`, `xarchie`, etc. D'un certain point de vue, `term` est très semblable aux autres protocoles série tel que `SLIP` et `PPP`, mais son avantage est d'être exécuté intégralement depuis l'espace utilisateur. Il ne requiert donc aucune intervention d'administrateurs système ou réseau.

Contrairement à `SLIP` ou `PPP`, votre machine ne possède pas sa propre adresse `IP`. Tout le trafic à destination de votre machine doit donc être adressé à une machine distante, qui se charge de le rediriger vers elle par l'intermédiaire de `term`.

3 Comment `term` fonctionne-t-il ?

Avant de vous lancer dans l'expérimentation de `term`, il est vivement recommandé de lire ce chapître dans son intégralité et de parcourir le fichier '`INSTALLATION`' fourni avec `term`. Examinez également les pages de manuel de `linecheck`, `(term)test` et `term`. Vous pourrez alors travailler plus facilement et plus rapidement.

3.1 Nomenclature

Supposons que vous appelez une machine par l'intermédiaire d'un quelconque serveur de terminal. Les termes *local* et *distant* se réfèrent respectivement au système local et à la machine distante, qui est connectée au réseau.

La machine locale n'est pas directement reliée au réseau. `term` est cependant capable, en utilisant la connexion au réseau dont dispose la machine distante, de lui fournir des services réseau, par l'intermédiaire de la liasion série. Examinons comment une machine dotée d'une connexion réseau traditionnelle fournit ces services. Dans un premier temps, l'utilisateur lance un programme tel que `ftp` ou `telnet`. Ce programme demande un service réseau par un appel système. Le système d'exploitation obtient alors ce service de son interface réseau (par exemple, il envoie et reçoit des paquets *éthernet*). `SLIP` et `PPP` procèdent exactement de la même façon, en transformant en interface réseau votre liaison modem. En pratique en effet, une liaison série n'est guère différente d'une liaison *éthernet*. L'inconvénient majeur de ces protocoles est qu'ils font de la machine connectée par modem une part intégrante du réseau, avec les nombreuses charges d'administration qui en découlent (plus, en fait, puisque la liaison modem doit également être administrée).

En l'absence d'une connexion réseau telle que `SLIP` ou `PPP`, que fait-on habituellement ? Et bien, on appelle une machine connectée au réseau, on lit son courrier, ses *news*, etc. Lorsqu'on a besoin d'un fichier, on commence par le transférer sur la machine distante, puis on le rapatrie sur la machine locale grâce à `kermit` ou un autre programme de communication. Tout ceci est quelque peu astreignant, d'autant plus que vous ne pouvez utiliser votre liaison modem que pour une opération à la fois. L'idée à la base de `term` est d'automatiser et de multiplexer ces opérations. `term` est lancé à la fois sur la machine locale et sur la machine distante, les deux processus communiquant par la liaison modem. Lorsque vous avez besoin d'un service réseau, vous adressez une requête au démon `term` local, qui la retransmet au démon `term` qui s'exécute sur la machine connectée au réseau. Le résultat est alors renvoyé par l'intermédiaire de la liaison modem.

Pour être plus concrêt, disons que vous souhaitez récupérer un fichier grâce à `ftp`. Tout d'abord, il vous faut une version de `ftp` capable de dialoguer avec `term`. Vous lancez ce `termftp` comme un `ftp` classique, par la commande `termftp nethost.gov`, par exemple. La différence est que cette version particulière de `ftp` adresse sa requête au démon `term` local plutôt qu'au noyau. Le démon `term` local retransmet cette requête au `term` distant par l'intermédiaire de la liaison modem, celui-ci se chargeant alors d'ouvrir une connexion `ftp` avec `nethost.gov` et de renvoyer les données. `term` est suffisament astucieux pour vous permettre de gérer de nombreuses actions à, la fois de telle sorte qu'il vous est possible d'avoir plusieurs sessions ouvertes simultanément sur la même liaison modem (par exemple, vous pouvez être connecté à une machine distante par `termtelnet` pendant que le transfert `termftp` se poursuit).

Ne vous inquiétez pas si tout ceci vous semble trop abstrait (ou bien confus) : ce qu'il faut ici retenir est qu'il y a *deux* occurences de `term` qui s'exécutent, chacune à une extrêmité de la liaison modem.

4 Mise en place

4.1 Ce dont vous devez disposer

Avant de commencer à construire `term`, il vous faut être sûr que votre noyau a été compilé avec l'option `TCP/IP`. L'interface `TCP/IP loopback` doit également avoir été activée. Une fois que ceci est fait, vous pouvez parcourir le reste de ce paragraphe.

4.2 Explication des concepts

Dans les dernières versions de `term`, deux nouveaux concepts ont fait leur apparition. Les deux paragraphes suivants y sont consacrés.

4.2.1 Le partage

Depuis la version 1.16 de `term`, le concept de *partage* de la connexion `term` avec d'autres utilisateurs est apparu. Si, par exemple, vous travaillez sur la machine distante via votre connexion `term` (vous avez utilisé `trsh` pour vous connecter depuis votre machine locale), un autre utilisateur peut utiliser la même connexion pour transférer par `ftp` un fichier sur son compte sur la machine locale, depuis un site `ftp` distant.

Lorsque vous désactivez les possibilités de partage (c'est-à-dire que vous utilisez `term` en mode *privé*), vous seul (en plus de *root* :-) pouvez utiliser la connexion `term`.

Bien sûr, vous n'avez besoin d'installer le `term` partagé qu'à l'extrêmité à laquelle vous voulez permettre à d'autres utilisateurs de profiter de la même connexion `term` que vous. Ainsi, si d'autres utilisateurs ont un compte sur votre machine locale et souhaitent l'utiliser depuis une autre machine du réseau distant, il vous faut activer le partage à l'extrêmité distante de la connexion `term`. De cette façon, ces personnes peuvent se connecter à votre machine locale en utilisant la même connexion `term` que vous. (NOTE : le premier exemple supposait que le partage soit activé à l'extrêmité locale de la connexion).

NOTE sur l'installation en tant que root : Lorsque vous installez `term` en tant que *root*, il vous faut au préalable créer un groupe *term* (avant la compilation) ne contenant aucun utilisateur. Ceci se fait par l'ajout de la ligne suivante au fichier '/etc/group' :

```
term::16:root
```

ou tout *GID* inutilisé autre que 16, si 16 est déjà attribué.

Après la compilation et l'installation, rendez `term` et ses clients *SUID*, grâce aux commandes :

```
chgrp term <client_term>
chmod g+s <client_term>
```

Procédez de même avec tout autre programme utilisant `term`.

4.2.2 Réseau `term` intégral

On utilise l'expression de *réseau term intégral* depuis la version 2.0.0 de `term`. Lorsque votre seule connexion au monde extérieur est une connexion `term`, vous disposez d'un *réseau term intégral*. Il vous faut alors compiler `term` avec cette option. Dans ce cas, un fichier '`termnet`' est placé dans votre répertoire partagé. Ceci indique à `term` qu'il assure votre unique connexion avec le monde extérieur.

Si vous avez également un autre type de connexion réseau vers l'extérieur, les programmes qui savent utiliser `term` tenteront dans un premier temps d'utiliser cette connexion. Si cela échoue, `term` sera lancé et une nouvelle tentative sera effectuée, avec la connexion `term`. Pour être plus clair, voici un exemple qui utilise la version `term` de `telnet`, laquelle peut fonctionner avec ou sans `term`.

```
telnet localhost
```

n'utilise pas `term` pour réaliser la connexion alors que :

```
telnet zeus.cs.kun.nl
```

utlisera `term` si vous n'avez pas d'autre type de connexion réseau.

Un *réseau term intégral* suppose que l'on mente à propos du nom de la machine locale, en prétendant qu'il s'agit de la machine distante. De plus, cela implique que `bind(0)` agisse toujours sur la machine distante. Sommairement, cela signifie que de nombreux programmes n'utilisant pas `term` deviennent inutilisables lorsque `term` est actif. Malheureusement, c'est le cas de la plupart des programmes et démons `UDP`.

4.3 Compilation de `term`

Si vous avez de la chance, un simple `make` devrait suffire. Il est cependant probable que vous aurez un peu plus de travail. Du fait des nouvelles possibilités des version récentes de `term`, la construction de l'exécutable est désormais un peu compliquée. On peut procéder de plusieurs façons.

Afin de détailler toutes ces manières de construire `term`, cette section est divisée en trois parties :

1. Compilation de `term`, versions 2.0.0 et ultérieures.

2. Compilation de `term`, versions 1.16 à 1.19.
3. Compilation de `term`, versions 1.15 et antérieures.

4.3.1 Compilation de term, versions 2.0.0 et ultérieures

Tout d'abord, assurez-vous d'avoir bien lu la partie '*réseau term intégral*' ci-dessus.

Il existe plusieurs façons de compiler l'exécutable et les clients `term` des versions 2.0.0 et ultérieures. Toutes peuvent être utilisées par *root* ou par un utilisateur normal :

1. Compilation de `term` en mode privé sans *réseau term intégral*.
2. Compilation de `term` en mode privé avec *réseau term intégral*.
3. Compilation de `term` en mode partagé sans *réseau term intégral*.
4. Compilation de `term` en mode partagé avec *réseau term intégral*.

Dans ces versions de `term`, une nouvelle procédure de compilation est apparue, grâce au *script* `configure`. Lorsque `configure` est lancé, il détermine le système d'exploitation, si le répertoire source est accessible ou non et si des options d'exécution ont été choisies. En fonction de ce qu'il a trouvé, il crée un fichier '`Makefile`' à partir du '`Makefile.in`' fourni avec `term`.

Deux des plus importantes options à configurer sont `--root` et `--user` qui déterminent si `term` doit être installé par *root* ou par un utilisateur classique. D'autres options peuvent être utilisées pour installer `term` à votre goût (dans une arborescence non standard, par exemple).

1. Compilation de `term` en mode privé sans *réseau term intégral*.

   Pour installer `term` de cette façon, vous devez lancer les commandes (que ce soit en tant que *root* ou en tant qu'utilisateur ordinaire) :

   ```
   ./configure --root OU --user
   make install installman
   ```

   Ceci construit les exécutables, les installe et installe les pages de manuel.

2. Compilation de `term` en mode privé avec *réseau term intégral*.

   Pour installer `term` de cette façon, vous devez lancer les commandes (que ce soit en tant que *root* ou en tant qu'utilisateur ordinaire) :

   ```
   ./configure --root OU --user
   make installnet installman
   ```

   Ceci construit les exécutables, les installe et installe les pages de manuel.

3. Compilation de `term` en mode partagé sans *réseau term intégral*.

   Pour installer `term` de cette façon, vous devez lancer les commandes (que ce soit en tant que *root* ou en tant qu'utilisateur ordinaire) :

   ```
   ./configure --root OU --user
   make share installman
   ```

   Ceci construit les exécutables, les installe et installe les pages de manuel.

4. Compilation de `term` en mode partagé avec *réseau term intégral*.

   Pour installer `term` de cette façon, vous devez lancer les commandes (que ce soit en tant que *root* ou en tant qu'utilisateur ordinaire) :

   ```
   ./configure --root OU --user
   make share installnet installman
   ```

   Ceci construit les exécutables, les installe et installe les pages de manuel.

4.3.2 Compilation de `term`, versions 1.16 à 1.19

Pour installer ces versions de `term` vous pouvez choisir l'une des procédures suivantes :

1. Compilation de `term` en mode privé, en tant qu'utilisateur ordinaire.
2. Compilation de `term` en mode partagé, en tant qu'utilisateur ordinaire.
3. Compilation de `term` en mode privé, en tant que *root*.
4. Compilation de `term` en mode partagé, en tant que *root*.

Comment activer/désactiver les fonctionnalités de partage de `term` durant la compilation est expliqué ci-après.

1. Vous êtes un utilisateur ordinaire (pas de compte *root*) et vous ne souhaitez PAS partager votre connexion `term` avec d'autres utilisateurs.

   Installez `term` de la façon suivante :

   ```
   make DO=install OS-type
   make installman
   ```

   Ceci compile puis installe `term`, ses clients et les pages de manuel.

   Il vous faut de plus créer un répertoire '`$HOME/.term`'. C'est le répertoire dans lequel `term` recherchera son fichier '`termrc`'.

2. Vous êtes un utilisateur ordinaire (pas de compte *root*) et vous souhaitez partager votre connexion `term` avec d'autres utilisateurs.

   Installez `term` de la façon suivante :

   ```
   make DO=installshare USERSHARE=$HOME/term OS-type
   make installman
   ```

   Ceci compile puis installe `term`, ses clients et les pages de manuel.

   Il vous faut de plus créer un répertoire '`$HOME/.term`' (nom par défaut) avec les droits `drwxrwxr-x`. Dans ce répertoire, vous trouverez au moins la *socket* utilisée par `term` pour sa connexion ('`tmp/private/socket=`').

3. Vous êtes *root* et vous ne souhaitez PAS partager votre connexion `term` avec d'autres utilisateurs.

   Installez `term` de la façon suivante :

   ```
   make DO=install OS-type
   make installman
   ```

   Ceci compile puis installe `term`, ses clients et les pages de manuel.

   Il vous faut de plus créer un répertoire '`/usr/local/lib/term`' (nom par défaut) avec les droits `drwxrwxr-x`. Dans ce répertoire, vous trouverez au moins la *socket* utilisée par `term` pour sa connexion ('`tmp/private/socket=`').

4. Vous êtes *root* et vous souhaitez partager votre connexion `term` avec d'autres utilisateurs.

   Tout d'abord, assurez-vous d'avoir lu la partie concernant le partage.

   Installez `term` de la façon suivante :

   ```
   make DO=installshare OS-type
   make installman
   ```

   Ceci compile puis installe `term`, ses clients et les pages de manuel.

   Il vous faut de plus créer un répertoire '`/usr/local/lib/term`' (nom par défaut) dont le propriétaire est le groupe *term*, avec les droits `drwxrwxr-x`. Dans ce répertoire, vous trouverez au moins la *socket* utilisée par `term` pour sa connexion ('`tmp/private/socket=`').

4.3.3 Compilation de `term`, versions 1.15 et antérieures

La compilation de ces versions de `term` ne devrait pas requérir d'autres commandes que :

```
make DO=install OS-type
make installman
```

Ceci compile puis installe `term`, ses clients et les pages de manuel. Tout est alors prêt à l'emploi.

Il vous faut de plus créer un répertoire '`$HOME/.term`'. C'est le répertoire dans lequel `term` recherchera son fichier '`termrc`'.

Les seules choses que vous pourrez souhaiter modifier sont certains des chemins du '`Makefile`' et certaines options du compilateur.

4.4 `client.a, libtermnet.a, libtermnet.sa, libtermnet.so`

Une bibliothèque est fournie avec `term`. Elle propose des fonctions pour les clients `term`.

Jusqu'à la version 1.16, cette bibliothèque était nommée `client.a`. Elles était bâtie lors de la compilation de `term`, puis utilisée lors de celle des clients. Elle n'était installée dans aucun répertoire.

Depuis la version 1.16, cette bibliothèque s'appelle `libtermnet.a`. Jusqu'à la version 1.19, elle est créée dans le répertoire `term`, puis utilisée pour la compilation des clients `term`. Elle n'est installée nulle part ailleurs.

A partir de la version 2.0.0, `libtermnet.so` et `libtermnet.sa` (bibliothèque partagée et données exportées et initialisées pour la bibliothèque) sont également créées lors de la compilation de `term`. Lors de l'installation de l'ensemble des composantes du paquetage, ces trois bibliothèques (`libtermnet.a, libtermnet.so` et `libtermnet.sa`) sont installées dans le répertoire '`/usr/local/lib`' (par défaut). Un lien est ensuite créé de `libtermnet.so.2.x.x` vers `libtermnet.so.2`. Finalement, `ldconfig` est lancé pour créer les liens et le cache nécessaires (pour le *linkage* à l'exécution, par `ld.so`). `ld.so` tient compte des versions les plus récentes des bibliothèques trouvées dans les répertoires précisés dans la ligne de commande, dans le fichier '`/etc/ld.so.conf`' et dans les répertoires '`/usr/lib`' et '`/lib`'. Si l'installation est effectuée correctement, ces trois bibliothèques peuvent alors être utilisées par tous les clients `term` qui sont construits avec des bibliothèques dynamiques plutôt que statiques. Ces bibliothèques peuvent également être utilisées pour le portage de vos propres programmes afin de leur permettre d'utiliser les fonctionnalités de `term` (voir plus loin).

4.5 Variables d'environnement

`term` reconnaît un certain nombre de variables d'environnement qui peuvent être positionnées par les utilisateurs. Les trois premières à être détaillées sont :

- `termDIR`
- `termSHARE`
- `termMODE`

Grâce à ces variables, il vous est possible de contrôler la façon dont `term` s'exécute.

Pour les versions de `term` 1.15 et antérieures, seule la variable `termDIR` est importante (ces versions ne proposent pas le mode partagé). Pour ces versions, positionnez `termDIR` de la façon suivante :

```
setenv termDIR $HOME     (csh ou tcsh)
export termDIR=$HOME     (ksh ou bash)
```

A partir de la version 1.16, `term` reconnait également les variables `termSHARE` et `termMODE`. Grâce à ces variables, on peut indiquer à `term` s'il doit fonctionner en mode privé ou partagé. Comment positionner ces variales pour les modes privé et partagé est expliqué ci-après.

`termMODE` peut avoir l'une des trois valeurs suivantes :

- 0 = privé

- 1 = partagé (système)
- 2 = partagé (utilisateur)

1. On peut utiliser `term` en mode privé en positionnant les variables `termDIR` et `termMODE` de la façon suivante :

   En csh ou tcsh :

   ```
   setenv termDIR $HOME
   setenv termMODE 0
   ```

   En ksh ou bash :

   ```
   export termDIR=$HOME
   export termMODE=0
   ```

2. Il y a deux manières de positionner les variables pour utiliser `term` en mode partagé :

   (a) Lorsque `term` est installé comme un programme *SUID*, seule `termMODE` doit être positionnée.

   ```
   setenv termMODE 2    (csh ou tcsh)
   export termMODE=2    (ksh ou bash)
   ```

   (b) Lorsque `term` est installé comme un programme *SGID*, il faut positionner les variables comme suit :

   En csh ou tcsh :

   ```
   setenv termMODE 1
   setenv termDIR /usr/local/lib/term
   setenv termSHARE $termDIR
   ```

   En ksh ou bash :

   ```
   export termMODE=1
   export termDIR=/usr/local/lib/term
   export termSHARE=$termDIR
   ```

   Positionner les variables de cette manière permet d'utiliser les anciens clients (*linkés* avec une ancienne version de `client.a`) en mode partagé.

A partir de la version 2.0.0, `term` reconnaît également la variable `termSERVER`. Celle-ci doit être positionnée si vous possédez plusieurs modems et utilisez plus d'une connexion à la fois. Pour préciser quelle connexion utiliser, il faut lancer `term` avec le nom d'un serveur.

```
nohup term -v /dev/modem1 Connection1 &
nohup term -v /dev/modem2 Connection2 &
```

Les utilisateurs doivent alors positionner la variable `termSERVER` sur le nom de la connexion qu'ils souhaitent utiliser :

```
setenv termSERVER Connection1   (csh ou tcsh)
export termSERVER=Connection2   (ksh ou bash)
```

4.6 Test de term

Tapez `make test` (ou `make termtest` pour les versions récentes de `term`) afin de compiler le démon de test de `term`. `(term)test` fonctionne en exécutant deux occurences de `term` sur votre système, une copie 'locale' et une copie 'distante'. Chacune d'elles lira votre '`termrc`' de façon à ce que vous puissiez en ajuster le comportement. Lancez `(term)test` maintenant. Vous devriez être en mesure de lancer un `trsh` et un `tupload`. Essayez :

```
tupload ./term /usr/tmp
```

ceci devrait placer une copie de l'exécutable `term` dans le repértoire '`/usr/tmp`'. La sortie du `term` local apparaît dans le fichier '`local.log`', celle du `term` distant dans '`remote.log`'. Vous pouvez lancer `term` avec l'option `-d255` de façon à ce que des informations de déboguage soient incluses dans ces fichiers ou pour vous permettre de déboguer votre '`termrc`'.

NOTE : lancez `test` par la commande `./test` de façon à ne pas lancer le `test` du système.

4.7 term et les programmes de communication

Avant de pouvoir utiliser `term`, il faut avoir établi une connexion par modem, par l'intermédiaire d'un programme de communication tel que `kermit` ou `seyon`. La documentation de votre programme de communication vous fournira les informations dont vous aurez besoin pour vous connecter à une machine distante.

Lorsque vous avez établi la connexion avec la machine distante, il vous faut interrompre ou quitter votre programme de communication pour pouvoir utiliser `term`, mais sans couper la connexion. Ceci est nécessaire pour éviter que le programme de communication subtilise des caractères à `linecheck` ou `term`.

Comment s'assurer que la connexion reste active et que le programme de communication ne s'approprie pas certains caractères est expliqué ci-après, pour quelques exemples.

4.7.1 kermit

Il est facile de lancer `term` lorsque vous utilisez `kermit` : à l'invite de `kermit`, tapez la commande `suspend`. Vous voila de retour à l'invite de `Linux`. A partir de là, vous pouvez établir votre connexion `term`.

4.7.2 seyon

Un moyen facile de lancer `linecheck` ou `term` à partir de `seyon` est de les placer dans le menu *Transfer* (contrôlé par le fichier '`$HOME/.seyon/protocols`').

Dans le fichier '`$HOME/.seyon/protocols`', ajoutez :

```
"linecheck" "$cd /tmp; linecheck"
"term" "$term -c off -w 10 -t 150 -s 38400 -l $HOME/tlog"
```

Ainsi, lorsque vous voulez lancer `linecheck` ou `term` sur la machine locale, vous pouvez sélectionner le menu *Transfer*, choisir l'option *linecheck* ou *term*, puis *Go*.

Bien entendu, il est toujours possible d'utiliser le bouton *shell command* et de taper 'linecheck' ou 'term' dans la boîte de dialogue *pop-up*. Ceci redirige automatiquement l'entrée et la sortie.

4.8 Rendons la liaison transparente

Supposons que vous savez établir une connexion modem entre vos machines locale et distante. De façon typique, vous numérotez grâce à un serveur de terminal quelconque pour vous connecter à la machine distante. Vous utilisez également quelque logiciel d'émulation de terminal, tel que `kermit` ou `seyon` pour dialoguer avec votre modem (les exemples fournis dans ce document utilisent `kermit`, puisque c'est ce dont l'auteur se sert). Si vous avez des problèmes avec votre modem ou avec votre logiciel de terminal, reportez-vous au `Serial-HOWTO`, il devrait vous permettre de les régler.

Une fois votre liaison établie, il faut la rendre aussi transparente que possible. Etudiez les commandes de votre serveur de terminal (*help* ou *?* sont en général un bon début). Choisissez les options 8 bits chaque fois que cela est possible. Il est possible que cela influe sur la façon de vous connecter à un système. Si le serveur utilise `rlogin`, par exemple, il faudra peut-être utiliser l'option `-8` afin de le rendre transparent. Faites particulièrement attention à ne pas utiliser le contrôle de flux `XON/XOFF`. Essayez plutôt d'utiliser le contrôle de flux `RTS/CTS` (*hardware*). L'étude de la documentation de votre modem vous apprendra certainement comment le configurer pour des communications `RTS/CTS` 8 bits.

4.9 Lançons `linecheck`

ATTENTION : dans certains documents, les options de `linecheck` sont données dans un ordre incorrect. L'auteur les a contrôlées et en donne ici l'ordre correct.

NOTE : à partir de la version 2.3.0, il n'est plus nécessaire de fournir à `linecheck` le nom d'un fichier de *log*. La sortie sera écrite dans le fichier '`linecheck.log`', dans le répertoire d'où `linecheck` a été lancé.

`linecheck` est un programme fourni avec `term` qui vérifie la transparence d'une liaison et fournit des informations utiles pour une configuration correcte de `term`. `linecheck` envoie chacun des 256 caractères 8 bits possibles sur la liaison et vérifie que tous ont été transmis correctement. Il faut indiquer à `term` quels caractères ne peuvent pas être transmis sur la liaison, ce que `linecheck` permet de repérer. `linecheck` doit être utilisé une fois que la liaison modem est aussi transparente que possible. Pour lancer `linecheck`, procédez comme suit :

1. Sur le système distant, lancez :

   ```
   linecheck linecheck.log
   ```

2. Revenez à votre système local et interrompez votre programme de communication (voir au-dessus).

3. Sur le système local, lancez :

   ```
   linecheck linecheck.log > /dev/modem < /dev/modem
   ```

Lorsque `linecheck` est terminé, les fichiers '`linecheck.log`' contiennent un ensemble de nombres, placés à la fin. Ce sont les caractères qui doivent être *évités* dans le '`termrc`' placé à l'autre extrêmité de la liaison. Sur le système de l'auteur, par exemple, le '`linecheck.log`' local ne disait rien et le '`linecheck.log`' distant indiquait d'*éviter* 29 et 157. C'est pourquoi son '`termrc`' local *évite* ces caractères, alors que son '`termrc`' distant n'en *évite* aucun. Si l'on *évite* des caractères d'un côté, il faut les *ignorer* de l'autre. C'est pourquoi, l'auteur doit *ignorer* 29 et 157 sur son système distant.

Si `linecheck` se plante, essayez :

```
linecheck linecheck.log 17 19
```

sur le système distant et :

```
linecheck linecheck.log 17 19 > /dev/modem < /dev/modem
```

sur le système local. Ceci a pour effet de ne pas envoyer les caractères de contrôle de flux `XON/XOFF` qui plantent votre ligne si vous utilisez un contrôle de flux logiciel. Si cela résoud le problème, *évitez/ignorez* 17 et 19 dans les deux '`termrc`'. Si votre serveur de terminal est sensible à d'autres caractères, procédez de même. Il est possible d'identifier de tels caractères lorsque `linecheck` se plante : tuez-le puis examinez les fichiers de *log*. Les derniers caractères transmsis sont de bons coupables potentiels. Essayez à nouveau en *évitant* ces caractères.

En résumé, le '`termrc`' local de l'auteur se présente ainsi :

```
escape 29
escape 157
```

et son '`termrc`' distant ainsi :

```
ignore 29
ignore 157
```

puisque le '`linecheck.log`' *distant* indiquait *d'éviter* 29 et 157.

4.10 Essayons de lancer `term`

Connectez-vous au système distant et rendez la liaison aussi transparente que possible (si ça n'est déjà fait). Lancez-y `term`, par exemple de la façon suivante :

```
exec term -r -l $HOME/tlog -s 38400 -c off -w 10 -t 150
```

Examinons une à une les options de cette ligne de commande (que nous aurions tout aussi bien pu inclure dans le '`termrc`', mais au prix d'une édition de fichier).

`exec` signifie que le programme (`term`, en l'occurence) remplace le *shell* à partir duquel il est lancé. On utilise `exec` lorsque l'on n'a plus l'intention de se servir du *shell* concerné; on économise ainsi la mémoire. Cependant, lors de la mise au point de la liaison, il peut être prudent de ne pas faire un `exec`, de façon à pouvoir tuer le `term` distant.

L'option `-r` doit être utilisée à une et une seule extrêmité. Celle-ci sera alors vue par `term` comme l'extrêmité distante de la liaison (qui peut donc être votre machine locale). Si cette option n'est pas utilisée d'un côté ou de l'autre, `term` se plantera spontanément.

`-l $HOME/tlog` permet de consigner toutes les erreurs dans le fichier '`tlog`', placé dans votre répertoire utilisateur. Celui-ci étant très utile pour les mises au point, il n'y a pas de raison de ne pas utiliser cette option.

`-s 38400` : l'auteur dispose d'un modem 14400 *bps*, avec compression. Afin d'obtenir des taux de compression optimaux, il faut envoyer les bits aussi vite que possible. Pour un modem plus lent, il faut utiliser une valeur plus faible. Notez que si vous possédez une machine « lente » dotée de ports séries disposant d'`UART 16450`, des valeurs de *bps* trop importantes risquent de provoquer des pertes de données en surchargeant le port série. `term` est capable de corriger ceci, mais si les fichiers de *log* comportent de nombreux messages d'erreur (ou si vous recevez de nombreux *warnings* des noyaux `linux` 0.99pl15 ou plus récents), il serait plus sage de diminuer cette valeur.

`-c off` : cette option désactive la compression. Lorsque l'on dispose d'un modem qui assure déjà la compression des données, il n'est nul besoin de les compresser une seconde fois.

`-w 10 -t 150` : ici encore, cette option permet d'optimiser une liaison modem rapide. La valeur de la fenêtre est 10 et celle du *timeout* 150, selon les recommendations de la page de manuel de `term_setup`.

Revenez sur votre machine locale et interrompez votre programme de communication (voir ci-dessus). Il est préférable qu'il ne tourne pas en même temps que `term`, car il risque d'entrer en conflit avec lui au niveau du port série. Si vous parvenez à convaincre votre programme de communication de ne pas raccrocher la ligne lorsque vous en sortez, vous pouvez le quitter.

Il faut maintenant lancer le `term` local :

```
term -c off -l $HOME/tlog -s 38400 -w 10 -t 150 < /dev/modem > /dev/modem &
```

Il faut indiquer à `term` où se trouve le modem. C'est pourquoi à la fois l'entrée et la sortie standards sont redirigées vers '`/dev/modem`' (c'est le rôle de $<$ et $>$). Enfin, la commande est lancée en arrière-plan, de façon à pouvoir continuer à utiliser le terminal pour autre chose, si on le souhaite.

`term` devrait maintenant fonctionner :-). Essayez un `trsh` et regardez ce qui se produit. Si cela plante ou si votre liaison semble excessivement lente, jetez un coup d'oeil aux fichiers '`tlog`' placés à chaque extrêmité. Contiennent-ils des *timeouts* ou des messages d'erreur ? Si c'est le cas, quelque chose ne va pas dans votre configuration. Essayez à nouveau (après avoir fini de lire ceci :-)). Notez qu'il est normal que la connexion ne semble pas rapide comme l'éclair, surtout si vous utilisez la compression. Les meilleures performances sont obtenues lors d'échanges tels que le transfert de fichiers.

4.11 Mettre fin à une connexion term

Il est probable qu'après avoir effectué un certain nombre de travaux, vous souhaiterez mettre fin à votre connexion `term`. Ceci peut être fait de quatre façons :

1. Tuer les programmes `term` à chaque extrêmité de la liaison. C'est la méthode la moins recommandée.
2. Une méthode un peu plus élégante est de taper la commande suivante sur la machine locale :

   ```
   echo '00000' > /dev/modem
   ```

   Ceci devrait mettre fin à votre connexion `term` proprement. Cette méthode fonctionne avec toutes les versions de `term`. Assurez vous que la suite de zéros en contient bien au moins *cinq*.
3. Dans le '`termrc`' des versions de `term` 2.0.0 et ultérieures, vous pouvez ajouter une entrée `terminate '<une chaîne de caractères>'`. Ceci indique la chaîne qu'utilisera `term` pour s'interrompre ('00000' par défaut). Elle doit être longue d'au moins cinq caractères afin d'éviter toute terminaison intempestive.
4. À partir de la version 1.14, un programme `tshutdown` est fourni (en fait, il s'agit d'un *patch* pour la version 1.14 et est dans le paquetage pour les versions plus récentes). `tshutdown` permet de mettre fin proprement à votre connexion `term`.

4.12 Effacer term

Bien, vous l'aurez voulu ! Puisque certains d'entre vous veulent se débarasser de `term`, nous indiquons ici les étapes à suivre.

- Effacez les répertoires et leur contenu. Suivant la façon dont vous avez installé `term`, un ou plusieurs des répertoires suivants est présent sur votre machine :

  ```
  $HOME/.term/termrc
  $HOME/.term/termrc.<server>
  $HOME/term/termrc
  $HOME/term/termrc.<server>
  /usr/local/lib/term/termrc
  /usr/local/lib/term/termrc.<server>
  /etc/termrc
  /etc/termrc.<server>
  ```

  Vous pouvez effacer simultanément ces répertoires et leur contenu. Pour ce faire, utilisez la commande `rm -rf`.
- Dans certains cas, vous avez créé un groupe *term* lors de l'installation. Vérifiez le fichier '`/etc/group`' et effacez la ligne qui concerne ce groupe, si elle s'y trouve.

- Il faut maintenant effacer le paquetage `term` ainsi que tous les exécutables modifiés pour `term`. C'est la partie la plus délicate. Les exécutables du paquetage de `term` doivent normalement se trouver dans le répertoire '`/usr/local/bin`' ou le répertoire '`$HOME/bin`'.

  En ce qui concerne les exécutables que vous avez modifiés vous-même pour une utilisation avec `term`, c'est à vous de voir, nous ne pouvons pas vous aider. Pensez également à effacer les fichiers de configuration ou autres qui correspondent à ces programmes.

- Pour effacer les bibliothèques, lancez les commandes suivantes :

  ```
  cd /
  find . -name 'libtermnet*' -exec /bin/rm {}
  ```

  Ceci localisera et effacera les bibliothèques.

- Pour effacer le fichier 'termnet.h', le plus simple est de taper :

  ```
  cd /
  find . -name termnet.h -exec /bin/rm {}
  ```

- Si vous avez installé les pages de manuel de `term`, elles doivent se trouver dans l'un des répertoires suivants :

  ```
  /usr/local/man/man1
  /usr/local/man/cat1
  $HOME/man/man1
  $HOME/man/cat1
  ```

  Il vous faut au minimum contrôler la présence des pages de manuel suivantes : `term`, `term_clients`, `term_setup`, `tdownload`, `linecheck`, `trdate`, `trdated`, `termrc`, `termtest`, `tmon`, `tredir`, `trsh`, `tshutdown`, `tudpredir`, `tupload`, `txconn` et finalement `tiptest`.

- Effacez le répertoire utilisateur temporaire et son contenu. Il s'agit du répertoire '`/usr/tmp/private`'.

Une fois ce petit exercice terminé, vous pouvez être quasiment certain d'avoir effacé tout ce qui concerne `term`.

4.13 Optimisation de la liaison

Une fois que vous avez réussi à faire fonctionner `term`, pourquoi ne pas essayer d'optimiser un peu les choses ? Un bon moyen de mesurer la vitesse de transmission de votre liaison est de lancer `tmon` dans une fenêtre tout en téléchargeant un fichier dans une autre. Essayez avec de gros fichiers de texte et avec des fichiers compressés. Les fichiers textes devraient être transférés environ deux fois plus vite que les fichiers compressés. Les paramètres sur lesquels vous pouvez jouer sont les *bps* [3] (`-s`), la compression (`-c`), la taille de fenêtre (`-w`), les *timeouts* (`-t`) et le recyclage (`-A`).

3. (N.D.T.) Le document original parle de *baud*, mais il s'agit d'une erreur commune :-)

Attention au paramètre de recyclage. Avec `term` version 1.19, l'auteur a obtenu une perte de 80% à 90% lorsqu'il a utilisé ce paramètre. Peut-être est-ce une bogue de la version 1.19 et n'est-ce valable que pour cette version.

`bps` : il s'agit du nombre maximal de bits que `term` tentera d'envoyer sur la liaison série. `term` évitera d'envoyer des caractères plus rapidement. Par défaut, utilisez la vitesse des ports série de votre ordinateur, mais soyez conscient que cela risque d'être trop rapide si votre modem fonctionne à une vitesse plus faible sur la ligne téléphonique. Cette option est prévue pour les systèmes qui placent les données dans un tampon. Lors de l'installation et de la mise au point, il est conseillé d'utiliser des valeurs faibles plutôt que des valeurs trop importantes. Pour des liaisons à haute vitesse (> 38400), il peut être avantageux de ne pas imposer de limite en utilisant la valeur '`off`'. `term` ne s'appuiera alors que sur le noyau pour effectuer le contrôle de flux.

`Compression`: à utiliser si votre modem n'effectue pas lui-même la compression. S'il la fournit, désactivez cette option afin de ne pas compresser deux fois les mêmes données, ce qui *augmente* à coup sûr le volume de données transmis. Les modems assurant la compression sont ceux qui utilisent les protocoles `MNP-5` ou `V42.bis`. Consultez la documentation de votre modem et les messages qu'il envoie lors d'une connexion.

`Fenêtre`: c'est le nombre de paquets de données que `term` accepte d'envoyer sur la liaison avant de recevoir un acquittement de la part du `term` distant. Avec un modem rapide, il peut être avantageux d'augmenter ce paramètre. Pour les liaisons plus lentes, cela risque de surcharger l'extrêmité distante.

`Timeout`: il s'agit de la durée pendant laquelle `term` attend un acquittement. Si vous avez augmenté la taille de la fenêtre et obtenez de nombreux *timeouts* dans vos fichiers de *log*, essayez d'augmenter ce paramètre.

Pour un modem `14400/V42.bis`, l'auteur utilise `-c off -w 10 -t 150` et obtient environ 1700 *cps* pour les fichiers compressés et 3500 *cps* pour les fichiers texte, les transfert étant effectués avec `tupload`.

4.14 Problèmes classiques

Dans cette partie sont exposées quelques démarches classiques à suivre lorsque l'utilisation de `term` ou de l'un de ses clients vous pose problème.

- Avez-vous bien nettoyé l'arborescence de `term` ? L'apparition de nouvelles versions de `term` est allée de pair avec de nombreuses modifications de l'arborescence '`/usr/local/lib/term`'. Si vous n'êtes pas conscient de cela, vous risquez de voir apparaître de nombreux messages d'erreur. Le mieux dans ce cas est d'effacer l'arborescence '`/usr/local/lib/term`' (en prenant soin de sauvegarder votre '`termrc`') et d'installer ensuite votre nouvelle version de `term`. Ainsi, vous devriez éviter de vous créer des problèmes avec une arborescence inadéquate.
- Avez-vous effacé les vieilles *sockets* ? Lors de la mise à jour de `term`, effacez toutes les *sockets* (appelées '`socket=`') créées par `term`. Ne pas le faire vous expose à d'étranges problèmes. Afin de déterminer à l'écoute de quelle *socket* se trouve `term`, vous pouvez utiliser le programme `netstat`.
- `term` ne se compile pas corectement sous `SunOS 4.1.3` ? C'est que vous avez probablement configuré `term` avec `./configure --user`. Lors de la compilation, vous voyez apparaître une erreur d'assembleur à propos d'une option '`-k`'. La cause de cette erreur est inconnue. La solution est de configurer `term` avec des bibliothèques statiques. Il vous faut donc utiliser `./configure --user --static` et continuer la compilation de façon normale. Tout devrait alors se dérouler normalement.

- `termtest` vous donne une erreur : '`term: failed to connect to term socket '/root/.term/sockettest'`'? Lorsqu'il est lancé, `termtest` s'attend à trouver l'exécutable `term` dans le répertoire où il est lui-même. Lorsque vous faîtes un `make install` avant de lancer `termtest`, l'exécutable `term` est déplacé dans '`/usr/local/bin`' (ou dans un autre répertoire d'exécutables).

  La solution consiste à créer un lien vers l'exécutable dans le répertoire source :

  ```
  ln -s /usr/local/bin/term /usr/src/term-<numero_version>/term
  ```

- Lancez-vous bien le bon exécutable ? `term` a été mis à jour à de nombreuses reprises et bon nombre de systèmes en possèdent différentes versions. Assurez-vous d'utiliser la bonne version. Notez que cela s'applique également à `linecheck`. Vous pouvez utiliser la commande `type -a` de `bash` ou bien la commande `whereis` pour déterminer quel est le programme qui est exécuté. Les versions de `term` postérieures à la 1.11 affichent normalement leur numéro de version au démarrage (bien que la 1.14 prétende n'être que la 1.12. Et oui !).

- Avez-vous mis le bon '`termrc`' au bon endroit ? Suivant les versions de `term` que vous utilisez et la façon dont vous l'avez installé (en tant que *root* ou en tant qu'utilisateur), ce fichier peut se trouver dans l'un des répertoires suivants :

  ```
  $HOME/.term/termrc
  $HOME/.term/termrc.<server>
  $HOME/term/termrc
  $HOME/term/termrc.<server>
  /usr/local/lib/term/termrc
  /usr/local/lib/term/termrc.<server>
  /etc/termrc
  /etc/termrc.<server>
  ```

  Certains systèmes sont fournis avec des fichiers '`termrc`' pré-installés. Assurez-vous de les avoir détruit avant de procéder à l'installation. Si vous faîtes l'installation en tant que *root*, vérifiez '`/.term`'. Lors de son exécution, `term` crée des fichiers (qui sont en fait des *sockets*). C'est pourquoi il possède son propre répertoire '`$HOME/.term`' dans lequel est placé le fichier '`termrc`' (notez qu'il n'y a PAS de . devant '`termrc`' !).

- `term` trouve-t-il son fichier '`termrc`' ? Lorsque vous lancez `term` des deux côtés, vous devriez voir des messages tels que :

  ```
  term version: 2.2.9
  Reading file:  /usr/local/lib/term/termrc
  Using shared mode.
  ```

  Si la seconde ligne n'apparaît pas, c'est que `term` ne trouve pas le fichier '`termrc`' et vous pouvez être sûr que quelque chose ne s'est pas déroulé normalement lors de l'installation (sauf si vous n'utilisez pas de fichier '`termrc`' et entrez tous les paramètres dans la ligne de commande :-)). Vérifier la localisation et les droits du fichier '`termrc`' sur le système où `term` ne le trouve pas.

– `term` ou votre répertoire '`.term`' est-il monté par `NFS` ? Si tel est le cas, il faut utiliser l'option `-Dterm_NFS_DIR` dans la ligne `CFLAGS` du '`Makefile`'. Malheureusement, en ce qui concerne l'auteur, ce paramètre provoque une erreur de compilation lorsque `term 1.19` est compilé sur une machine sous `SunOS 4.*`.

– Tous les fichiers et répertoires ont-ils les bons propriétaires et des droits adéquats ? Cela ne devrait pas être un problème dans la mesure où les propriétaires et les droits sont déterminés lors de l'installation. Néanmoins, soyez conscient de ce fait si vous effectuez le portage pour `term` de vos propres applications. Si de plus vous modifier le mode d'exécution de `term` (c'est-à-dire du mode *privé* vers le mode *partagé*), il faut changer en conséquence les propriétaires et les droits des fichiers.

– Si vous obtenez une erreur `gethostbyname: <hostname>: Non-authoritative 'host not found'`, ou `server failed`, il vous faut vérifier les points suivants :

1. Le fichier '`/etc/hosts`' est-il configuré correctement ? Si `<hostname>` n'est pas le nom de votre machine (les vieilles distributions *SLS* et les distributions *Slackware* sont fournies avec '`darkstar`' pour nom de machine, par exemple), il vous faut modifier ce fichier. Il doit au minimum contenir une ligne telle que :

```
# Local Hosts Format:
#  IP_NUMBER            HOSTNAME        ALIASES
#
# Here is the name of your host, first, followed by any aliases
127.0.0.1       localhost       linuxpc.domain  linuxpc
```

2. Vos fichiers '`/etc/rc*`' et '`/etc/resolv.conf`' sont-ils lisibles (`chmod ugo+r`) ?
3. Vérifez enfin que vous avez installé l'interface `TCP/IP loopback` sur votre machine.

– Les fichiers de *log* de `term` sont remplis de messages '`timed out`' de tous types ? Cela signifie que votre connexion n'est pas optimisée. Quelques messages de ce type de temps en temps ne posent guère de problème. Ils sont sans doute dus à des facteurs temporels influant sur la qualité de la connexion physique entre vos machines locale et distante.

  Si, en revanche, le nombre de ces messages est très élevé, votre connexion sera fortement ralentie. Il vous faut alors jouer sur les paramètres présentés dans la partie '*Optimisation de la liaison*'. C'est malheureusement la partie la plus expérimentale de l'installation. Aucune règle définitive concernant les valeurs des paramètres ne peut être donnée, tant les facteurs susceptibles d'influencer la qualité de la connexion sont nombreux. Ces facteurs varient d'une connexion à l'autre et même d'un moment à l'autre pour une même connexion.

– Un `ftp` normal avec des ports redirigés ne fonctionne pas chez vous ? C'est un problème connu : la redirection des ports requis par `ftp` (les ports 20 et 21) ne permet pas à `ftp` de fonctionner. La seule solution est de récupérer un `ftp` modifié pour fonctionner avec `term`. Malheureusement, même de telles versions de `ftp` semblent ne pas fonctionner.

5 Les clients term

Un certain nombre d'utilitaires sont fournis dans le paquetage `term`, en particulier : `trsh`, `tmon`, `tupload`, `tredir`, `txconn` et, dans les versions les plus récentes, `trdate` et `trdated`. De plus, depuis la version 2.1.0, `tdownload` s'est ajouté à cette liste. Cette partie traite de `trsh`, `tmon`, `tupload`, `tdownload`, `trdate` et `trdated`. Les autres clients sont décrits dans des parties qui leur sont propres. Aucun client `term` ne peut fonctionner tant que vous n'avez pas mis en route une liaison `term`.

`tmon` est un utilitaire simple qui vous permet de surveiller les performances de votre liaison. Il imprime un histogramme des caractères envoyés et reçus. On le lance par la simple commande `tmon`. Attention, les versions 1.11 et ultérieures comportent une bogue qui tronque certaines données.

`trsh` est similaire à `rsh`. Sans argument, il lance un *shell* interactif sur le système distant (autrement dit, il vous y loge). `trsh` est un des moyens de base pour accéder à l'extrémité distante de la liaison `term`. Si un argument est passé, celui-ci est exécuté comme commande sur le système distant. Par exemple :

```
trsh ls
```

renvoie la liste des fichiers de votre répertoire utilisateur (`$HOME`) sur le système distant.

`tupload` transfère un fichier dont le nom est passé en argument d'une extrémité à l'autre de la liaison. Par défaut, ce fichier est placé dans le répertoire d'où vous avez lancé `term` (de l'autre côté de la liaison). Pour placer les fichiers dans un autre répertoire, donnez son nom comme second argument. Par exemple, si vous souhaitez copier le fichier '`term114.tar.gz`' dans le répertoire '`/usr/tmp`' de la machine distante, tapez :

```
tupload term114.tar.gz /usr/tmp
```

Lorsque vous utilisez `tupload`, vous avez à votre disposition des *jokers* comme, par exemple, dans la commande '`tupload a.*`'. Le *shell* se charge d'effectuer les correspondances et la commande effectivement exécutée est '`tupload a.1 a.2 ......`'.

`tdownload` permet le transfert d'un fichier dont le nom est passé en argument de la machine distante vers la machine locale. Par défaut, les fichiers seront placés dans le répertoire duquel vous avez lancé `term` sur la machine locale. Pour placer les fichiers dans un autre répertoire, passez son nom comme second argument à `tdownload`. Par exemple, si vous souhaitez copier le fichier '`term114.tar.gz`' dans '`/usr/tmp`' sur la machine locale, tapez :

```
tdownload term114.tar.gz /usr/tmp
```

Il n'est PAS possible d'utiliser les *jokers* avec `tdownload` pour la simple raison que le répertoire distant n'est pas accessible au *shell* et que toute expansion de *jokers* se réfère au répertoire courant sur la machine locale (où est exécuté le *shell*).

`trdate` est un utilitaire qui lit l'heure sur la machine distante et la modifie en conséquence sur la machine locale. Il doît être exécuté en tant que *root*.

`trdated` est la version démon de `trdate`. Lorsqu'il est lancé dans '`rc.local`', il est exécuté comme un démon et met l'heure à jour toues les 5 minutes (par défaut). `trdated` fonctionne même si aucune connexion `term` n'est active; il commence à modifier l'heure dès que la connexion est effectivement établie.

6 X et term

`term` permet à plusieurs utilisateurs d'ouvrir des fenêtres *X windows* sur la machine locale, à partir de clients s'exécutant sur une machine du réseau. C'est le rôle de `txconn`, qui est lancé sur le système distant par la simple commande `txconn`. Il se place en arrière-plan et renvoie un nombre sur la sortie standard. Ce nombre est le numéro de *display* que doivent utiliser les clients pour accéder au serveur `X` de la machine locale. Voici un exemple qui devrait clarifier les choses. Nous sommes connectés via `trsh` à l'hôte `term` distant, nommé `toto`. Sur `toto`, nous lançons les commandes suivantes :

```
toto$ txconn
Xconn bound to screen 10
:10
toto$
```

Désormais, sur chaque hôte dont nous voulons qu'il exécute un client `X` et envoie l'affichage sur la machine locale, il suffit de faire :

```
setenv DISPLAY toto:10
```

(en `bash` utiliser `export DISPLAY=toto:10`). Dans certains cas, il sera de plus nécessaire de faire un :

```
xhost + toto
```

ou même un :

```
xhost +
```

sur la machine locale.

Maintenant, à chaque fois que nous démarrons un client, il tente de se connecter au *display* 10 de la machine `toto`. Comme `txconn` est à l'écoute de ce *display* et retransmet tous les paquets du protocole `X` vers le serveur `X` de la machine locale, via `term`, la fenêtre s'ouvrira sur la machine locale.

Il est également possible de faire le contraire, à savoir de lancer un client sur la machine locale et d'envoyer l'affichage sur la machine distante. Nous ne détaillerons cependant cette procédure qu'après avoir présenté `tredir`.

Le protocole `X` n'est pas particulièrement efficace et gaspille une grande largeur de bande. C'est rarement un problème dans le cas d'un réseau *éthernet*, mais cela devient désastreux pour une liaison par modem. Il paraît que `X11R6` met en place une version à faible largeur de bande du protocole, `LBX`. Si cependant vous utilisez `X11R5`, vous pouvez avoir recours à un utilitaire appelé `sxpc` qui compresse le protocole `X` et améliore ainsi les temps de réponse lors de l'utilisation de liaisons séries. Le paquetage `sxpc` contient un article sur son fonctionnement avec `term` et en est d'autant plus recommandé.

7 tredir

tredir est l'un des utilitaires les plus puissants parmi ceux fournis avec term. Il permet d'utiliser avec term les plus importants services réseau. Avant d'en expliquer l'usage, il est nécessaire de donner quelques bases sur le fonctionnement de ces services. Nous les avons déjà évoqués, mais sans expliquer précisément en quoi ils consistent. Comme leur nom l'indique, ce sont des services qui sont fournis par le réseau. Des exemples de tels services sont : telnet qui permet de se *loger* sur une machine à partir d'une autre, le protocole de transfert de fichiers ftp qui permet l'envoi de fichiers entre machines et smtp (*Simple Mail Transfer Protocol*) qui est sollicité à chaque fois que vous envoyez du courrier électronique. Chaque service réseau est associé à un numéro de port. L'affectation des numéros de ports aux différents services est donnée par le fichier '/etc/services' qui doit être identique sur toutes les machines reliées à un même réseau.

Comment ces services sont-ils appelés ? Sur chaque machine du réseau tourne un démon nommé inetd qui est à l'écoute des tentatives de connexion sur les ports réseau. Ces tentatives proviennent soit du réseau, soit de la machine locale. On obtient un service réseau donné en se connectant au port inetd adéquat. Lorsqu'une requête de service réseau est envoyée, inetd sait exactement quel service est demandé grâce au numéro du port sollicité. S'il est configuré pour, il fournit alors le service adapté à la connexion qui le demande. La configuration d'inetd est donnée par le fichier '/etc/inetd.conf' qui contient la liste des services fournis. Pour plus d'informations, reportez-vous aux pages de manuel d'inetd et inetd.conf.

Il est possible de communiquer directement avec les services réseau en utilisant telnet (et non termtelnet). Par exemple, pour dialoguer avec le démon sendmail (smtp) de la machine nom_de_la_machine, vous pouvez utiliser telnet nom_de_la_machine smtp ou bien telnet nom_de_la_machine 25 (puisque 25 est le numéro de port affecté à smtp dans '/etc/services'). Vous devriez alors recevoir un gentil message de bienvenue de la part du démon de la machine distante. C'est une astuce très utile pour résoudre les problèmes réseau et contrôler les ports redirigés par tredir (voir ci-après).

tredir fonctionne de façon très similaire à inetd. Il s'exécute en arrière-plan comme un démon et est à l'écoute des différents ports, dans l'attente d'une requête. Lors qu'une demande de service est faite, plutôt que de fournir ce service comme le fait inetd, tredir retransmet la requête au term distant qui transmet la requête au réseau et renvoie le résultat à la machine locale. tredir peut envoyer la requête à n'importe quelle machine du réseau, mais la dirige par défaut vers la machine placée à l'autre extrémité de la liaison term. tredir REDIRige les services réseau TCP (*Transimssion Control Protocol*).

La syntaxe classique de tredir est la suivante :

```
tredir [cet_ordinateur_ci:]port [cet_ordinateur_la:] port
```

Un exemple devrait clarifier tout ceci. Redirigeons un port local sur le port telnet de la machine distante. Pour ce faire, nous utilisons :

```
tredir 2023 23
```

Maintenant, quiconque se connecte au port 2023 de la machine locale sera redirigé vers le port 23 (telnet) de la machine distante. Voici un exemple de session où la machine locale est ma_machine.modem.maison et la machine distante se nomme netsun.

```
$ tredir 2023 23
```

```
Redirecting 2023 to 23
$ telnet localhost 2023
Trying 127.0.0.1...
Connected to ma_machine.modem.maison
Escape character is '^]'.

SunOS UNIX (netsun)
login:
```

Cet exemple est particulièrement utile. Si nous avions fait le `tredir` sur `netsun`, nous pourrions alors faire un `telnet` vers `ma_machine` depuis le réseau, simplement en nous connectant au port redirigé de `netsun`, *i.e.* `telnet netsun 2023`.

Le principe général de l'utilisation de `tredir` est de rediriger le service souhaité vers une machine du réseau. L'exemple suivant nous permet de lire les *news* sur la machine locale à partir d'un serveur de *news* du réseau, en utilisant la liaison `term`. Les *news* sont fournies par le service `nntp` dont le numéro de port est le 119. Tout lecteur de *news* digne de ce nom vous permet de spécifier le numéro de port à utiliser, grâce à un fichier de configuration ou une variable d'environnement. Donnons à ce port local le numéro 2119 et supposons maintenant que notre serveur de *news* est `news.domain.org`. Nous allons rediriger le port 2119 vers le port 119 de `news.domain.org`, puis nous indiquerons à notre lecteur de *news* que le serveur `nntp` se trouve sur le port 2119 de la machine locale. Etant donné que cela dépend du lecteur de *news* utilisé, nous allons juste tester la liaison avec `telnet` plutôt que de vraiment lancer un tel programme.

```
$ tredir 2119 news.domain.org:119
Redirecting 2199 to news.domain.org:119
$ telnet localhost 2119
Trying 127.0.0.1...
Connected to ma_machine.modem.maison.
Escape character is '^]'.
200 news.domain.org InterNetNews NNRP Server INN 1.4 07-Dec-41 ready
(posting ok).
```

Si vous parvenez jusqu'ici, tout ce qu'il vous reste à faire est de configurer votre lecteur de *news* pour être capable de lire les *news* via `term`. N.B. : si vous lisez les *news* de cette façon, soyez certain que dans tous vos envois, vous spécifiez dans la ligne d'en-tête `Reply-To:` une adresse *email* du réseau à laquelle on peut vous joindre, sinon les gens qui voudront vous contacter enverront leur courrier à l'adresse (fantaisiste) constituée des données quelconques que votre lecteur de *news* placera dans la ligne d'en-tête `From:`.

7.1 `tredir` : Attention, chien méchant !

Le lecteur astucieux, après avoir lu le dernier exemple, se demandera certainement pourquoi le port 2119 a été redirigé vers le port 119. Etant donné que les programmes de lecture de *news* utilisent par défaut le port 119, pourquoi ne pas faire un `tredir 119 news.domain.org:119` et éviter la configuration du lecteur ? La réponse est que tous les ports de numéro inférieur à 1024 sont des ports *réservés*, à l'écoute desquels seuls un super-utilisateur peut se mettre. Si vous prenez le risque de *suider* `tredir` ou si vous le lancez en tant que *root*, vous pourrez alors rediriger les ports réservés et éviter de vous casser les pieds avec les services restants.

Un autre inconvénient de l'utilisation des ports réservés est qu'`inetd` les écoute fréquemment et qu'un port ne peut être écouté que par un seul programme à la fois. Afin d'utiliser un tel port, il vous faut modifier le fichier `inetd.conf` de façon qu'`inetd` ne soit plus à l'écoute du port que vous souhaitez rediriger. Pour ce faire, commentez la ligne du service incriminé, en plaçant un # au début de cette ligne. Le super-utilisateur doit alors envoyer un signal HUP à `inetd` pour qu'il relise son fichier de configuration.

7.2 Quelques astuces `tredir`

Dans cette partie, nous allons présenter quelques-unes des utilisations les plus courantes de `tredir`. Nous avons déjà indiqué comment rediriger les services `nntp` et `telnet` et allons maintenant passer à des exemples un peu plus compliqués.

7.2.1 `X Windows`

Dans une précédente partie, nous avons expliqué comment permettre à un client s'exécutant sur le réseau d'ouvrir une fenêtre sur la machine locale, grâce à `txconn`. La même technique peut être utilisée sur votre machine locale pour qu'un client envoie son affichage vers l'autre extrémité de la liaison `term`. Mais comment afficher un client `X` sur une machine qui n'est pas l'autre bout de la liaison ? La réponse réside dans la connaissance du fait qu'`X` utilise un service réseau particulier, à l'instar des autres programmes que nous avons évoqués. Un serveur `X` est à l'écoute de requêtes réseau sur un port dont le numéro est donné par la formule : $port = 6000 + numero\,de\,display$. Par exemple, un serveur `X` gérant le *display* 0 sur une machine donnée sera à l'écoute du port 6000. S'il gère le *display* 2, il écoutera le port 6002. Lorsque vous positionnez votre variable d'environnement `DISPLAY` sur `machine:n`, vos clients `X` tenteront de se connecter au port $6000 + n$ de `machine`.

Nous pouvons utiliser cette astuce pour permettre à des clients `X` de votre machine locale d'ouvrir des fenêtres sur des *displays* distants. Supposons que nous voulons lancer un `xterm` tournant sur la machine locale avec l'affichage sur le *display* 0 de la machine `xmachine`, qui est quelque part sur le réseau. Premièrement, choisissons un numéro de `display` : 2, par exemple (n'utilisez pas 0, c'est celui que votre serveur `X` local utilise). Faisons-en le *display* 0 de `xmachine`. En termes de ports, cela signifie qu'il nous faut rediriger le port local 6002 vers le port distant 6000. Les commandes sont les suivantes :

```
$ tredir 6002 xmachine:6000
$ setenv DISPLAY localhost:2
$ xterm
```

Ceci devrait ouvrir un `xterm` sur la machine `xmachine`. Notez que nous avons positionné `DISPLAY` sur `localhost:2`. La justification en est que les clients `X` utilisent parfois des *sockets unix domain* plutôt que des *sockets internet domain* lorsqu'ils se connectent à un *display* local (lorsque `DISPLAY` contient `:2`, par exemple). `localhost:2` demande l'utilisation d'une connexion `TCP`.

En ce qui concerne `xmachine`, la requête `X` provient de la machine placée à l'extrémité distante de votre liaison `term` (`remotemachine`). C'est pourquoi, si vous voulez autoriser la connexion, vous devez utiliser soit `xhost + remotemachine` sur `xmachine`, soit `xauth` pour mettre à jour le fichier `.Xauthority` de votre machine locale pour le *display* 2, en utilisant la clé de `xmachine`.

Une fois encore, pour accélérer les connexions `X`, vous pouvez utiliser le programme `sxpc` pour établir la liaison. `sxpc` contient des explications sur la façon d'utiliser `tredir` et de l'autoriser, en utilisant `xauth`.

7.2.2 term et le mail

Bien, vous l'aurez voulu ! Le courrier électronique a la réputation justifiée d'être l'un des services les plus difficiles à mettre en place sur les machines `UNIX`. Faire fonctionner `term` avec le courrier électronique implique une bonne compréhension des mécanismes de ce dernier, ce qui déborde largement le cadre de ce document. Pour en savoir plus sur le courrier électronique, procurez-vous un ouvrage sur l'administration de systèmes `UNIX` et/ou le `FAQ` (*frequently asked questions*) du *newsgroup* `comp.mail.misc`, disponible par `ftp` anonyme sur `rtfm.mit.edu`, dans le répertoire '`/pub/usenet/comp.mail.misc`'. Il existe actuellement deux paquetages qui vous permettront d'utiliser le courrier électronique avec `term`. Tous deux sont accessibles par `ftp` anonyme sur `sunsite.unc.edu`. Il s'agit de `term.mailerd+mail` de Byron A. Jeff et de `BCRMailHandlerXXX` de Bill C. Riemers.

Ceci étant dit, voici une description succinte de la façon dont le courrier électronique fonctionne. Deux parties sont à considérer : l'envoi et la réception de messages. Commençons par envoyer des messages de votre machine locale vers le réseau.

Il y a deux sortes de programmes de courrier électronique. Les premiers sont appelés `MUA` (*mail user agent*) et permettent de lire, composer et envoyer des messages. Parmi les `MUA` les plus connus, on trouve `elm` , `pine`, `Mail` et `vm`. Les `MUA` ne font aucun travail de réseau, ils se contentent d'assembler les messages. Le vrai travail d'envoi des messages est réalisé par le second type de programmes, à savoir les `MTA` (*mail transfer agents*) qui sont appelés par les `MUA`. Ils prennent les messages, décident de l'endroit où il faut les envoyer (en fonction de l'adresse fournie) et les envoient effectivement sur le réseau.

Les deux `MTA` les plus courants sur les systèmes `Linux` sont `sendmail` et `smail`. L'idée de base et de faire se connecter votre `MTA` à un autre `MTA` tournant sur une machine du réseau, laquelle saura quoi faire de vos messages. Pour cela, il faut rediriger un port local sur le port `smtp` de la machine placée sur le réseau. Ensuite, indiquez à votre `MTA` qu'il doit envoyer à celui du réseau tout message qu'il ne sait pas traiter. Celui-ci saura alors comment envoyer le message vers sa destination.

Comment cela se passe-t-il avec `smail` ? Tout d'abord, on redirige un port vers le port `smtp` de la machine du réseau (`mailhost`) :

```
tredir XXXX mailhost:25
```

où XXXX est le numéro du port local auquel `smail` va se connecter (notez qu'il est nécessaire de donner un nom à ce port dans le fichier '`/etc/services`' pour que `smail` le reconnaisse). `smail` possède plusieurs fichiers de configuration généralement placés dans '`/usr/local/lib/smail`'. Ceux qui nous intéressent sont '`config`', '`routers`' et '`transports`'. Notez bien que nous supposons que vous avez déjà configuré `smail` correctement pour traiter vos messages locaux. Une fois de plus, consultez la documentation si ça n'est pas le cas.

Dans le fichier 'config', il faut placer la définition suivante :

```
smart_path=localhost
```

localhost étant la machine à laquelle smail se connecte lorsqu'il ne sait pas quoi faire d'un message.

Dans le fichier 'routers', ajoutez :

```
smart_host:
driver=smarthost,
transport=termsmtp;
path = localhost
```

Enfin, ajoutez dans 'transports' :

```
termsmtp: driver=tcpsmtp,
          inet,
          return_path,
          remove_header="From",
          append_header="From: VOTRE_ADRESSE_RESEAU",
          -received,
          -max_addrs, -max_chars;
          service=VOTRE_SERVICE_SMTP,
```

Dans les quelques lignes ci-dessus, les lignes d'en-tête (*header*) changent tous les en-têtes des messages qui partent de chez vous en l'adresse VOTRE_ADRESSE_RESEAU, qui est l'adresse réseau à partir de laquelle vous voulez que les messages semblent être envoyés. Si plusieurs utilisateurs utilisent votre liaison term, il vous faudra avoir recours à quelque chose de plus acrobatique, comme maintenir une base de données des adresses réseau des utilisateurs locaux et les insérer dans les en-têtes From:.

La ligne service est le nom du port local que vous avez redirigé vers le port smtp de la machine connectée au réseau. Avec la version de smail qu'utilise l'auteur, il est impossible d'utiliser juste un nombre. Il est nécessaire de mettre un nom tel que « toto » et de définir « toto » dans le fichier '/etc/services' comme étant le numéro du port redirigé. Si vous utilisez un tredir *suidé* et ne faites que rediriger le port smtp (25), il n'est nul besoin de définir ce « toto ».

Cela devrait suffire pour vous mettre sur la voie. Si vous décidez d'utiliser sendmail, les principes sont les mêmes, bien que les détails diffèrent. Ronald Florence (ron@mlfarm.com) a affirmé à l'auteur que le sendmail de SUN ne peut envoyer une file de plusieurs messages vers le port redirigé, mais que le sendmail BSD 8.6.9 fonctionne correctement. Ronald a appliqué à sendmail les modifications suivantes de façon à le faire fonctionner avec term. Dans son cas, le port sendmail par défaut (25) est utilisé pour le trafic local d'un réseau *éthernet* et le courrier électronique *Internet* est donc transmis à un port TCP redirigé.

```
#
```

```
# Creation du mailer termsmtp qui envoie le courrier via un port TCP redirige
#
Mtermsmtp,P=[TCP], F=mDFMuCXe, S=22, R=22, A=TCP $h PORTNUMBER
```

Ici, `PORTNUMBER` est le numéro du port redirigé de la machine locale. Cela doit être un port inutilisé de numéro supérieur à 2000. Il faut ensuite indiquer à `sendmail` à quelle machine se connecter et faire de `termsmtp` le *mailer* par défaut.

```
#
# Principal mailer de relai
#
DMtermsmtp
#
# Principal hote de relai : utilise le mailer $M pour envoyer
# le courrier vers d'autres domaines
#
DR HOSTNAME
CR HOSTNAME
```

Où `HOSTNAME` est le nom de votre machine locale (est-ce que `localhost` fonctionne ?). La dernière ligne suit la règle 0 pour retransmettre le courrier *Internet*.

```
# Envoyer les autres noms valides au forwarder
R$*<@$*.$+>$*   $#$M    $@$R    $:$1<@$2.$3>$4  user@any.domain
```

Lorsque la connexion `term` avec l'hôte *Internet* est établie, lancez les commandes suivantes sur la machine locale.

```
tredir PORTNUMBER internet.host:25
/usr/lib/sendmail -q
```

Nous allons maintenant nous préoccuper de recevoir du courrier électronique à l'aide de `term`. Nous supposerons que le courrier est envoyé à votre compte sur la machine `mailhost` connectée au réseau. La solution la plus simple est d'utiliser `trsh` ou `termtelnet` pour se connecter à `mailhost` et d'y lire votre courrier. Il est cependant possible de télécharger automatiquement votre courrier sur la machine locale, grâce à `POP` (*post office protocol*). `POP` a été précisément conçu dans ce but : envoyer le courrier électronique à des machines n'ayant que des accès intermittents au réseau. Pour utiliser `POP`, il faut qu'il y ait un serveur `POP` installé sur `mailhost`. Si tel est le cas, vous pouvez utiliser un client `POP` pour télécharger votre courrier régulièrement. Comme on pouvait s'y attendre, `tredir` joue un rôle dans la procédure. Le port utilisé par le service `POP` est le 110 (*n.b.* il existe un ancien protocole `POP-2` qui utilise le port 109. Nous parlons ici de `POP-3` qui est la dernière version de `POP`). Plusieurs clients `POP` sont disponibles. L'un d'eux, écrit en langage `perl` par William Perry et maintenu par l'auteur, est `pop-perl-1.X`. On peut le trouver sur `sunsite` dans `/pub/Linux/system/Mail`.

Pour utiliser POP, il faut rediriger un port local vers le port 110 de mailhost et configurer le client de façon à ce qu'il récupère le courrier sur localhost en utilisant le port local. Supposons par exemple qu'il y a un serveur POP qui tourne sur mailhost. Nous redirigeons le port local 2110 et lançons le client pop-perl.

```
$ tredir 2110 mailhost:110
Redirecting 2110 to mailhost:110
$ pop
Username: bill
Password: <votre mot de passe pour mailhost>
Pop Host: localhost
Pop Port: 2110
Starting popmail daemon for bill
```

Si vous n'avez pas de serveur POP à votre disposition, le paquetage BCRMailHandler contient un programme permettant de télécharger le courrier électronique vers votre machine locale grâce à term. L'auteur ne l'a jamais utilisé, mais quiconque le possède est encouragé à le commenter. Vous pouvez également utiliser le paquetage term.mailerd+smail. Malheureusement, aucun de ces deux paquetages ne fonctionne avec les versions 2.0.0 et ultérieures de term.

8 tudpredir

tudpredir est très semblable à tredir dans ce qu'il fait et dans la façon dont il est exécuté. La grosse différence entre les deux programmes est que l'un (tredir) sert à rediriger vers la connexion term les services réseau TCP, alors que l'autre (tudpredir) permet de rediriger les services réseau UDP (*User Datagram Protocol*). Une autre différence importante est que tredir se place automatiquement en arrière-plan une fois qu'il a établi la communication avec le port local, alors que tudpredir doit être mis en arrière-plan manuellement.

La syntaxe classique de la commande tudpredir est :

```
tudpredir [cette_machine_ci:]port [cette_machine_la:]port
```

9 Automatisations

Maitenant que vous savez comment obtenir de term tous vos services réseau, il peut être agréable de mettre en place une initialisation et une configuration automatiques de votre connexion. Il y a à peu près une infinité de façons d'y parvenir, selon le programme de communication que vous utilisez et la façon dont vous vous connectez au système distant.

Il existe un programme que l'auteur n'a pas utilisé, mais dont il a entendu dire le plus grand bien. Il s'agit de fet, un frontal pour term. Il est conçu pour vous connecter à un système distant et lancer term et tous vos tredir. Tout commentaire sur fet est le bienvenu.

Voici maintenant un exemple d'un ensemble de commandes qui utilise `kermit` pour se connecter au système distant et effectuer toutes les initialisations de `term`. Il est clair qu'il vous faudra modifier ces exemples pour les adapter à votre procédure de connexion.

La commande qui est effectivement lancée est le script shell 'knet' :

```
#!/bin/sh
/usr/bin/kermit -y $HOME/.kerm_term > $HOME/klog < /dev/null 2>& 1
exec $HOME/bin/tstart >> $HOME/klog 2>& 1
```

Le script '.kerm_term' est le suivant :

```
pause 2
# Le numero a composer
output atdtXXXXXXX \13
# Connexion au serveur de terminal
input 145 {name: }
output MYNAME \13
input 3 {word: }
output MYPASSWORD \13
input 5 {xyplex>}
# Rendre la ligne transparente
output term telnet-t \13
output term stopb 1 \13
# Connexion au systeme distant.
output telnet hotedistant.domainequelconque.org \13
input 10 {ogin: }
output MYOTHERNAME \13
input 3 word:
output MYOTHERPASSWORD \13
pause 5
# On lance le term distant
output exec term -s 38400 -l $HOME/tlog -w 10 -t 150  \13
! /usr/bin/term -r -l $HOME/tlog -s 38400 -c off -w 10 -t 150 < /dev/modem > /dev/modem &
# Lancer les autres clients ici si necessaire
suspend
!killall -KILL term
```

Finalement, le script 'tsart' qui lance les clients `term` est le suivant :

```
#!/bin/sh
#
# Ceci permet au courrier electronique de sortir.
```

```
# On peut lire les news ici et recuperer le courrier .
#
/usr/local/bin/tredir 2025 25 2119 newshost:119 2110 pophost:110
#
# On peut lancer X windows ici
#
/usr/local/bin/trsh -s txconn
#
# Pour recevoir le courrier electronique...
#
/usr/local/bin/pop
#
# On vide la queue...
#
/usr/bin/runq
#
# La, c'est fini.
#
echo ^G^G > /dev/console
```

Lorsqu'enfin vous souhaitez terminer la connexion, revenez à `kermit` et quittez-le. La dernière ligne du *script* tue le `term` local et replace le système dans son état initial.

(`Note de l'auteur : plutôt que de faire '!killall -KILL term', il devrait être possible de faire un '!tshutdown'. Cela devrait également marcher, non?`)

Comme nous l'avons déjà dit, il y a des zillions de façons de procéder. Ces exemples ne sont fournis que pour vous aider à démarrer. D'autres exemples peuvent être trouvés dans les paquetages `autoterm` et `JoeltermStuff`.

10 Portage de logiciels pour une utilisation avec term

En principe, tout programme pouvant être utilisé avec un réseau peut également l'être avec `term`. Certains d'entre eux peuvent être obtenus avec un support `term` intégré. En font partie : `telnet`, `(nc)ftp`, `Mosaic` et nombre d'autres. La plupart de ces programmes ont été compilés pour les versions de `term` 1.17 et antérieures; ils devraient cependant continuer à fonctionner avec les versions de `term` plus récentes.

Une autre façon de faire fonctionner des programmes avec `term` est d'en assurer vous-même le portage. C'est le sujet dont traite cette partie.

Le dernier moyen de modifier vos programmes pour `term` est de les `termifier`.

10.1 Portage et compilation des sources

Le portage pour `term` peut être réalisé selon une procédure relativement simple :

Si le programme doit être installé dans '`/usr/local`' par *root* :

1. Ajoutez le paramètre de compilation `-include /usr/local/include/termnet.h`
2. et ajoutez `-ltermnet` à la liste de bibliothèques.

S'il doit être installé dans votre répertoire utilisateur :

1. Ajoutez le paramètre de compilation `-include $HOME/term/termnet.h`
2. et ajoutez `-L$HOME/term -ltermnet` à la liste de bibliothèques.

Compilez maintenant le logiciel comme l'indique le document '`INSTALL`' ou '`README`' fourni avec. Cela devrait suffire !

Les commandes devraient alors fonctionner à la fois avec et sans `term`.

```
telnet localhost
```

n'utilise pas `term` pour effectuer la connexion, mais :

```
telnet bohr.physics.purdue.edu
```

utilisera `term`, uniquement si vous n'avez pas d'autre type de connexion réseau.

Certaines commandes telles que `rlogin` ne peuvent être lancées que par *root* et par le propriétaire de la connexion `term` (utilisateurs privilégiés).

Certaines commandes `term` seront transparentes et n'utiliseront `term` que s'il n'y a pas d'autre possibilité. C'est en particulier le cas de `telnet` et `ftp`.

D'autres ont besoin d'un paramètre qui leur indique qu'elles peuvent utiliser `term`. C'est le cas de `xarchie`, `fsp` et `ytalk`.

Vous pouvez paramétrer ces programmes pour qu'ils utilisent `term` en positionnant la variable d'environnement `termMODE` selon les indications du fichier '`README.security`', ou bien en utilisant `make installnet`. En fin de compte, le fichier '`termnet`' créé contiendra des instructions spéciales pour le réseau. Pour l'instant, seule son existence est contrôlée.

Si vous ajoutez une connexion *éthernet* à vote système, il vous suffit d'effacer le fichier '`termnet`' et vous pourrez continuer à utiliser les mêmes exécutables !

`NOTE` : Les programmes dont le portage a été réalisé à l'époque de `client.a` peuvent être recompilés pour être utilisés avec des versions plus récentes de `term` en changeant simplement la référence à `client.a` en `libtermnet.a`.

10.2 termify

Ce paquetage permet de convertir des exécutables *linkés* de façon dynamique, en vue d'une utilisation avec term.

Avant de pouvoir utiliser termify il vous faut vous assurer que vous avez bien à votre disposition la version 2.2i de term (est-ce la version 2.2.8?) ou une version plus récente, ainsi que la version 4.5.26 ou une version plus récente de libc.so. Il vous faut alors créer le fichier 'libt.so.4' dans le répertoire '/lib' (voir le fichier 'README' du paquetage).

À l'heure actuelle, il vous faut reconstruire le fichier 'libt.so.4' à chaque fois que vous changez de version de term.

Une fois cette bibliothèque créée, vous pouvez laisser termify digérer le programme que vous voulez rendre utilisable avec term, grâce à la commande :

```
termify <nom du programme>
```

Si le résultat ne vous plaît pas, l'opération inverse est possible :

```
termify -u <nom du programme>
```

Le paquetage contient également un *script* permettant de *termifier* complètement smail, de telle façon qu'aucune définition de transport particulière ne soit nécessaire. La seule modification que vous aurez éventuellement à apporter est le champ 'From: '.

11 Clients term

11.1 Clients term disponibles sur les sites ftp

Voici une liste des applications capables de fonctionner avec term. Elle n'est pas certainement pas exhaustive; toute information complémentaire est donc la bienvenue. Autant que possible, le site et le répertoire dans lequel l'application peut être trouvée (autant que le sache l'auteur) sont fournis. Lorsque le site indiqué est sunsite.unc.edu, l'application peut être trouvée dans l'un des répertoires suivants :

1. /pub/Linux/apps/comm/term/apps
2. /pub/Linux/apps/comm/term/extra

Le paquetage term :

```
tupload
tdownload          (versions 2.1.0 et plus recentes)
trsh
tmon
tredir
```

tudpredir	(versions 2.0.0 et plus recentes)
txconn	
trdate(d)	
tshutdown	
libtermnet	

Transfert de fichiers :

ftpd	sunsite.unc.edu
termncftp	sunsite.unc.edu
ncftp185	sunsite.unc.edu:/pub/Linux/system/Network/file-transfer
fsp	sunsite.unc.edu:/pub/Linux/system/Network/file-transfer

Systèmes d'information :

lynx	
Mosaic	sunsite.unc.edu:/pub/Linux/system/Network/info-systems/Mosaic
chimera	
netscape	sunsite.unc.edu:/pub/Linux/system/Network/info-systems
httpd	
xgopher	
gopher	sunsite.unc.edu

Connexion à distance :

termtelnet	sunsite.unc.edu
rlogin	physics.purdue.edu:/pub/bcr/term/extra
rsh	physics.purdue.edu:/pub/bcr/term/extra

Netnews :

tin 1.3	sunsite.unc.edu:/pub/Linux/system/Mail/news
news2	sunsite.unc.edu

Courrier électronique :

slurp	sunsite.unc.edu
smail	sunsite.unc.edu
term.mailerd+smail	sunsite.unc.edu
BCRMailHandlerXXX	physics.purdue.edu:/pub/bcr/term

Scripts d'automatisation :

JoeltermStuff	sunsite.unc.edu
autoterm	sunsite.unc.edu
fet	sunsite.unc.edu

Autres programmes :

inetd	sunsite.unc.edu
rdate	sunsite.unc.edu
xgospel	sunsite.unc.edu:/pub/Linux/games/x11/networked
termify	physics.purdue.edu:/pub/bcr/term/extra
xboard	sunsite.unc.edu
ircII	sunsite.unc.edu:/pub/Linux/system/Network/chat
whois	
xwebster	sunsite.unc.edu
sxpc	ftp.x.org:/R5contrib
xztalk	sunsite.unc.edu:/pub/Linux/apps/sound/talk

11.2 Le paquetage `termnet`

Le paquetage `termnet-2.0.4-Linux-bin.tar.gz` (`sunsite.unc.edu:/pub/Linux/apps/comm/term`) contient un ensemble de clients `term` précompilés ainsi qu'un ensemble de *scripts*, de pages de manuel et `libtermnet` 2.00.04. Les clients ont été compilés avec cette version de `libtermnet.so`. Le paquetage contient les clients suivants :

fet	perl	sperl4.036	tmon	tshutdown	xgopher
finger	perl4.036	suidperl	trdate	tudpredir	ytalk
ftp	rcp	taintperl	trdated	tupload	
fwhois	rlogin	telnet	tredir	txconn	
ncftp	rsh	term	trsh	xarchie	

ATTENTION : Le paquetage contient l'ensemble précompilé complet des clients de `term` 2.0.4, ainsi que `term` lui-même. N'installez pas ce paquetage avant d'être sûr de ce que vous faîtes. Lorsque vous commencerez à déplacer les exécutables, vous allez détruire les autres versions de `term` et de ses clients.

11.3 Applications ne fonctionnant pas encore avec `term`

1. `DOOM` : Le problème avec ce jeu semble être le fait qu'il utilise le port 5039 à la fois en tant que client et en tant que serveur.

2. `NFS` : Le serveur `NFS` est supposé n'accepter des requêtes que si la socket demandant la connexion est attachée à un port inférieur à 1024. Ceci semble poser problème. Cepedant, certains serveurs `NFS` possèdent une option '`insecure`'. Dans ce cas, il se pourrait finalement que `NFS` fonctionne, si la gestion des `RPC` est ajoutée à `term`.

12 term et la sécurité

Dans cette partie sont évoqués certains des aspects de la sécurité de `term`. Les problèmes y sont évoqués et des moyens d'accroître la sécurité sont donnés.

12.1 trsh

`trsh` pose des problèmes de sécurité lorsqu'il est utilisé pour accéder à la machine locale depuis le système distant. Le problème posé par `term` de ses clients est qu'en plus du propriétaire de la connexion `term`, *root* peut lancer, par l'intermédiaire de la connexion des programmes modifiés pour `term`.

Ceci implique en particulier que le *root* du système distant peut lancer `trsh` et donc se faire passer pour le propriétaire de la connexion `term` assez facilement. Si cet utilisateur sur la machine locale est *root*, les risques encourus sont considérables.

La solution à ce problème est simple : il vous suffit d'ajouter la ligne suivante dans le fichier '`termrc`', sur la machine locale :

```
denyrsh on
```

Cette ligne indique que personne ne peut lancer de `trsh` sur votre machine depuis le système distant. L'accès à votre machine est cependant encore possible par l'intermédiaire de la connexion `term`, en utilisant `telnet` et les ports redirigés.

12.2 txconn et xauth

`txconn` n'est pas particulièrement sûr. N'importe qui peut se connecter à votre serveur local par l'intermédiaire de `term` et se livrer à toutes sortes d'exactions. Si ce genre de problèmes vous inquiète, il est conseillé d'utiliser `xauth` pour autoriser vos connexions. Un exemple de l'utilisation de `xauth` pour sécuriser vos connexions est donné dans la partie suivante.

12.3 sxpc, xhost et xauth

L'utilisation de `xauth` est très importante pour assurer la sécurité lorsque vous utilisez `sxpc`. Si vous n'utilisez pas `xauth` en conjonction avec `sxpc`, tous les risques d'un `xhost +` s'appliquent. Entre autres, ils sont les suivants :

- Quelqu'un peut regarder ce qui est affiché à votre écran.
- Quelqu'un peut savoir ce que vous tapez.
- Quelqu'un peut entrer des commandes dans vos fenêtres (par exemple une commande pour effacer tous vos fichiers :().

`xauth` est fourni avec `X` depuis la R4. Nous décrivons ici comment en mettre en place une utilisation de base. Attention, cette configuration est vulnérable au *snooping*.

NOTE : lorsque vous utilisez `xauth`, votre variable d'environnement `$DISPLAY` ne doit PAS être positionnée sur `localhost` (ou `localhost:quelquechose`). Si votre variable d'environnement `$DISPLAY` utilise `localhost`, les clients seront incapables de trouver les informations d'autorisation adéquates. La solution est d'utiliser le nom réel de la machine. Si vous suivez les instructions de compilation contenues dans le fichier 'README' et compilez sans l'option -DNOGETHOSTNAME, tout devrait fonctionner correctement.

Appelons C la machine sur laquelle vous allez lancer les clients et D celle sur laquelle ils seront affichés.

Tout d'abord, choisissez une *clé* composée d'au plus 16 couples de chiffres hexadécimaux (c'est-à-dire un nombre pair de caractères compris entre 0 et 9 et a et f). Dans l'exemple qui suit, il vous faudra remplacer <clé> par sa valeur.

Sur C :

```
% xauth
xauth:  creating new authority file $HOME/.Xauthority
Using authority file $HOME/.Xauthority
xauth> add Chostname:8 MIT-MAGIC-COOKIE-1 <cle>
xauth> exit
```

Sur D :

```
% xauth
xauth:  creating new authority file $HOME/.Xauthority
Using authority file $HOME/.Xauthority
xauth> add Dhostname/unix:0 MIT-MAGIC-COOKIE-1 <cle>
xauth> add Dhostname:0 MIT-MAGIC-COOKIE-1 <cle>
xauth> exit
```

Lorsque vous lancez le serveur X sur D, il faut alors lui passer le paramètre -auth $HOME/.Xauthority. Peut-être vous faudra-t-il créer ou modifier le fichier '$HOME/.xserverrc' pour contrôler la façon dont le serveur X est lancé. Par exemple :

```
#!/bin/sh
exec X -auth $HOME/.Xauthority $*
```

Assurez-vous que votre fichier '.Xauthority' n'est accessible en lecture que par vous, à la fois sur C et sur D.

13 Sources d'information

Dans cette partie sont indiquées un certain nombres d'adresses ftp et d'URL utiles auxquelles vous trouverez des logiciels et des informations sur term.

ftp :

```
sunsite.unc.edu:/pub/Linux/apps/comm/term/<toute l'arborescence>
sunsite.unc.edu:/pub/Linux/docs/HOWTO
physics.purdue.edu:/pub/bcr/term/<toute l'arborescence>
```

URL :

```
http://sunsite.unc.edu/mdw/HOWTO/term-HOWTO.html
http://zeus.cs.kun.nl:4080/term-howto/term-HOWTO.html
http://physics.purdue.edu/~bcr/homepage.html
```

netnews :

```
comp.os.linux.announce       annonces des nouvelles versions de term et du term-HOWTO
comp.os.linux.help           voici ou poser vos questions sur term
comp.os.linux.misc           un autre endroit pour les questions
comp.protocols.misc          on peut trouver ici des reponses aux questions sur term
```

Lorsque vous posez des questions dans les *news*, assurez-vous de fournir toutes les informations utiles pour y répondre (version de `term`, comment vous avez installé votre connexion, etc.). Actuellement, de nombreuses versions de `term` sont utilisées et elles ont toutes des problèmes spécifiques ou communs. Ainsi, si vous souhaitez des réponses utiles, précisez au moins la version de `term` que vous utilisez. Dans le cas contraire, certaines suppositions hasardeuses devront être faites pour vous aider à résoudre votre problème.

14 Stabilité des versions de `term`

A l'heure actuelle, de nombreuses versions de `term` ont été répandues. Bill Riemers, qui s'occupe de la maintenance de `term`, nous confie la liste suivante, qui indique quelles en sont les versions stables et celles qu'il est plus prudent d'éviter.

```
term110         --> stabilite inconnue
term111         --> stabilite inconnue
term112         --> stabilite inconnue
term113         --> stabilite inconnue
term114         --> version BETA relativement stable
term115         --> version BETA instable
term116         --> version BETA instable
term117         --> version BETA instable
term118         --> version BETA moyennement stable
term119         --> version GAMMA stable
term-2.0.X      --> versions BETA relativement stables
term-2.1.X      --> versions BETA plus stables
term-2.2.X      --> nouvelles versions BETA
term-2.3.X      -->
```

15 Performances de `term`

Grâce à Bill McCarthy, nous avons maintenant à notre disposition un tableau indiquant les performances de `term` obtenues avec différents modems et différents types de connexions. Son but est de donner aux utilisateurs débutants comme expérimentés une idée des configurations utilisées par leurs collègues et des résultats qu'ils obtiennent.

```
                           LINUX term CHART 8/14/94


|__type/vitesse modem__|_vitesse_ligne_|_cps moyen_|__maxi__|__version__|
| 1)  USR SP 14.4      |   9600        |  950      |  963   | 1.17      |
| 2)  USR SP 14.4      |  14400        | 1376      |  n/a   | 1.18p06   |
| 3)  Zoom 2400        |   2400        |  220      |  230   | 1.19      |
| 4)  Boca V.32bis 14  |  57600        | 1400      |  n/a   | 1.01/09?  |
| 5)  Viva 14.4        |  14400        | 1300      |  n/a   | 1.16      |
| 6)  USR SP 14.4      |  14400+       | 1550      | 1680   | 1.19      |
| 7)  Intel 14.4 Fax   |  14400        | 1400      | 1650   | 2.0.4     |
| 8)  cable tv hookup  |  57600        | 1500      | 1800   | 1.18p06   |
| 9)  Twincom 144/DFi  |  57600        | 1500      | 4000?  | 2.0.4     |
| 10) USR SP 14.4      |  14400        | 1200      | 1500   | 1.08      |
| 11) cable tv hookup  |  19200        | 1300      | 1800   | 1.19      |
|----------------------------------------------------------------------|

+ options/parametres du termrc :

1) default escapes   2) window 5       3) baudrate 2400    4) n/a
   baudrate 9600        timeout 200       window 3
   window 10                              noise on
   timeout 150

5) compress off      6) baudrate 19200  7) ignore 19+17    8) compress off
   window 10            compress on        window 4           escape 0, 13,
   timeout 150                             timeout 90         16-19, 255
   baudrate 38400                                             baudrate 0
                                                              shift 224
                                                              flowcrtl 500
                                                              window 10
                                                              timeout 70
                                                              retrain on
                                                              breakout 24
9) compress off     10) compress off      11) baudrate 19200
   baudrate 57600       baudrate 38400        compress on
   window 10            escape 17, 19         shift 224
   timeout 200          remote                escape 0, 13 16-17
   noise on                                   19, 255
   share on                                   window 10
   remote                                     timeout 40

Eviter des caracteres a une extremite implique qu'il faut les ignorer a l'autre.
```

16 Trucs et astuces trouvés sur le Net

Dans les *newsgroups* traitant de `Linux`, de nombreuses questions sur `term` reviennent réguliément, suivies de leurs réponses. Afin de réduire l'encombrement des *newsgroups*, certaines sont traitées ici, que l'auteur ait ou non personnellement expérimenté les solutions proposées.

- Nombreux sont ceux qui se plaignent d'ennuis avec `vi`, lequel refuse d'afficher 24 lignes dans une fenêtre qui en comporte pourtant bien 24. C'est en particulier le cas des utilisateurs d'`Ultrix`. Pour résoudre ce problème, trois méthodes sont envisageables :

  1. Se connecter au système distant par la commande :

     ```
     trsh -s telnet <hostname>
     ```

  2. Ajouter `resize; clear` dans votre fichier '`.login`'.
  3. La meilleure solution semble être d'entrer sur le système distant :

     ```
     stty 38400
     ```

- Nombreux sont ceux qui ont des problèmes avec des connexion `term` qui s'interrompent, quelles qu'en soient les raisons. C'est pourquoi ils souhaitent connaître l'état de la connexion avant de lancer des applications. Cet état peut être déterminé grâce aux petits *scripts shell* donnés en exemple ici :

  Pour ceux qui utilisent `tcsh` :

  ```
  if ( { trsh -s true } ) then
    ...
  endif
  ```

  et pour ceux qui utilisent `bash` :

  ```
  if trsh -s true; then
    ...
  fi
  ```

- `Netscape`, le nouveau client `WWW` semble poser des problèmes à ceux qui souhaitent le faire fonctionner avec `term`. Voici comment y parvenir :

  ```
  1. Termifiez netscape.
  2. Lancez termnetscape.
        Dans le menu Options | Preferences | Mail/Proxys,
        laissez vides _toutes_ les boites 'proxy'.
        Mettez 'remotehost' et 80 dans les boites SOCKS.
  3. Ne tenez pas compte de l'erreur qui survient lorsque vous
        quittez le menu Options.
  4. Si termnetscape ne fonctionne pas correctement :
        dans le menu Options | Preferences | Mail/Proxys,
  ```

```
         laissez vides _toutes_ les boites 'proxy'.
         Mettez 'none' et 80 dans les boites SOCKS.
     5. Ne tenez pas compte de l'erreur qui survient lorsque vous
         quittez le menu Options.
```

17 Divers

Compléments souhaitables à ce document :

- Détailler le traitement des problèmes.
- Détailler les considérations de sécurité.
- `termwrap`.
- Suggestions.

En tout état de cause, si vous avez des suggestions, des critiques ou quelque remarque que ce soit à propos de ce document, n'hésitez pas. Patrick Reijnen est l'auteur actuel du `term-HOWTO` et peut être joint aux adresses suivantes : `patrickr@cs.kun.nl` ou `patrickr@sci.kun.nl`.

Cette remarque s'applique également au traducteur, dont l'adresse est : `ruch@labri.u-bordeaux.fr`.

18 Remerciements

Nombreux sont ceux à qui l'auteur souhaite adresser ses remerciements. Avant tout, merci à Michael O'Reilly et à l'ensemble des développeurs de `term`, qui ont mis à notre disposition un si bel outil. L'auteur remercie également tout ceux qui ont fait part de leurs expériences et ont contribué à ce document. En font partie Bill Reynolds, le précédent auteur de ce HOWTO, Ronald Florence, Tom Payerle, Bill C. Riemers, Hugh Secker-Walker, Matt Welsh, Bill McCarthy, Sergio, Weyman Martin et bien d'autres dont il oublie de mentionner les noms.

Le traducteur tient à adresser à Xavier Cazin (`xc@itp.itp.fr`), coordinateur des traductions françaises de la documentation `Linux`, ses plus vifs remerciements pour le travail qu'il accomplit et reste à son entière disposition. Il tient également à remercier Olivier Belhomme (le premier cyber-S.D.F., sans domaine fixe :)) pour son travail de relecture, qui a permis à ce document de ne pas arriver jusqu'à vous truffé d'erreurs.