本稿は Linux Japan 誌 2000 年 7 月号に掲載された記事に補筆修正したものです.

X window System とネットワーク

ネットワークに繋がってない UNIX の面白みは半減すると言われます. 今では, どんな OS でも, あるいは携帯電話からもメールが出せるし, WEB サイトを見ることも可能です. それらはクライアントとして働きます. 筆者もそれらクライアントのユーザーとしてインターネット生活を楽しんでいますが, UNIX ではご存じの通りサーバー側にもなれるところが自慢ですね. 今流行りの WEB やデータベースのサーバーなどに限定せずとも, クライアント・サーバー型のネットワークアプリケーションが, UNIX 上で最初に実装されテストされるという現実があることは UNIX にとって大きな誇りでしょう. そうネットワークにおいてはまず UNIX ありきなのです. その基礎中の基礎というべき, telnet・rlogin コマンドを使って, ネットワークの楽しみがどこにあるのか考えようというのが今回のテーマの筈だったのですが, それの応用としての X のリモート接続の部分が長くなったので, 最初の基礎の方は軽く流すことにしました. 機会があったら, 重箱の隅をつついたような話もしたいとも思います (いらんってか (^^;).

telnet

リモートホスト (`remote [192.168.0.1]`) に telnet でログインする様子は

```
$ telnet remote
Trying 192.168.0.1...
Connected to remote.
Escape character is '^]'.

Vine Linux 1.1 (Rheingau)
Kernel 2.2.10 on an i586
login: matuda                    <-- ログイン名の入力
Password:     <-- パスワードはエコーバックされない
Last login: Mon Apr 24 00:01:28 on tty2
```

のようになります. こうして遠くにあるホスト上で CUI ベースの仕事が始められます. これが telnet の基本的な使い方ですが, telnet には別の使い方もあります. TCP/IP のアプリケーションはポート番号でサービスを区別します. 特に若い番号は標準のサービスに固定的に割り当てられていて (/etc/services 参照), telnet はこのポート番号で接続して対話的にサービスを利用することができます. 例えば, メールの配送プロトコル SMTP(Simple Mail Transfer Protocol) は 25 番ですから, SMTP の命令 (HELO, MAIL FROM, RCPT TO, DATA, QUIT) を使って, 次のように直接メールを送信を依頼できるのです.

```
$ telnet aya 25            <-- ポート番号 25 で接続
Trying 127.0.0.1...
Connected to aya.
Escape character is '^]'.
220 aya.film.s.dendai.ac.jp ESMTP Sendmail \
8.9.3/3.7W1.0; Mon, 24 Apr 2000 21:22:40 +0900
HELO localhos              <-- 送信側のドメイン通知
250 aya.film.s.dendai.ac.jp Hello IDENT:\
matuda@localhost[127.0.0.1], pleased to meet you
MAIL FROM:matuda@localhost         <-- 送信者通知
250 matuda@localhost... Sender ok
RCPT TO:root@localhost             <-- 受信者指定
250 root@localhost... Recipient ok
DATA                <-- メッセージ (本文) の送信開始
354 Enter mail, end with "." on a line by itself
SMTP で直接送ります。日本語大丈夫かな？
では、
.                                  <-- メッセージの終了
250 VAA14398 Message accepted for delivery
QUIT                                          <-- 終了
221 aya.film.s.dendai.ac.jp closing connection
Connection closed by foreign host.
```

POP3(110 番) や HTTP(80 番) も同じようにして telnet でデータのやりとりが体感できますから, いろいろと試してみましょう.

rlogin, rsh

Telnet ではログイン手続きが必要です. ローカルのホストでログイン手続き (正規ユーザーの確認) を済ませたつもりになっていると, 2 重の手間と感じられるでしょう. その場合, `rlogin` を用いれば, 認証が省略できます. そのために, リモートホストに次のような, 信頼できるホスト名 (必要ならユーザー名) の一覧 .rhosts を作成します.

```
192.168.0.4
aya
kirara.co.jp
```

以降, このファイルに登録されたホストからは

```
$ rlogin remote
Last login: Mon Apr 24 21:00:31 from localhost
remote:~$                          <-- ログインしている
```

パスワード認証なしでログインができるようになります. なお, このような機密ファイルのパーミッションは 600 としなければなりません. 忘れずに

```
$ chmod 600 .rhosts
```

を実行してください. rlogin が可能ならば rsh も使える可能性があります. つまり, 次のようにリモートホスト上でコマンドを実行させることができるのです.

```
$ rsh リモート コマンド
```

実行結果は，ローカルのディスプレイに表示されます．rlogin で一端リモートにログインせずにコマンドが実行できるので大変便利です．ただし，クラッカーがいつ攻撃を仕掛けてくるか判らない昨今では，インターネットに接続されたホスト上で rlogin/rsh が許可されていることは，セキュリティの観点から非常に好ましくないとされています．そういう意味では telnet も使うべきでないとされています．また root アカウントに関しては極めて重大なセキュリティホールになる可能性が高いので，rlogin/rsh はできないようにしておくのが良いとされています．root ユーザー用の . rhosts は絶対に作らないのがセキュリティの観点からの常識です．

X への接続：認証

筆者は CUI こそ UNIX と常々主張しているのですが，GUI の基本 X を体験した時に，これは凄いと感心しました．とりわけ描画を要求する X クライアントと実際の表示を受け持つ X サーバーが，異なるホストにあっても良いというネットワーク透過性を実現していることに心底驚きました．こんなにも優れた X プロトコルはいつかグラフィックの標準プロトコルとなるに違いないと思ったのですが，少しプログラムを書いてみると判りますが Image の操作が難しいし，速度の問題もあって，今のところ UNIX の世界の標準に留まっているようです．

xhost

ローカル (眼前) の X サーバーに，リモートのホストで動いている X クライアントの描画要求を受け付ける方法の一つにホスト単位の認証を行う xhost があります．この方法はとても簡単で，まずローカル側でリモートホストからの接続を

```
$ xhost +リモート
```

として許可します．この準備が整ったら，リモート先にログインして，明示的にローカルのディスプレイ番号 (通常:0) を指定して

```
$ X クライアント -display ローカル:0
```

と起動するか，環境変数 $DISPLAY を設定して

```
$ export DISPLAY=ローカル:0
$ X クライアント
```

と起動すれば良いのです．学校や職場内のネットワークに属するホストはある程度信頼して良いので，この方法で高速なホストにログインし，結果のみローカルに表示させるなどという使い方をしているユーザーも多いことでしょう．リモートからの接続許可を取り消すには

```
$ xhost -リモート
```

とします．単に xhost 打つと，現在接続許可になっているホスト名の一覧が表示されます．

xauth

xhost はホスト単位の認証ですから，接続を許されたリモートホストを使用している全てのユーザーの実行する X クライアントに，認証なしの接続が許可されます．当然悪用される可能性もあります．例えば xev のようなイベント監視ツールをこっそりと (窓なしで) 仕掛けてキー入力を盗み見る手口などが例としてあげられます．実際，“xwininfo” でルートウィンドウの id 番号を取得して

```
$ xwininfo
...
xwininfo: Window id: 0x26 (the root window)...
...
$ xev -id 0x26
```

とすれば，ルートウィンドウ上のイベントが全て表示されます．もし他のウィンドウの id を取得できれば，そのウィンドウのイベントも全てお見通しとなるのです，コワー．

そこで，もっと安全性の高い，認証データ (パスワード) を使用する接続の方法があります．xauth はいくつかある方式のそれぞれの認証データを扱うツールです．xauth 自身で認証を行って X サーバーに接続するわけではないので混乱しないでください．

auth 付きで X を起動するには，認証レコード (クッキー) が用意されていなければなりません．一般には $HOME/.Xauthority に保存されていますから，どんなサーバー用のクッキーがあるかを見てみましょう．

```
$ xauth list
aya/unix:0  MIT-MAGIC-COOKIE-1 \
            8a446ed496f0cc4f46ebb1353d998166
aya/unix:8  MIT-MAGIC-COOKIE-1 \
            ef13893cfe2de0e047ec5c4b220862ea
```

ホスト名 aya の ディスプレイ番号 :0 と :8 がありま

した．認証方式は MIT-MAGIC-COOKIE-1 です．もし，“xauth list” で何も表示されないなら，クッキーを作る必要があります．そのためのユーティリティ，その名もずばり “mcookie” を使って

```
$ xauth add :0 . ‘mcookie‘
```

とします．mcookie は**バッククォート**で囲んでください (bash では$(mcookie)$も可)．これはシェルのコマンド置換 (Command Substitution) という機能を用いるからです．すなわち，コマンドの名前に替えてコマンドの実行結果が引数として渡されます．これは UNIX の常識．こうしてディスプレイ番号 :0 用のクッキーが準備できました (xauth list で再確認してください)．そこで，オプション -auth ... を付けて

```
$ startx -- :0 -auth $HOME/.Xauthority \
                    -audit 4 &> Xserver.log
```

のように X を起動しましょう． -audit 以下はログを取るためのオプション指定ですので，煩わしいと思うなら必要ありません．

このようにして立ち上げると，xhost コマンドでは

```
$ xhost
access control enabled, \
       only authorized clients can connect
```

とだけ表示されて，初期には xhost により接続を許可されているホストが無いことが判ります．そして，接続時に同じクッキーを提示する X クライアントだけが接続を許可されるのです．

という訳で，リモートの X クライアントが，ローカルの (眼前の) X サーバーに接続するには，このクッキーが必要ですから．リモート先に，同じクッキーを設定しなければなりません．NFS でホームを共用しているような場合には，$HOME/.Xauthority も共通ですから，何もする必要がありません．ところが，外部 (自宅など) から接続する場合には，リモート側で

```
xauth add ローカル IP:0 . 同じクッキー
```

を実現する必要があります．

ローカルホストの IP アドレスは DNS によりローカルホスト名となっていても構いませんから，簡単に入力できます．問題は，クッキー (英数文字 32 個の並び) ですが，これを間違えずに入力するのは面倒です．したがって，rsh や ftp を利用してクッキーを転送する方法が，Remote-X-Apps.txt.gz に記述されています．すなわち，ローカル側で

```
$ xauth nlist :0 | rsh リモート xauth nmerge -
```

としなさいとあります．この方法はローカル側が固定 IP アドレスを持っている場合には旨くいくのですが，プロバイダー経由でその都度臨時の IP アドレスを割り振られる場合には，まず rsh が失敗します．それは当然で，.rhosts にローカルホスト名が登録されていないからです．発想を変えて，リモート先で

```
$ rsh ローカル xauth nlist :0 | xauth nmerge -
```

とするなら，ローカル側の .rhost にリモート接続する固定のホストを登録しておけばよいので転送には成功します．なお rsh で xauth を起動するには，設定によっては絶対パス指定：/usr/X11R6/bin/xauth としなければならない場合もありますから注意しましょう．これでも旨く行かない場合もありますが，それについては電話回線でのところで説明します．

これでローカルと同じクッキーを設定することができましたので，xhost と同じく，X クライアント起動時にディスプレイをローカル:0 と明示的に指定して，

```
$ X クライアント -display ローカル:0
```

とするか，環境変数 $DISPLAY をローカル:0 に設定して

```
$ export DISPLAY=ローカル:0
$ X クライアント
```

とするかで，ローカルの X サーバーにリモートの X クライアントを表示させることができます．

電話回線経由で：LBX

電話回線などの低速な通信網を使って X をリモート接続するのは，だいたいにおいて自分はいらいらするし他人には迷惑だし，あまりお薦めできません．昔なら『止めとけ，テレタイプだけで十分』というところです．が，他人の迷惑顧みず使ったもん勝ちのビデオ配信が実際に行われる昨今となっては，遠慮は却って悪貨をはびこらせる結果になるでしょうから，X のリモート接続を試すのも悪くはないでしょう．もちろんそれが良貨という訳ではないのですが (^^;

さて，低速回線を使わざるを得ない場合には転送データ量を少なくす工夫を凝らさなければなりません．X においても，当然低速回線使用を前提にして，クライアントからの要求やサーバーからの応答を圧縮して伝送するための拡張 LBX (Low BandWidth extension to X) が定まっています．なお，X サーバーが LBX に対応しているかどうかは，xdpyinfo で調べることができます．

```
$ xdpyinfo
....
number of extensions:  19
BIG-REQUESTS
DOUBLE-BUFFER
DPMS
LBX <--- あった
MIT-SCREEN-SAVER
...
```

以下に紹介する２つのツールで LBX を利用するには，クライアントは特別な指定をする必要がありません．単にツールが起てたみせかけ (fake を仮にこう訳します) サーバーに接続するだけです．もちろん，接続の際の認証に必要なクッキーをリモートホストに送る手間 (xhost を使うなら必要ありません) とツールを立ち上げる手間とが余分にかかります．

今回の試験を行うにあたって，Paul D. Smith 氏 (psmith@baynetwaorks.com) の書いた The LBX Mini-HOWTO が参考になりました．JF では伊佐治哲氏により翻訳されています (LBX.txt.gz) から是非一読ください．

dxpc [1]W[3]

LBX.txt とは逆順になりますが，まず dxpc から説明します．Dxpc ではローカルとリモート両方で起動した dxpc 間で通信を行い，X プロトコルを圧縮・伝送・伸長します．ローカル側で xhost により リモートが接続許可になっている場合を例にしますと，まずリモート側で dxpc を次のようにクライアントモードで立ち上げます．

```
$ export DISPLAY=ローカル:0
$ dxpc [-s1] -f
$ export DISPLAY=:8
```

リモート上の見せかけのサーバーは :8 をデフォルトで使うことになっています．これを変更したければ，dxpc でオプション -d n (n は番号) を指定します．その場合には当然最後の行が DISPLAY=:n となります．また，ここでのローカルとは DNS に登録された名前です．すなわち，プロバイダーから割り当てられた自宅ホストの IP アドレスあるいはそれに対応するホスト名です．そこで，リモートにログインした後に，このプロバイダー上の名前を調べます．それはログイン情報から簡単に取得できます，例えば who -m を実行すると，

```
$ who -m
matuda   ttyp0   Apr 13 09:49 (***.t3.rim.or.jp)
```

となって，最後の欄にログインを掛けてきたホスト名前が表示されます．リモートの X クライアントは，最後に設定された見せかけの X サーバー :8 に描画要求を出しますが，裏で dxpc がプロトコルを圧縮した上で本当のサーバー ローカル:0 側の窓口の dxpc に伝えるのです．ローカル側もこの要求を受けるために dxpc をサーバーモードで起動します

```
dxpc [-s1] -f リモート
```

これでリモートからの接続要求を待つデーモンが動きます．後はリモート側で X クライアントを普通に起動してください．オプション -s1 を付けて起動しておくと，切断時に圧縮率の統計を表示します．

次に，より安全とされるクッキーによる認証下での接続方法を説明します．といっても xhost とそんなに差があるわけではなく，クッキーを設定するだけのことで xauth のところで説明した手順を踏めばよいのです．その通り実行して，登録状況もついでに確かめると，

```
$ rsh ローカル xauth nlist :0 | xauth nmerge -
$ xauth list
aya/unix:0  MIT-MAGIC-COOKIE-1  ...
```

となっていて，あれれ，筆者が勝手に付けた自宅ホストの名前のままです．これでは困ります．次のように登録されて欲しいのです．

```
$ xauth list
***.t3.rim.or.jp:0  MIT-MAGIC-COOKIE-1  ...
```

あれやこれやオプションを試してみましたが xauth だけではどうにもなりません．また，mkxauth という便利なツールも役に立ちません．ここは地道に，まずはクッキーのみを転送してファイルに保存し (仮に Cookie としましょう)，ローカル名を明示的に与えて

```
xauth add ローカル:0 . ‘cat Cookie‘
```

としましょう．一連の手続きはつぎのようにまとめられます．

```
#! /bin/bash

XAUTH=/usr/X11R6/bin/xauth
FROMHOST=‘who -m |awk ’{print $6}’‘
rsh $FROMHOST $XAUTH list :0 \
              |awk ’{print $3}’ > Cookie
xauth add $FROMHOST:0 . ‘cat Cookie‘
xauth add unix:8 . ‘cat Cookie‘
exit 0
```

最後の方で，見せかけの X サーバー:8 に対してもクッキーを設定しました．実は，dxpc を用いる場合には，この値は何でもよいのですが，設定自体はされている必要があります．なぜなら，見せかけとはいえ一端は :8 に接続許可を求めるからです．

これで，やっとプロバイーダー上の名前に対して，同

じクッキーを設定できました．なお，安全確保のためにこんなに苦労してにもかかわらず，クッキーが平テキストでネットワーク上を流れてしまいますから，厳密には安全とはいえません．データ自身を ssh などで暗号化して流すべきであるとは LBX.txt にも書いてある通りです．その辺の話は，LJ の他の専門記事に譲ります．

さて，“xclock -digital” をローカルとリモートで起動して並べた例を図 1 に示します．

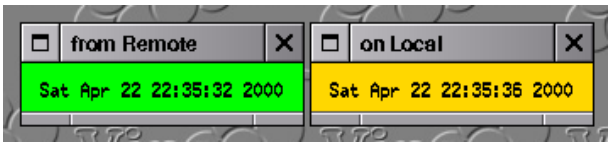

図 1　Xclock をデジタルモードで，自宅のローカルホストと研究室のリモートで立ち上げて，自宅側に表示させた例：時間が３秒ずれています．

lbxproxy [2]W[3]

lbxproxy はなぜか xhost による認証下では旨く接続できませんでした．クッキーによる認証の準備が整っているとして説明をします．Dxpc と異って，リモート側でのみ lbxproxy を立てることになります．すなわち，リモートで

```
$ lbxproxy [-compstats] -display ローカル:0 :8 &
$ export DISPLAY=:8
$ X クライアント
```

のように使います．みせかけとはいえ :8 のクッキーも設定しておかなければいけません．

-compstats は SIGHUP (KILL ではいけません) で切断した際に圧縮率の統計を表示するオプションです．研究室のリモートホスト上で時計クライアントを

```
xclock -bg gold -update 1
```

で立ち上げ，自宅ホストのサーバーに一分間描画させた場合の圧縮効率を以下に示します．

```
# 自宅ホスト上での結果
Requests:  normal = 11284, reencoded = 4896, \
           compressed = 2826
           3.99:1 overall reduction ratio
Responses: normal = 4100, reencoded = 2492, \
           compressed = 1068
           3.84:1 overall reduction ratio

# 自宅ホストから研究室のホストを利用した場合の結果
Requests:  normal = 11792, reencoded = 5108, \
           compressed = 2852
           4.13:1 overall reduction ratio
Responses: normal = 4092, reencoded = 2516, \
           compressed = 1198
           3.42:1 overall reduction ratio
```

数字を信じれば，ざっと 1/4 に減ったことになります．もちろん X クライアントによって圧縮率は随分と違います．なおここで，自宅ホスト上とあるのは，一つのホストで実験した結果です． 手順は，auth 付きで :0 を立ち上げて，kterm などを起動します．そこでの DISPLAY を見せかけの :8 と設定し，lbxproxy で：0 に転送するのです．すなわち，kterm 上で

```
$ lbxproxy  -display :0 :8 &
$ export DISPLAY=:8
$ xclock -bg gold -update 1
```

とします．実際にリモートにログインした場合と結果がほぼ同じなので，他の X クライアントについても事前に予測がたてられそうですね．

auth 付きで X を立ち上げる際に -audit 以下のオプションを指定して，Xserver.log にログを取っている場合には，

```
$ tail -f Xserver.log
```

などとして，X クライアントからの接続要求の様子を眺めてみてください．旨く接続できない場合のデバッグに利用できます．

参考文献

[1] 新しい Differential X Protocol Compressor のサイト．Zanchary Vonler さんに代わって Kevin Vigor さんが保守を続けています．W[3]
http://www.vigor.nu/dxpc/

[2] X-org の FTP アーカイブ．W[3]
ftp://ftp.x.org/pub/R6.5.1/lbxproxy/