# Petit guide d'exécution à distance des applications X

Version française du *Remote X Apps mini-HOWTO*

*Vincent Zweije < zweije@xs4all.nl >*

V 0.7.5, 8 décembre 2001

Ce petit guide décrit comment exécuter des applications X à distance. C'est-à-dire, comment faire pour qu'un programme X s'affiche sur un écran d'ordinateur différent de celui sur lequel il s'exécute. Ou, autrement dit, comment faire tourner un programme X sur un ordinateur différent de celui devant lequel vous êtes assis. L'accent de ce petit guide sera mis sur les questions de sécurité. Ce petit guide contient également des informations sur la manière de faire tourner des applications X en local, mais avec un identificateur d'utilisateur (user-id) différent ainsi que des informations sur la façon de mettre en place un ordinateur comme terminal X. Adaptation française : Albert-Paul Bouillot, Frédéric Bothamy *fbothamy@mail.dotcom.fr* . Relecture de la version française : Olivier Kaloudoff *kalou@kalou.net* .

## Contents

# 1 Introduction

Ce petit guide constitue un guide sur la manière de faire tourner des applications X à distance. J'ai rédigé ce document pour plusieurs raisons :

1. Il y a eu de nombreuses questions, sur Usenet, sur la manière de faire tourner des applications X à distance ;
2. J'ai vu beaucoup, beaucoup de conseils d'utilisation de « `xhost +hostname` » ou même de « `xhost +` » pour réaliser des connexions X. **C'est d'une insécurité totale**, et il existe de bien meilleures méthodes ;
3. Je n'ai pas connaissance d'un document simple décrivant les options dont *on peut* disposer. Si vous avez des informations complémentaires, s'il vous plaît, faites-le moi savoir (en anglais) : <*zweije@xs4all.nl*> .

Ce document a été écrit en pensant à des systèmes de type Unix. Si le système d'exploitation de votre ordinateur local ou de celui qui est à distance est de type différent, vous devriez trouver ici des informations sur la manière dont les choses se passent. Cependant, il vous faudra modifier les exemples par vous-même pour les utiliser sur votre propre système.

La version (anglaise) la plus récente de ce document est toujours disponible sur le WWW à *http://www.xs4all.nl/˜zweije/xauth.html* . Il est également disponible en tant que mini-HOWTO Linux « Applications X à distance » (Remote X Apps) à : *http://www.tldp.org/HOWTO/mini/Remote-X-Apps* . Les (mini-)HOWTO du projet de documentation Linux (LDP) sont disponibles par http ou ftp sur *www.tldp.org* .

La version française la plus récente de ce document est toujours disponible sur le site du projet *traduc.org* à .

Ceci constitue la version 0.7.5. Aucune garantie, seulement de bonnes intentions. Je suis ouvert aux suggestions, idées, ajouts, pointeurs utiles, corrections (typo), et cætera. Je veux que cela reste un document simple et lisible, dans la bonne moyenne du style des guides pratiques du projet de documentation Linux. Les querelles seront redirigées vers /dev/null. 

Le contenu de ce petit guide a été mis à jour le 8 décembre 2001 par *Vincent Zweije* . La version française de ce document a été mise à jour le 4 mars 2003 par *Frédéric Bothamy* . La relecture de cette nouvelle version française a été réalisée par *Olivier Kaloudoff* .

# 2 Lectures complémentaires

Un document, en rapport avec cela, sur le WWW traite de « Que faire quand Tk dit que votre écran n'est pas sûr », *http://ce-toolkit.crd.ge.com/tkxauth/* . Il a été écrit par *Kevin Kenny* . Il suggère une solution similaire à celle de ce document pour l'authentification X (xauth). Cependant, Kevin vise plus à l'utilisation de xdm pour diriger xauth à votre place.

On m'a indiqué que le volume 8 de la série consacrée au système X Window, le « Guide de l'administrateur du système X Window » de chez *O'Reilly and Associates* était une bonne source d'informations. Cependant, ce guide n'a pas été mis à jour depuis sa publication d'origine en 1992. Il ne couvre donc que X11R4 et X11R5, tout ce qui est spécifique à X11R6 n'est pas couvert.

Il y a également un autre document qui ressemble beaucoup à celui que vous êtes en train de lire, dont le titre est « Securing X Windows », et qui est disponible à *http://ciac.llnl.gov/ciac/documents/ciac2316.html* .

Consultez également les forums de diffusion Usenet, tels que : `comp.windows.x`, `comp.os.linux.x` et `comp.os.linux.networking`.

# 3 Le contexte

Vous utilisez deux ordinateurs. Sur le premier, vous êtes dans l'environnement X Window pour taper au clavier et regarder l'écran. Sur le second, vous effectuez un important traitement graphique. Vous voulez que les sorties du second soient affichées sur l'écran du premier. Le système X Window rend cela possible.

Naturellement, vous devez disposer d'une connexion à un réseau pour pouvoir le réaliser. De préférence rapide, car le protocole X est un dévoreur de ressources réseau. Mais, avec un peu de patience et un protocole de compression de données adapté, vous pouvez même faire tourner des applications par l'intermédiaire d'un modem. Pour un protocole de compression pour X, vous pouvez aller consulter les sites : dxpc *http://www.vigor.nu/dxpc/* ou LBX

*http://www.traduc.org/docs/HOWTO/mini/lecture/LBX.html* `http://www.traduc.org/docs/HOWTO/mini/lecture/LBX.html`

(disponible en version originale sur le site de l'auteur :

*http://www.paulandlesley.org/faqs/LBX-HOWTO.html* `http://www.paulandlesley.org/faqs/LBX-HOWTO.html` ).

Vous avez deux choses à faire pour réaliser tout cela :

1. Indiquer à l'unité d'affichage locale (le serveur) qu'elle doit accepter les connexions venant de l'ordinateur à distance.
2. Dire à l'application à distance (le client) de rediriger ses sorties vers votre unité d'affichage locale.

# 4 Un peu de théorie

Le mot magique est `DISPLAY (unité d'affichage)`. Dans le système X Window, une unité d'affichage est constituée (en simplifiant) d'un clavier, d'un mulot et d'un écran. Une unité d'affichage est gérée par un programme serveur, plus connu sous le nom de serveur X. Le serveur fournit des fonctionnalités d'affichage aux autres programmes qui se connectent à lui.

Une unité d'affichage est identifiée par un nom, de type, par exemple :

- `DISPLAY=light.uni.verse:0`
- `DISPLAY=localhost:4`

- DISPLAY=:0

Un nom d'unité d'affichage est constitué d'un nom d'hôte (par exemple : `light.uni.verse` et `localhost`), du signe deux point (`:`), et d'un numéro de séquence (tels que `0` et `4`). Le nom d'hôte de l'unité d'affichage est le nom de l'ordinateur sur lequel tourne le serveur X. Si le nom de l'hôte est omis, cela signifie qu'il s'agit de l'ordinateur local. D'habitude, le numéro de séquence est 0 – cela peut changer s'il y a plusieurs unités d'affichage connectées sur le même ordinateur.

Si jamais il vous arrive de voir le nom d'une unité d'affichage avec un `.n` supplémentaire accolé à son nom, c'est qu'il s'agit d'un numéro d'écran. Une unité d'affichage peut, en théorie, avoir plusieurs écrans. Cependant, d'habitude, il n'y en a qu'un, qui porte le numéro `n=0`, et c'est le numéro par défaut.

D'autres formes de `DISPLAY` existent, mais celle-ci suffira pour notre propos.

Pour celui qui est curieux de technique :

- `hostname:D.S` signifie écran `S` sur unité d'affichage `D` de l'hôte `hostname` : le serveur X de cette unité d'affichage est à l'écoute du port TCP `6000+D`.
- `host/unix:D.S` signifie écran `S` sur unité d'affichage `D` de l'hôte `host` : le serveur X de cette unité d'affichage est à l'écoute du socket de domaine UNIX `/tmp/.X11-unix/XD` (et donc, seul `host` peut l'atteindre).
- `:D.S` est équivalent à `host/unix:D.S`, où `host` est le nom de l'hôte local.

# 5 Dire au client ...

Le programme client (par exemple, votre application graphique) sait à quelle unité d'affichage il doit se connecter en consultant la variable d'environnement `DISPLAY`. Cependant ce paramétrage peut être modifié en lançant le client avec l'argument `-display hostname:0` dans la ligne de commande. Quelques exemples peuvent clarifier les choses.

Notre ordinateur est connu du monde extérieur sous le nom light, et nous sommes dans le domaine uni.verse. Si nous fonctionnons avec un serveur X normal, l'unité d'affichage est connue comme étant `light.uni.verse:0`. Nous voulons faire tourner le programme de dessin xfig sur un ordinateur à distance, appelé `dark.matt.er`, et afficher sa sortie ici, sur light.

Supposons que vous vous soyez déjà connecté par telnet à l'ordinateur distant, `dark.matt.er`.

Si l'interpréteur de commande de l'ordinateur éloigné est csh :

```
dark% setenv DISPLAY light.uni.verse:0
dark% xfig &
```

Ou, d'une autre manière :

```
dark% xfig -display light.uni.verse:0 &
```

Si c'est sh qui tourne sur l'ordinateur à distance :

```
dark$ DISPLAY=light.uni.verse:0
dark$ export DISPLAY
dark$ xfig &
```

Ou, autrement :

```
dark$ DISPLAY=light.uni.verse:0 xfig &
```

Ou, bien sûr, également :

```
dark$ xfig -display light.uni.verse:0 &
```

Il paraît que certaines versions de telnet transmettent automatiquement la variable `DISPLAY` à l'ordinateur hôte éloigné. Si vous avez l'une de celles-ci, vous avez de la chance, et c'est effectivement automatique. Si ce n'est pas le cas, la plupart des versions de telnet *doivent* transmettre la variable d'environnement `TERM`, et avec un bidouillage judicieux, il est possible de superposer la variable `DISPLAY` sur la variable `TERM`.

L'idée, sous-jacente à cette superposition, est de réaliser une sorte de script pour effectuer ceci : avant la connexion par telnet, donnez la valeur de `DISPLAY` à `TERM`. Puis, lancez telnet. Du côté de l'ordinateur distant, dans le fichier `.*shrc` concerné, lisez la valeur de `DISPLAY` à partir de `TERM`.

# 6 Dire au serveur ...

Le serveur n'acceptera pas de connexions venant de n'importe où. Vous ne voulez pas que n'importe qui puisse afficher des fenêtres sur votre écran. Ou lire ce vous tapez – souvenez-vous que votre clavier fait partie de votre unité d'affichage !

Trop peu de gens semble réaliser que permettre l'accès à leur unité d'affichage pose des problèmes de sécurité. Quelqu'un qui dispose d'un accès à votre unité d'affichage peut lire et écrire sur vos écrans, lire vos frappes au clavier, et suivre les déplacements de votre mulot.

La plupart des serveurs disposent de deux manières d'authentifier les demandes de connexions qui arrivent : le mécanisme de la liste d'hôtes (xhost) et le mécanisme du mot de passe secret (magic cookie) (xauth). De plus, il y a ssh, l'interpréteur de commande sécurisé, qui peut acheminer les connexions X.

Veuillez noter que certains serveurs X (de XFree86) peuvent être configurés pour ne pas écouter sur le port habituel TCP avec le paramètre `-nolisten tcp`. La configuration par défaut de Debian GNU/Linux, notamment, désactive l'écoute sur le port TCP par le serveur X. Si vous désirez utiliser X à distance sur un système Debian, vous devriez réactiver ceci en modifiant la façon dont est lancé le serveur X. Veuillez voir le fichier `/etc/X11/xinit/xserverrc` pour un point de départ.

## 6.1 Xhost

Xhost permet les accès basés sur les nom d'hôtes. Le serveur entretient une liste des hôtes qui sont autorisés à se connecter à lui. Il peut aussi désactiver complètement la vérification des hôtes. Attention : cela signifie que plus aucun contrôle n'est effectué, et donc, que *n'importe quel* hôte peut se connecter !

Vous pouvez contrôler la liste des hôtes du serveur avec le programme `xhost`. Pour utiliser ce mécanisme dans l'exemple précédent, faites :

```
light$ xhost +dark.matt.er
```

Ceci permet toutes les connexions à partir de l'hôte `dark.matt.er`. Dès que votre client X a réalisé sa connexion et affiche une fenêtre, par sécurité, supprimez les permissions pour d'autres connexions avec :

```
light$ xhost -dark.matt.er
```

Vous pouvez désactiver la vérification des hôtes avec :

```
light$ xhost +
```

Ceci désactive la vérification des accès des hôtes et donc permet à *tout le monde* de se connecter. Vous ne devriez *jamais* faire cela sur un réseau où vous n'avez pas confiance dans *tous* les utilisateurs (tel internet). Vous pouvez réactiver la vérification des hôtes avec :

```
light$ xhost -
```

`xhost -` par lui-même *ne supprime pas* tous les hôtes de la liste d'accès (ce qui serait tout à fait inutile - vous ne pourriez plus vous connecter de n'importe où, pas même de votre hôte local).

*Xhost est un mécanisme vraiment très peu sûr.* Il ne fait pas de distinction entre les différents utilisateurs sur l'hôte à distance. De plus, les noms d'hôtes (en réalité des adresses) peuvent être manipulés. C'est mauvais si vous vous trouvez sur un réseau douteux (déjà, par exemple, avec un accès PPP téléphonique à Internet).

## 6.2 Xauth

Xauth autorise l'accès à tous ceux qui connaissent le bon secret. On appelle un tel secret un enregistrement d'autorisation ou cookie. Ce mécanisme d'autorisation est désigné cérémonieusement comme étant le MIT-MAGIC-COOKIE-1.

Les cookies pour les différentes unités d'affichage sont stockés ensemble dans `~/.Xauthority`. Votre fichier `~/.Xauthority` doit être inaccessible pour les utilisateurs groupe/autres. Le programme xauth gère ces cookies, d'où le surnom xauth dans ce schéma.

Vous pouvez spécifier un fichier cookie différent avec la variable d'environnement `XAUTHORITY`, mais vous aurez rarement besoin de le faire. Si vous ne savez pas quel fichier cookie votre xauth utilise, faites un `xauth -v` et il vous l'indiquera.

Au démarrage d'une session, le serveur lit un cookie dans le fichier qui est indiqué par l'argument `-auth`. Ensuite, le serveur ne permet la connexion que des clients qui connaissent le même cookie. Quand le cookie dans `~/.Xauthority` change, *le serveur ne récupérera pas la modification.*

Les serveurs les plus récents peuvent générer des cookies à la volée pour des clients qui le demandent. Les cookies sont cependant encore conservés dans le serveur : ils ne finissent pas dans `~/.Xauthority` à moins qu'un client ne les y mettent. Selon David Wiggins :

> Une possibilité supplémentaire , qui peut vous intéresser, a été ajoutée dans X11R6.3. Par l'intermédiaire de la nouvelle extension SECURITY, le serveur X lui-même peut générer et renvoyer de nouveaux cookies à la volée. De plus, on peut désigner les cookies comme étant « douteux » de sorte que les applications qui se connectent avec de tels cookies auront une capacité opératoire restreinte. Par exemple, ils ne pourront pas regarder les entrées au clavier/mulot, ou le contenu des fenêtres, d'autres clients « fiables ». Il y a une nouvelle sous-commande « generate » de xauth pour rendre cette fonctionnalité, pas forcément facile, mais au moins possible à utiliser.

Xauth possède un avantage clair, au niveau de la sécurité, sur xhost. Vous pouvez limiter l'accès à des utilisateurs spécifiques sur des ordinateurs spécifiques. Il ne permet pas l'usurpation d'adresse comme le permet xhost. Et, si vous le désirez, vous pouvez encore utiliser xhost en parallèle pour permettre des connexions.

### 6.2.1 Fabrication du cookie

Si vous voulez utiliser xauth, vous devez lancer le serveur X avec l'argument `-auth authfile`. Si vous utilisez le script **startx** pour lancer le serveur X, c'est le bon endroit pour le faire. Créez l'enregistrement d'autorisation comme indiqué ci-dessous dans votre script startx.

Extrait de `/usr/X11R6/bin/startx` :

```
mcookie|sed -e 's/^/add :0 . /'|xauth -q
xinit -- -auth "$HOME/.Xauthority"
```

Mcookie est un petit programme du paquet util-linux, site primaire *ftp://ftp.win.tue.nl/pub/linux-local/utils/util-linux* . Autrement, vous pouvez utiliser md5sum pour créer quelques données aléatoires (de, par exemple, `/dev/urandom` ou `ps -axl`) au format cookie :

```
dd if=/dev/urandom count=1|md5sum|sed -e 's/^/add :0 . /'|xauth -q
xinit -- -auth "$HOME/.Xauthority"
```

Si vous ne pouvez pas éditer le script startx (parce que vous n'êtes pas root), demandez à votre administrateur système de configurer startx correctement, ou, à la place, laissez-le configurer xdm. S'il ne peut, ou ne veut, pas, vous pouvez écrire un script `~/.xserverrc`. Si vous avez ce script, il sera exécuté par xinit au lieu du véritable serveur X. Alors, vous pourrez lancer le serveur X véritable à partir de ce script avec les arguments adaptés. Pour faire cela, faites utiliser par votre `~/.xserverrc` le `mcookie` de la ligne ci-dessus pour créer un cookie puis lancer le véritable serveur X :

```
#!/bin/sh
mcookie|sed -e 's/^/add :0 . /'|xauth -q
exec /usr/X11R6/bin/X "$@" -auth "$HOME/.Xauthority"
```

Si vous utilisez xdm pour gérer vos sessions X, vous pouvez utiliser xauth facilement. Définissez les ressources du DisplayManager.authDir dans `/etc/X11/xdm/xdm-config`. Xdm passera l'argument `-auth` au serveur X à son démarrage. Au moment de la connexion sous xdm, xdm place le cookie dans `~/.Xauthority` pour vous. Consultez xdm(1) pour de plus amples informations. Par exemple, mon `/etc/X11/xdm/xdm-config` contient la ligne suivante :

```
DisplayManager.authDir: /var/lib/xdm
```

#### 6.2.2 Transfert du cookie

Maintenant que vous avez lancé votre session X sur le serveur hôte `light.uni.verse` et que vous avez votre cookie dans `~/.Xauthority`, il vous faut transférer le cookie sur le client, `dark.matt.er`. Il y a plusieurs façons de le faire.

**Répertoires personnels (home) partagés** Le plus simple est que vos répertoires sur `light` et `dark` soient partagés. Les fichiers `~/.Xauthority` sont les mêmes, donc le cookie est transféré instantanément. Cependant, il y a un piège : lorsque vous mettez un cookie pour `:0` dans `~/.Xauthority`, dark va croire que c'est un cookie pour lui au lieu de light. Il faut que vous utilisiez un nom d'hôte explicite à la création du cookie : on ne peut pas faire autrement. Vous pouvez installer le même cookie pour, à la fois, `:0` et `light:0` avec un peu d'astuce :

```
#!/bin/sh
mcookie|sed -e 's/^/add :0 . /' -e p -e "s/:/$HOST&/"|xauth -q
exec /usr/X11R6/bin/X "$@" -auth "$HOME/.Xauthority"
```

**En utilisant le shell à distance, `rsh`** Si les répertoires *home* ne sont pas partagés, vous pouvez transférer le cookie au moyen de rsh, le shell à distance :

```
light$ xauth nlist "${HOST}:0" | rsh dark.matt.er xauth nmerge -
```

1. Extraire le cookie de votre fichier local `~/.Xauthority` (`xauth nlist :0`).
2. Le transférer vers dark.matt.er (| `rsh dark.matt.er`).
3. Le mettre dans `~/.Xauthority` là (`xauth nmerge -`).

Notez l'utilisation de `${HOST}`. Vous devez transférer le cookie qui est explicitement associé à l'hôte local. Une application X distante interpréterait une valeur d'unité d'affichage égale à `:0` comme étant une référence à la machine distante, ce qui ne correspond pas à ce que l'on veut !

**Manuellement, par Telnet** Il est possible que `rsh` ne fonctionne pas chez vous. En plus de cela, `rsh` a un inconvénient en ce qui concerne la sécurité (noms d'hôtes usurpés, si je me souviens bien). Si vous ne pouvez, ou ne voulez, pas utiliser `rsh`, vous pouvez également transférer le cookie manuellement, comme ceci :

```
light$ echo $DISPLAY
:0
light$ xauth list $DISPLAY
light/unix:0 MIT-MAGIC-COOKIE-1 076aaecfd370fd2af6bb9f5550b26926
light$ rlogin dark.matt.er
Password:
dark% setenv DISPLAY light.uni.verse:0
dark% xauth
Using authority file /home/zweije/.Xauthority
xauth> add light.uni.verse:0 . 076aaecfd370fd2af6bb9f5550b26926
xauth> exit
Writing authority file /home/zweije/.Xauthority
dark% xfig &
[15332]
dark% logout
light$
```

Consultez également rsh(1) et xauth(1x) pour de plus amples informations.

**En automatisant la méthode Telnet** Il doit être possible de superposer le cookie sur la variable `TERM` ou `DISPLAY` quand vous utilisez telnet sur l'hôte éloigné. Cela doit fonctionner de la même manière que de superposer la variable `DISPLAY` sur la variable `TERM`. Regardez la section 5 : Dire au client. De mon point de vue, sur ce sujet, vous prenez vos responsabilités, mais cela m'intéresse si quelqu'un peut me confirmer ou m'infirmer cela.

Notez, cependant, qu'avec certains Unix les variables d'environnement peuvent être visibles par les autres et vous ne pourrez pas empêcher la visualisation du cookie dans `$TERM` si certains veulent le voir.

### 6.2.3 Utilisation du cookie

Une application X, telle que `xfig` ci-dessus, sur `dark.matt.er`, ira automatiquement voir le cookie dans `~/.Xauthority` pour s'authentifier.

L'utilisation de `localhost:D` entraîne une petite difficulté. Les applications X clientes traduisent `localhost:D` en `host/unix:D` pour effectuer la recherche du cookie. Effectivement, cela signifie qu'un cookie pour `localhost:D` dans votre `~/.Xauthority` n'a *aucun* effet.

Si l'on y réfléchit, c'est logique. L'interprétation de `localhost` dépend entièrement de la machine sur laquelle s'effectue cette interprétation. Si ce n'était pas le cas, cela causerait un horrible bazar dans le cas d'un répertoire personnel (home) partagé, par exemple par l'intermédiaire de NFS, avec plusieurs hôtes interférant chacun avec ses propres cookies.

### 6.3 Ssh

Les enregistrements d'autorisation sont transmis sur le réseau sans codage. Si vous vous souciez de ce que l'on puisse espionner vos connexions, utilisez ssh, le shell sécurisé. Il effectuera des transmissions X sécurisées au moyen de connexions chiffrées.

Pour activer la transmission X par ssh, utilisez l'option de la ligne de commande `-X` ou écrivez ce qui suit dans votre fichier local de configuration de ssh :

```
Host remote.host.name
  ForwardX11 yes
```

Le serveur ssh (`sshd`) du côté distant positionnera automatiquement la variable `DISPLAY` sur l'extrémité du tunnel X transmis. Le tunnel distant récupère son propre cookie ; le serveur ssh distant le génère pour vous et le place dans `~/.Xauthority` là-bas. Ainsi, l'autorisation X avec ssh est complètement automatique.

De plus, il est génial pour d'autres choses aussi. C'est une bonne amélioration structurelle de votre système. Allez simplement voir *http://www.ssh.org/* , la page d'accueil de ssh.

Si vous possédez d'autres informations sur les méthodes d'authentification ou de chiffrement des connexions X, par exemple, grâce à Kerberos, envoyez-les moi, je les intégrerai ici.

## 7 Les applications X avec un identificateur d'utilisateur (User-id) différent

Supposez que vous vouliez faire tourner un outil graphique de configuration qui nécessite d'avoir les privilèges du compte *root* alors que la session X actuelle se déroule sous votre compte. Cela peut sembler étrange au premier abord, mais le serveur X *ne* permettra *pas* à cet outil d'accéder à votre unité d'affichage. Comment cela est-il possible alors que *root* peut normalement tout faire ? Et comment contourner ce problème ?

Élargissons le propos au cas où l'on veut faire tourner une application X, sous un identificateur d'utilisateur `clientuser`, alors que la session X a été lancée par `serveruser`. Si vous avez lu le paragraphe sur les *cookies*, il est évident que `clientuser` ne peut pas accéder à votre unité d'affichage : `~clientuser/.Xauthority` ne contient pas le cookie magique qui permet d'accéder à l'unité d'affichage. Le cookie correct se trouve dans `~serveruser/.Xauthority`.

### 7.1 Plusieurs utilisateurs sur le même hôte

Naturellement, tout ce qui marche pour un X distant marchera aussi pour un X à partir d'un identificateur d'utilisateur différent (particulièrement `slogin localhost -l clientuser`). Et ici l'hôte client et l'hôte serveur sont précisément les mêmes. Cependant, quand les deux hôtes sont les mêmes, il y a quelques raccourcis pour transférer le *cookie magique*.

On supposera que l'on utilise `su` pour passer d'un identificateur utilisateur à l'autre. Essentiellement, il faut écrire un script qui appelle `su`, mais enveloppe la commande que `su` exécute d'un peu de code qui effectue les tâches nécessaires pour le X distant. Ces tâches nécessaires sont l'initialisation de la variable `DISPLAY` et le transfert du *cookie magique*.

L'initialisation de `DISPLAY` est relativement facile ; il faut simplement définir `DISPLAY="$DISPLAY"` avant d'exécuter l'argument de la commande su. Donc, il faut simplement faire :

```
su - clientuser -c "env DISPLAY="$DISPLAY" clientprogram &"
```

Ce n'est pas tout, il faut encore transférer le cookie. On peut le retrouver en utilisant `xauth list "$DISPLAY"`. Cette commande renvoie le cookie dans un format qui convient pour l'utiliser dans la commande `xauth add` : ce dont nous avons justement besoin !

On pourrait imaginer passer le cookie par l'intermédiaire d'un canal de transmission. Manque de chance, ce n'est pas si facile de passer quelque chose à la commande `su` par l'intermédiaire d'un canal de transmission car `su` attend le mot de passe de l'entrée standard. Cependant, dans un script shell on peut jongler avec quelques descripteurs de fichiers et arriver à le faire.

Donc, on écrit un script de ce style en le paramétrant avec `clientuser` et `clientprogram`. Pendant que nous y sommes, améliorons un peu ce script, ça va le rendre un peu moins compréhensible mais un peu plus robuste. Le tout ressemble à cela :

```
#!/bin/sh

if [ $# -lt 2 ]
then echo "usage: `basename $0` clientuser command" >&2
     exit 2
fi

CLIENTUSER="$1"
shift

# FD 4 becomes stdin too
exec 4>&0

xauth list "$DISPLAY" | sed -e 's/^/add /' | {

    # FD 3 becomes xauth output
    # FD 0 becomes stdin again
    # FD 4 is closed
    exec 3>&0 0>&4 4>&-

    exec su - "$CLIENTUSER" -c \
         "xauth -q <&3
          exec env DISPLAY='$DISPLAY' "'"$SHELL"'" -c '$*' 3>&-"

}
```

Je pense que c'est portable et que cela fonctionne suffisamment correctement dans la plupart des circonstances. Le seul défaut auquel je pense en ce moment est dû à l'utilisation de `'$*'`, les guillemets simples dans `command` vont perturber les guillemets de l'argument(`'$*'`) de la commande `su`. Si cela entraîne quelque chose de vraiment gênant, envoyez-moi un courrier électronique.

Nommez le script `/usr/local/bin/xsu`, et vous pouvez faire :

```
xsu clientuser 'command &'
```

Cela ne peut pas être plus facile, à moins que vous ne vous débarrassiez du mot de passe. Oui, il existe des moyens pour y arriver (`sudo`), mais ce n'est pas l'endroit pour en parler.

Le petit script `xsu` mentionné ci-dessus a servi comme base pour un script plus étendu appelé `sux` qui a, apparemment, trouvé sa place comme paquet dans la distribution *Debian* .

### 7.2 Root est l'utilisateur client

Évidemment, tout ce qui marche pour un client non root doit fonctionner pour root. Cependant, avec root vous pouvez faire cela encore plus facilement, car celui-ci peut lire le fichier `~/.Xauthority` de tout le monde. Il n'y a pas besoin de transférer le cookie. Tout ce qu'il y a à faire consiste à initialiser `DISPLAY`, et à faire pointer `XAUTHORITY` sur `~serveruser/.Xauthority`. Donc, vous pouvez écrire :

```
su - -c "exec env DISPLAY='$DISPLAY' \
                  XAUTHORITY='${XAUTHORITY-$HOME/.Xauthority}' \
                  command"
```

Et, en mettant cela dans un script, cela donne quelque chose comme

```
#!/bin/sh
if [ $# -lt 1 ]
then echo "usage: `basename $0` command" >&2
     exit 2
fi
su - -c "exec env DISPLAY='$DISPLAY' \
                  XAUTHORITY='${XAUTHORITY-$HOME/.Xauthority}' \
                  "'"$SHELL"'" -c '$*'"
```

Nommez le script `/usr/local/bin/xroot`, et vous pouvez faire :

```
xroot 'control-panel &'
```

Cependant, si vous avez déjà initialisé `xsu` , il n'y a pas de vraie raison de faire cela.

## 8 Faire tourner un gestionnaire de fenêtres distant

Un gestionnaire de fenêtres (comme `twm`, `wmaker`, ou `fvwm95`) est une application comme n'importe quelle autre. La procédure normale devrait fonctionner.

Enfin, presque. Il ne peut tourner, au plus, qu'un seul gestionnaire de fenêtres à un instant donné dans une unité d'affichage. Si vous faites déjà tourner un gestionnaire de fenêtre local, vous ne pouvez pas lancer le gestionnaire distant (il le dira et s'arrêtera). Il faut tuer (ou simplement quitter) le gestionnaire local en premier.

Par manque de chance, beaucoup de scripts de sessions X se terminent par un

```
exec le-gestionnaire-de-fenetres-de-votre-choix
```

et cela signifie que quand le gestionnaire de fenêtre (local) se termine, votre session se termine, et le système (xdm ou xinit) considère que votre session est terminée, et effectivement, vous déconnecte.

Vous aurez encore à faire quelques contorsions, mais vous devez y arriver et ce n'est pas trop difficile. Amusez-vous un peu avec votre script de session (normalement `~/.xsession` ou `~/.xinitrc`) pour arriver à vos fins.

Attention, un gestionnaire de fenêtres permet souvent de faire tourner de nouveaux programmes qui s'exécuteront sur la machine locale. C'est-à-dire locale à la machine sur lequel tourne le gestionnaire de fenêtres. Si vous faites tourner un gestionnaire de fenêtres distant, il lancera des applications distantes, et ce n'est peut-être pas ce que vous voulez. Naturellement, elles continueront à s'afficher sur l'unité d'affichage qui est locale pour vous.

# 9 Mettre en place un terminal X

Trouvez une utilisation à votre vieux PC ! Faites-en un endroit supplémentaire pour pouvoir travailler ! Pas besoin d'acheter un nouveau matériel coûteux ! Vous avez déjà tout ce qu'il faut !

Sérieusement, vous pouvez mettre en place un vieux PC comme terminal X. Un terminal X est un ordinateur qui fondamentalement n'exécute rien d'autre qu'un serveur X. Vous pouvez vous connecter dessus et obtenir une session X, avec des xterms, xbiff, xclock et tout autre client X concevable. Cependant, les clients X s'exécuteront sur l'hôte distant et ils utiliseront le serveur X distant pour afficher leur sortie sur votre terminal X local. Même le gestionnaire de fenêtre s'exécutera à distance.

Un terminal X consomme peu de ressources en comparaison d'une machine Unix complète. J'ai ici un terminal X constitué par un processeur 486, 16 Mo de RAM et 250 Mo d'espace disque. Oh et une connexion réseau, naturellement. Il n'a même pas besoin d'avoir des répertoires utilisateurs.

Pour trouver des lectures liées à ce sujet, jetez un coup d'oeil aux documents suivants :

- Le *petit guide XDM et Terminal X* ( ). Ce document est une description complète de ce qui est possible avec XDMCP et xdm, appliqué à la construction de terminaux X. Vous devriez vraiment le lire.
- Le *guide pratique XDMCP* ( ). Ce document décrit les étapes nécessaires à la mise en place de xdm pour utiliser des serveurs X distants, comme depuis un terminal X. La configuration du serveur X dans une telle situation est décrite de façon plus succinte.
- Le *petit guide Xterminal* ( ). Il n'est pas actuellement maintenu, mais il peut contenir quelques informations intéressantes pour vous.

À la différence des documents ci-dessus, le document que vous êtes en train de lire se limite à une courte description de XDMCP, mais il insiste plus sur les problèmes de sécurité impliqués.

## 9.1 Une fois de plus, un peu de théorie en premier

En ce qui concerne X, le terminal X va n'exécuter rien d'autre qu'un serveur X. Ce serveur X sera configuré pour dialoguer avec l'hôte distant en utilisant XDMCP (le protocole de contrôle de gestion d'affichage X, « X Display Manager Control Protocol »). Il va demander à l'hôte distant une session X. L'hôte distant fournira une fenêtre de connexion dans le terminal X et après la connexion, il exécutera une session X avec tout l'habillage y compris le gestionnaire de fenêtres, tout cela utilisant X distant pour l'affichage sur le terminal X.

Vous aurez probablement remarqué que l'hôte distant agit comme un serveur, bien que pas comme un serveur X. L'hôte distant fournit les sessions X aux serveurs X qui les demandent. Ainsi, selon XDMCP, l'hôte distant est en fait un serveur, fournissant des sessions X, également connu sous le nom de serveur XDMCP. Le serveur X joue le rôle d'un client XDMCP ! Vous me suivez ?

Le programme qui fournit le service XDMCP sur le serveur XDMCP est `xdm`. Donc, pour exécuter un terminal X, vous devez configurer deux programmes : `X` (le client XDMCP) sur le terminal X et xdm (le serveur XDMCP) sur l'hôte distant.

Vous devez toujours vous rappeler que le protocole X (et le protocole XDMCP) n'est pas chiffré. Si vous devez utiliser X à distance, tout ce qui transite sur le réseau peut être espionné par d'autres hôtes du réseau. Ceci est particulièrement néfaste avec les sessions X distantes car la première chose qui se passe est la connexion en donnant l'utilisateur et le mot de passe. **Vous ne devez donc exécuter X à distance que sur un réseau sécurisé !**

## 9.2 Configurer `X` comme client XDMCP

Si vous désirez mettre en place une machine Linux comme terminal X, vous avez besoin de très peu de ressources. Fondamentalement, vous avez besoin de ce qu'il est nécessaire d'avoir pour faire fonctionner une machine Linux dépouillée plus un serveur X. Spécifiquement, vous *n'avez pas* besoin des clients et bibliothèques X. Il peut être utile d'installer certaines polices X, mais vous pouvez également utiliser un serveur de polices situé quelque part sur le réseau.

Il existe plusieurs moyens pour un serveur X d'obtenir une session X d'un serveur XDMCP. La plus simple est d'aller directement à un serveur XDMCP connu et de lui en demander une. Le serveur peut également émettre en multi-diffusion une requête et utiliser le premier serveur XDMCP qui répond. Enfin, le serveur X peut demander à un serveur XDMCP de lui fournir une liste des hôtes qui acceptent de fournir une session et laisser l'utilisateur choisir l'hôte de session.

1. Si vous connaissez l'hôte qui va vous fournir une session, utilisez-le directement. Exécutez

   ```
   X -query sessionhost
   ```

   et, en supposant que `xdm` fonctionne sur `sessionhost`, vous obtiendrez une fenêtre de connexion et, après connexion, une session X.

2. Si l'hôte duquel vous obtiendrez la session vous importe peu, utilisez la méthode d'émission de multi-diffusion. Exécutez

   ```
   X -broadcast
   ```

   et, en supposant que `xdm` fonctionne sur un hôte quelque part sur le réseau, vous obtiendrez une fenêtre de connexion du premier (et, on l'espère, du plus rapide) `xdm` qui répond et, après connexion, une session X.

3. Si vous désirez choisir l'hôte où vous voulez avoir votre session, demandez au serveur XDMCP une liste des hôtes. Exécutez

   ```
   X -indirect xdmcpserver
   ```

   et, en supposant que `xdm` est bien configuré sur ce serveur, une liste des hôtes vous sera présentée parmi lesquels vous pourrez choisir. Choisissez-en un ; vous obtiendrez la fenêtre de connexion pour cet hôte et, après connexion, la session que vous avez demandée.

Il se peut que vous ayez remarqué l'absence de l'option `-auth`. Le serveur X utilisera XDMCP pour négocier le mot de passe secret (« magic cookie ») avec le serveur XDMCP. Le serveur XDMCP placera le cookie dans votre `~/.Xauthority` après la connexion.

Après la fermeture d'une session, le serveur X va boucler, il reviendra au serveur XDMCP d'origine et lui demandera une nouvelle session (ou une liste de choix). Si vous ne désirez pas cela, vous pouvez utiliser l'option `-once`. Notez que ceci ne semble pas fonctionner avec l'option `-indirect` à cause de l'implémentation du « chooser ».

Quand vous avez déterminé la façon dont vous allez exécuter le serveur X, vous pouvez alors le placer dans un script de démarrage ou même l'exécuter directement à partir de `/etc/inittab`. Veuillez consulter la documentation de votre propre distribution pour savoir comment modifier vos scripts d'amorçage ou `/etc/inittab`.

N'exécutez *pas* un serveur X ainsi depuis le fichier de configuration `Xservers`. `xdm` s'attend à pouvoir se connecter à de tels serveurs et il pourrait les tuer s'il ne peut pas se connecter.

## 9.3 Configurer `xdm` comme serveur XDMCP

Le programme qui fournit le service XDMCP (le service de session) est généralement `xdm`. Il existe des variantes de celui-ci, comme `wdm` ou `gdm` sur Linux, mais ceux-ci fonctionnent fondamentalement de la même façon. Assurez-vous donc que `xdm` ou une variante est installé sur l'hôte où vous désirez exécuter vos sessions X. Si vous disposez d'une connexion graphique locale depuis l'hôte de session X, `xdm` est déjà installé ; la plupart des distributions Linux sont ainsi fournis de nos jours.

En plus de `xdm`, vous aurez besoin des programmes que vous désirez exécuter dans une session X. C'est-à-dire, tous les clients X comme `xterm`, `xfig`, `xclock`, des gestionnaires de fenêtre et ainsi de suite. Cependant, pour un serveur XDMCP, vous n'avez *pas* besoin d'installer de serveur X ; le serveur X fonctionnera à la place sur le terminal X.

À partir de l'histoire sur le serveur X ci-dessus, vous pouvez en conclure qu'il y a fondamentalement deux types de services XDMCP. Il y a le service *direct* qui consiste à permettre la connexion d'un client XDMCP et à lui fournir une session X. Il y a également le service *indirect* dans lequel le serveur fournit une liste d'hôtes fournissant un service direct, au choix pour le client XDMCP.

Tous les services `xdm` sont configurés dans le fichier d'accès, généralement situé à `/etc/X11/xdm/Xaccess` ou un emplacement semblable. Cet emplacement est en fait défini dans le fichier de configuration général de `xdm` `/etc/X11/xdm/xdm-config` par la ressource `accessFile`. Veuillez voir votre manuel `xdm` pour l'emplacement par défaut.

1. Si vous désirez autoriser `xdm` à fournir des sessions X aux clients XDMCP, que ce soit par multi-diffusion ou non, placez le nom d'hôte du client XDMCP (le serveur X, vous vous souvenez ?) seul sur une ligne dans le fichier `Xaccess`. En fait, vous pouvez placer un expression rationnelle correspondant à plusieurs hôtes. Voici quelques expressions valides :

   ```
   xterm023.my.domain      # xterm023.my.domain peut obtenir une session X
   *.my.domain             # tout hôte dans my.domain peut obtenir une session X
   *                       # tout hôte sur Internet peut obtenir une session X (non sécurisé)
   ```

   Vouloir fournir une session X à tout hôte sur Internet est discutable. De façon évidente, tout service que vous fournissez est une faille de sécurité potentielle dans la sécurité de votre serveur. D'un autre côté, le serveur devrait être sécurisé lui-même et un client XDMCP demandant une session X doit fournir une authentification valide avant que la session X ne soit accordée.

   De plus, la session X utilise une connexion X distante qui n'est pas chiffrée. La paire nom d'utilisateur/mot de passe de connexion sera transportée sur cette connexion. Toute personne pourrait alors espionner des combinaisons valides d'utilisateur/mot de passe tout comme sur des connexions telnet simple. Ceci est même pire que d'avoir son mot de passe secret (« magic cookie ») espionné.

Prenez vos propres décisions ici, mais je recommande de ne pas activer ce service au monde entier à moins d'avoir une bonne raison.

2. Si vous désirez fournir aux clients XDMCP (`X -indirect xdmcpserver`) une liste de choix (une liste d'hôtes pour choisir duquel ils obtiendront une session X), faite suivre l'expression rationnelle du client par le mot-clé `CHOOSER` et la liste des hôtes que le client peut choisir. À la place de la liste des hôtes, vous pouvez également spécifier `BROADCAST` ; avec ceci, `xdm` émet en multi-diffusion sur le réseau pour interroger les serveurs désirant fournir une session. Des exemples valides :

```
xterm023.my.domain      CHOOSER seshost1 seshost2
*.my.domain             CHOOSER BROADCAST
*                       CHOOSER extseshost1 extseshost2
```

   Le premier exemple permet à `xterm023` de choisir entre des sessions sur `seshost1` et sur `seshost2`. Le deuxième exemple permet à tout hôte dans `my.domain` de choisir n'importe quel hôte fournissant une session X. Le troisième exemple permet à tout hôte de choisir une session entre `extseshost1` et `extseshost2`.

   Ce n'est probablement pas une bonne idée de faire `* CHOOSER BROADCAST`. Ceci permettrait à tout hôte en dehors de votre réseau d'obtenir des informations sur les hôtes dans votre réseau. Vous ne voulez probablement pas communiquer une telle information. En fait, autoriser un choix pour tout hôte extérieur n'est probablement pas très utile de toute façon, car vous ne devriez pas autoriser des connexions arbitraires directes non plus.

Quand vous avez reconfiguré `xdm`, envoyez-lui le signal `HUP` pour le forcer à relire ses fichiers de configuration.

```
# kill -HUP pid-of-xdm
#
```

## 9.4 XDMCP techniquement

Techniquement, pour autant que je puisse le voir, XDMCP n'est pas tout à fait ce à quoi vous pouvez vous attendre d'après la description ci-dessus. `xdm` peut rediriger des serveurs X se connectant, vers un autre endroit et il utilise cette astuce pour implémenter la liste de choix. Ainsi, le choix se produit dans xdm et non dans le serveur X bien que la liste de choix soit représentée dans l'affichage du serveur X. C'est également pour cela que l'option `-once` du serveur X ne se combine pas avec `-indirect`.

# 10 Maintenance

D'ordinaire, la première fois que vous allez essayer de faire tourner une application X à distance, ça ne marchera pas. Voici quelques-uns des messages d'erreur habituels, leur cause probable et des solutions pour vous aider à progresser.

```
xterm Xt error: Can't open display:
```

Il n'y a pas de variable DISPLAY renseignée dans votre environnement et vous n'avez pas non plus lancé l'application avec le drapeau `-display`. L'application assume que la variable display contient une chaîne de caractères vide, ce qui est syntaxiquement incorrect. La solution à cela consiste à s'assurer que la variable DISPLAY est correctement renseignée dans l'environnement (avec `setenv` ou `export` selon votre shell).

```
_X11TransSocketINETConnect: Can't connect: errno = 101
xterm Xt error: Can't open display: love.dial.xs4all.nl:0
```

Erreur 101 signifie « Réseau inaccessible ». L'application n'arrive pas à se connecter au serveur à travers le réseau. Vérifiez que la variable `DISPLAY` est correctement renseignée et que la machine serveur est accessible à partir de votre client (ce qui devrait être le cas, car après tout vous êtes probablement connecté au serveur en ayant une session telnet avec votre client).

```
_X11TransSocketINETConnect: Can't connect: errno = 111
xterm Xt error: Can't open display: love.dial.xs4all.nl:0
```

Erreur 111 signifie « Connexion refusée ». La machine à laquelle vous êtes en train d'essayer de vous connecter peut être atteinte, mais le serveur indiqué n'existe pas à cet endroit. Vérifiez que vous utilisez le nom d'hôte correct et le numéro d'unité d'affichage adéquat.

Sinon, il est possible que le serveur X a été configuré pour *ne pas* écouter sur le port TCP habituel. Pour savoir s'il s'agit de ce cas, regardez si le serveur X a été lancé avec le paramètre `-nolisten tcp` et si oui, enlevez-le.

```
Xlib: connection to ":0.0" refused by server
Xlib: Client is not authorized to connect to Server
xterm Xt error: Can't open display: love.dial.xs4all.nl:0.0
```

Le client pourrait réaliser une connexion avec le serveur, mais celui-ci ne permet pas au client de l'utiliser (pas autorisé). Assurez-vous que vous avez transféré le bon cookie au client, et qu'il n'est pas périmé (le serveur utilise un nouveau cookie au démarrage d'une nouvelle session).