Ubuntu Magazine Japan ウブントゥ マガジン ジャパン

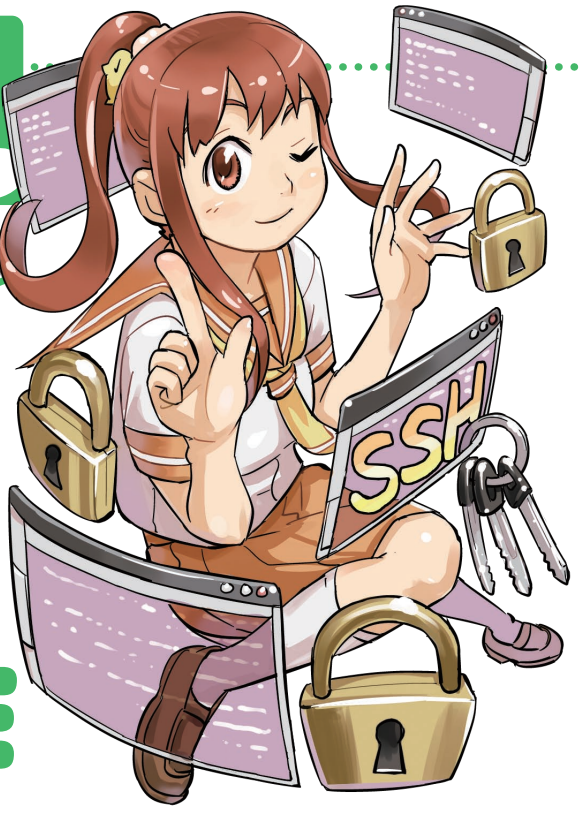
ご利用になる前に必ずお読みください

このＰＤＦファイルの内容についてのご質問・お問い合わせは株式会社アスキー・メディアワークスでは一切お受けできません。ご自身の責任においてご利用ください。

クレジット：
●文/水野 源（Ubuntu Japanese Team）
●まんが/瀬尾浩史（ブログ『架空線-AERIAL LINE-』）
●図版/古川誠之
●デザイン/シオズミタロウ
●初出/株式会社アスキー・メディアワークス「Ubuntu Magazine Japan vol.05」（http://ubuntu.asciimw.jp/）2010 年 8 月 31 日発行

not scary!! コマンドなんて怖くない not scary!!

SSH（Secure Shell）でUbuntuサーバをリモート操作



サーバをコマンドで操作しちゃおう！

Ubuntuの機能をネットワーク越しに使う！

Vol.3で端末とコマンドの基礎を、Vol.4でシェルスクリプトを紹介したCLI（コマンドライン・インタフェース）入門シリーズも今回で3回目。今回はsshコマンドを使ったUbuntuのリモート操作方法を説明していくぞ。前号までの内容と合わせれば、端末上でほとんどの作業がこなせるようになるだけでなく、Ubuntuサーバを Windows や Mac からコマンドで操作できてしまうぞ。ネットワークごしにUbuntuの機能やアプリケーションを利用できるので、クライアントPCにUbuntuをインストールしなくても、Ubuntuのパワーを利用できるぞ。Linuxの機能は使いたいけれど、デスクトップはWindowsだろjkなんてキミも安心なのだ！

ただしその前に！ 前号までの内容を読んでないワルい子は、今すぐ本屋さんにダッシュするか、太っ腹な本誌のブログ（http://ubuntu.asciimw.jp/）からバックナンバーのPDFファイルをゲットするんだ！ さぁ急いで！

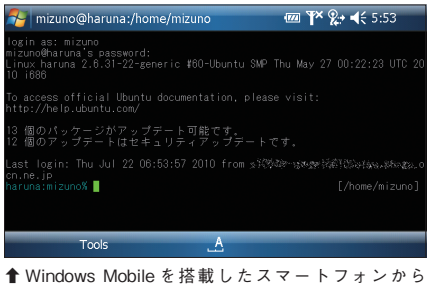
SSHとコマンドでリモート操作する仕組み

CLIがもっともよく使われるシーンのひとつが、ネットワーク越しに別マシンをリモート操作する場合だ。リモート操作といえばVNCのようなリモートデスクトップを思い浮かべる人も多いと思うが、リモートデスクトップは非常に重く、モバイル回線などで使用するには少々荷が重い。さらに言えば、Ubuntu Server をはじめとしたサーバOSにはそもそもGUI環境がインストールされていないことも多く、グラフィカルなリモートデスクトップは使えないことも多いのだ。そこで登場するのが、コマンドベースでリモート操作を行えるSSH（セキュアシェル）だ。SSHとは他のコンピュータと暗号化通信をするための仕組み（プロトコル）のひとつ。さまざまな機能があるが、とりあえずは「端末からネットワークを経由して、安全に他のコンピュータへログインできるもの」と理解しておこう。基本的にコマンドでの操作になるため、スマートフォンのようなモバイル端末からでも快適に使用できるのがうれしい特徴だ。インターネット越しにUbuntuを操作するなら、SSHでのCLI操作が一番軽快なのだ。

SSHサーバってどうインストールするの？

それではUbuntuマシンを2台用意して、SSHを試してみよう。もちろんVirtualBoxのような仮想環境のマシンを使ってもOKだ。SSHでは「操作される側」がサーバ。サーバではあらかじめSSHサーバプログラムを起動しておく必要があるので、「openssh-server」パッケージをインストールしておこう。サーバの基本的な設定は自動的に行われるため、パッケージをインストールするだけでサーバの準備は完了だ（**リスト1**）。操作する側（クライアント）はsshコマンドを使用してサーバに接続するが、sshコマンドが含まれる「openssh-client」パッケージはUbuntuに標準搭載されているので、クライアント側で特に準備は必要ない。

携帯電話からUbuntu!



⬆Windows Mobile を搭載したスマートフォンから Ubuntu に SSH 接続してリモート操作したところ。

List 01 openssh-server パッケージのインストール

```
（サーバ側で実行）
$ sudo apt-get install openssh-server
```

⬆もちろん、Ubuntu ソフトウェアセンターからもインストールできるが、この特集ではコマンドで。

●文 / 水野源（Ubuntu Japanese Team） ●マンガ / 瀬尾浩史 ●図版 / 古川誠之

SSHサーバに接続してみよう!!

それではサーバに接続してみよう。クライアントでGNOME端末を起動して、sshコマンドを実行しよう（**リスト2**）。sshコマンドに必要なパラメータは、サーバ上のユーザ名とサーバのIPアドレスだ。ユーザ名は「-l」オプションで指定するほか、IPアドレスの前に「ユーザ名@」の形で指定することもできる。ちなみにクライアントでsshコマンドを実行するユーザとサーバ上のユーザが同一の場合は、ユーザ名の指定は省略することも許されている。

SSH接続には鍵が必要になる

さて、はじめてサーバに接続する時には（**リスト3**）のようなメッセージが表示されたと思う。これはサーバのホスト公開鍵を受け入れるかどうかという問い合わせだ。SSHサーバはホスト鍵というものをもっており、クライアントは過去に接続したサーバの鍵をローカルに保存している。接続するたびにサーバが通知する鍵と自分が保存している鍵を比較して、接続先のサーバが正当なものかどうかを判断しているのだ。そして初回接続時は鍵がローカルに保存されていないため、このようなメッセージが表示されるというわけだ。ここでは当然「Yes」を入力しておこう。

余談だが、SSHは通信を暗号化するために高速な共通鍵暗号方式を採用している。これはデータの暗号化と復号化に共通の鍵を用いる暗号化方式だが、サーバとクライアントが同じ鍵を持つためには、事前に安全な方法で「鍵の配送」を行っておかなければならない。そこでSSHクライアントは、生成した共通鍵をサーバのホスト公開鍵を用いて暗号化し、サーバへ送信する。サーバは秘密鍵で暗号を復号化することで、安全に共通鍵を得ることができるわけだ。詳しくは「ハイブリッド暗号」や「公開鍵暗号」「鍵配送問題」などのキーワードで調べてみて欲しい。

ホスト鍵が変更になって接続できない場合

もしもサーバのホスト鍵が変更されたなどの理由で、サーバから通知された鍵とローカルに保存されている鍵が異なった場合は、（**リスト4**）のような警告が表示されてログインは行われない。ホスト鍵が異なるということは、全く異なるサーバに誘導されるといった攻撃を受けている可能性があるからだ。このような場合は、すみやかにサーバ管理者に問い合わせておこう。当然だが、自分が管理しているサーバを再インストールして鍵が作り直された場合などは心配する必要はない。このような場合にはクライアントに保存されている古い鍵を破棄して、新規接続からやりなおそう。過去に接続したサーバの鍵情報は「~/.ssh/known_hosts」というテキストファイルに記録されているぞ。このファイルから該当するサーバのエントリを削除するには、「ssh-keygen」コマンドの「-R」オプションを使おう（**リスト5**）。

これでサーバに接続成功!

パスワードの入力が成功したら、ログインは完了だ。プロンプトにサーバのホスト名が表示されることから分かるように、ここで動いているシェルはサーバ上で動いているシェルだ。つまりこのシェル上でコマンドを実行すれば、サーバ上で（つまり、サーバのCPUとメモリを使って）プログラムが動作するというわけ。さぁ思う存分コマンドを叩いて、Ubuntuサーバをリモート操作しよう!

List 02 sshコマンドでサーバに接続

（クライアント側で実行）

```
$ hostname
↑hostnameコマンドを実行
ubuntu-desktop
↑クライアントのホスト名はubuntu-desktop
$ ssh -l mizuno 192.168.1.2
↑sshコマンドにユーザ名とサーバのIPアドレスを指定
mizuno@ubuntu-server's password:
↑サーバのログインパスワードを入力
Linux ubuntu-server
2.6.31-22-generic #60-Ubuntu SMP
Thu May 27 00:22:23 UTC 2010 i686

To access official Ubuntu
documentation, please visit:
http://help.ubuntu.com/

12 packages can be updated.
11 updates are security updates.

Last login: Sat Jul 17 17:31:03
2010 from ubuntu-desktop.local
$
↑サーバにログインが完了して、プロンプトが表示された！
$ hostname
↑サーバ上でhostnameコマンドを実行
ubuntu-server
↑サーバのホスト名が表示された。つまりプログラムは
サーバ上で動いている
```

↑sshコマンドを使ってサーバに接続しよう。sudo同様、パスワードのエコーバックはないので注意。

List 03 初回はサーバのホスト公開鍵を受け入れる

```
$ ssh -l mizuno 192.168.1.2
The authenticity of host '[ubuntu-
server]:22 ([192.168.1.2]:22)'
can't be established.
RSA key fingerprint is 26:2c:9a:92
:ff:5f:74:f0:f5:b2:3c:1a:5e:07:e5:
d1.
Are you sure you want to continue
connecting (yes/no)?
```

↑あらかじめ管理者に正しい鍵を確認しておき、鍵が正当なことが確認できたら受け入れるようにしよう。

List 04 警告が表示される場合

```
$ ssh 192.168.1.2
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST
IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS
DOING SOMETHING NASTY!
(...略...)
```

↑この警告が表示される場合は、鍵が差し替えられたか、違うサーバに誘導されている可能性がある。

List 05 known_hostsからエントリを削除

```
$ ssh-keygen -R 192.168.1.2
↑ -RオプションにサーバのIPアドレスかホスト名を
指定しよう

$ man ssh-keygen
↑ 詳細はssh-keygenコマンドのマニュアルを参照
```

↑「ssh-keygen」コマンドは後述する鍵の作成以外にも使える。詳細はmanページを参照してほしい。

!! 「-v」でSSH接続のプロセスを確認!

sshコマンドの冗長表示モードを使うことで、デバッグメッセージを表示できるぞ。冗長表示モードではSSH接続のプロセスを表示でき、サーバのバージョンやsshコマンドが読み込んでいる設定ファイル、さらには使用している鍵ファイルや鍵のタイプなどが端末に表示されるのだ。デバッグメッセージは何かトラブルが発生した場合などにも、原因を追跡する手がかりになる。たとえばSSHサーバにうまくログインできない場合は、認証のどの段階で失敗しているのか、読み込んでいる設定ファイルや使用している鍵ファイルは正しいか、などを簡単に確認することができるのだ。デバッグメッセージを表示させるには、sshコマンドに「-v」オプションを追加するだけでいい。デバッグメッセージは三段階あり、「v」の数を増やすことでさらに詳細なメッセージを表示させることもできるぞ。

冗長表示モード

```
$ ssh -v 192.168.1.2
OpenSSH_5.3p1 Debian-3ubuntu4,
OpenSSL 0.9.8k 25 Mar 2009
↑サーバのバージョン
debug1: Reading configuration data
/home/mizuno/.ssh/config
↑最初に個人用設定ファイルを読んでいる
（略）
debug1: Connecting to haruna
[192.168.1.2] port 22.
↑サーバのポート22に接続
debug1:Connection established.
↑接続が確立
```

公開鍵認証を積極的に使おう！

SSHでログインを行う際には、サーバのログインパスワードを入力する必要があった。暗号化通信とはいえ、ネットワーク上をパスワードが流れるのは好ましくないと思うかもしれない。そこでSSHではパスワード認証のほかに、公開鍵認証という機構を備えている。詳細な説明は省くが、ユーザはあらかじめ秘密鍵と公開鍵の鍵ペアを作成しておき、公開鍵だけをサーバに登録しておく。サーバはログインを試みているユーザの公開鍵でデータを暗号化し、クライアントは秘密鍵でデータを復号化する。復号化したデータのハッシュをサーバに送信し、サーバが正しいハッシュを確認するとユーザ認証が完了するという仕組みだ。

鍵ペアを作成するには、「ssh-keygen」コマンドを使おう（**リスト6**）。鍵の種類や長さを指定して鍵を作成することができるが、基本的にはオプションなしのデフォルトのままでかまわないだろう。これで2048ビットの大きさで、RSAという暗号方式の鍵が作成されるのだ。

「ssh-keygen」コマンドを実行すると、鍵ファイルの場所を尋ねられる。ファイルはデフォルトの「~/.ssh/id_rsa」で問題ないので、そのまま［Enter］キーを押して決定しよう。次に秘密鍵のパスフレーズを入力する必要がある。秘密鍵を使う際にはパスフレーズの入力が必要で、正しいパスフレーズを入力しないと秘密鍵を使うことができないようになっている。パスフレーズのない鍵を作ることも可能だが、鍵は単なるテキストファイルなので、もしも盗まれてしまったら大変だ。鍵に鍵をかけるというのも変かもしれないが、不正利用を防ぐためにも秘密鍵には忘れずにパスフレーズを設定しておくようにしよう。パスワードが8文字程度の「ワード」であるのに対し、パスフレーズは「フレーズ」なので、スペースなどをも含んだ「句」を指定することができる。十分に長くて安全なフレーズを設定するのがいいだろう。

List 06 ユーザの鍵ペアを作成

```
$ ssh-keygen
Generating public/private rsa key
pair.
Enter file in which to save the key
(/home/mizuno/.ssh/id_rsa):
Enter passphrase (empty for no
passphrase):
Enter same passphrase again:
Your identification has been saved
in /home/mizuno/.ssh/id_rsa.
Your public key has been saved in
/home/mizuno/.ssh/id_rsa.pub.
The key fingerprint is:
b8:3f:ac:e1:31:24:06:c1:a3:4c:95:e
5:25:f9:17:52 mizuno@mizuno-laptop
The key's randomart image is:
+--[ RSA 2048]----+
|.+.+             |
| .=+ =           |
|. S.= .          |
| o = + o         |
|  E S + S        |
| . . * +         |
|    E . +        |
|   S   o         |
|       .         |
+-----------------+
```

⬆鍵の指紋を表したアスキーアートも表示される。

作成した公開鍵をサーバに登録する

次に、作成した公開鍵をサーバに登録する必要がある。公開鍵ファイルは「~/.ssh/id_rsa.pub」というテキストファイルで、この中身をサーバ上の「~/.ssh/authorized_keys」というファイルの末尾に追記しよう。「authorized_keys」は、使用する公開鍵が順番に記述されているテキストファイルだ。新しいサーバにはじめて鍵を登録する場合は「id_rsa.pub」ファイルをコピーしてリネームするだけでかまわないが、既に鍵が登録されている場合は上書きしてしまったら大変だ。こんな時は以前紹介したリダイレクトを使用して、ファイルの末尾に追記するといいだろう（**リスト7**）。しかし（**リスト7**）のように追記を行うには、何らかの方法で公開鍵ファイルをサーバ上にコピーする必要がある。このコピーと追記の作業をまとめて行ってくれる便利なコマンドが「ssh-copy-id」だ（**リスト8**）。「ssh-copy-id」は短いシェルスクリプトなので、中身を読んでみるのも面白いぞ。

鍵の登録が正常に終了すると、SSH接続時にログインパスワードに代わって秘密鍵のパスフレーズを入力するプロンプトが表示されるようになる（**リスト9**）。ここで正常なパスフレーズを入力すれば、ログインが可能になる。クライアントがGUI環境でGNOME端末を使っている場合は、プロンプトの代わりに「パスワードと暗号鍵」（Seahorse）のパスフレーズ入力ダイアログが表示されることもある。

［アプリケーション］-［アクセサリ］-［パスワードと暗号鍵］を開いてみると、作成した鍵が管理されているのがわかるはずだ。ちなみに「ssh-copy-id」と同様に、任意のサーバへの公開鍵の登録が行えたりする。登録したい鍵を右クリックして［SSHの鍵の設定］を開いたら、鍵を登録したいサーバのアドレスとユーザ名を入力して［設定］をクリックしよう。サーバのパスワードを正しく入力できれば、登録は完了だ。

List 07 公開鍵をauthorized_keysに追記

```
（サーバ側で実行）
$ cat id_rsa.pub >> ~/.ssh/
authorized_keys
```

⬆リダイレクトを使って末尾に追記しよう。「>」を使ってしまうとファイルを上書きしてしまうので注意。

List 08 公開鍵ファイルをサーバに登録

```
（クライアント上で実行）
$ ssh-copy-id -i ~/.ssh/id_rsa.pub
192.168.1.2
```

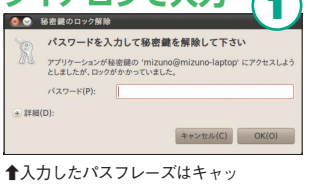
⬆-iオプションで鍵ファイルを指定し、その後にサーバのIPアドレスを指定しよう。

List 09 秘密鍵のパスフレーズの入力

```
$ ssh 192.168.1.2
Enter passphrase for key '/home/
mizuno/.ssh/id_rsa':
```

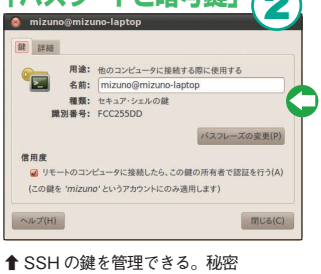
⬆SSH接続時に、秘密鍵のパスフレーズの入力を促されるようになる。パスワードではなく鍵のパスフレーズが必要。

ダイアログで入力 1



⬆入力したパスフレーズはキャッシュされるので、再度の入力は不要になるぞ。

「パスワードと暗号鍵」 2



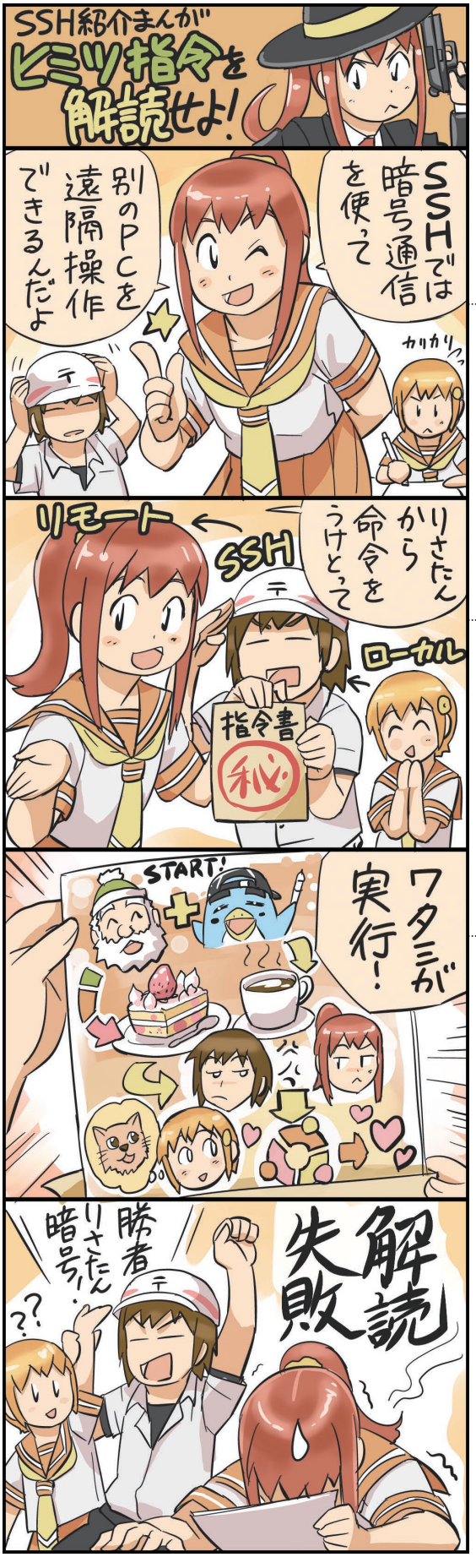
⬆SSHの鍵を管理できる。秘密鍵のパスフレーズの変更はここから行うのがわかりやすいかも。

サーバに公開鍵を登録 3

⬆公開鍵を登録したいサーバ名を入力することで、GUIからも公開鍵の登録が行えるぞ。

公開鍵を登録してもパスワードを聞かれる場合

このような場合はサーバのホームディレクトリ内にある「.ssh」ディレクトリや「authorized_keys」ファイルの権限を確認しておこう。これらの書き込み許可が第三者に与えられていると、当然だがSSHは鍵を信用しないのだ。「.ssh」ディレクトリは700、「authorized_keys」は600のパーミッションにしておこう。



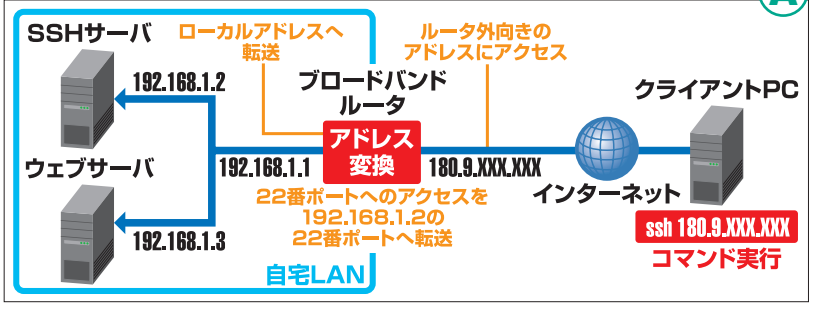
インターネット越しにおうちサーバへアクセス

外出先から自宅のSSHサーバに接続してファイルを見たり、LAN内のサービスを利用できたら便利ではないだろうか？　ここでは一般的なブロードバンドルータを使って、自宅内のサーバをインターネットから接続可能にする方法を紹介しよう（図A）。

インターネット上に存在するノードには、グローバルIPアドレスというIPアドレスが割り当てられている。最近のブロードバンド環境ではブロードバンドルータを利用してPPPoE接続をしているケースが大半だと思われるが、この場合はブロードバンドルータのWAN側ポートに、プロバイダからグローバルIPアドレスが割り当てられていることになる（グローバルなIPアドレスが割り当てられるかどうかは、プロバイダによって異なる。一部のケーブルテレビ回線などでは、プライベートなIPアドレスが割り当てられることもある）。このIPアドレスを指定すれば、インターネット上のどこからでも、自宅のルータまでたどり着けるわけだ。だが、実際に接続したいのはルータではなく、その内側のLANで動作しているサーバだ。LAN内では主に「192.168.～」や「172.16.～」などの数字ではじまるプライベートIPアドレスが使用されており、これらのアドレスを持つ機器にインターネットから直接接続することはできない。

そこで多くのブロードバンドルータにはアドレス／ポート変換機能が搭載されている。これはインターネットからルータに向けられたアクセスを、LAN内の特定のマシンへ転送する機能だ。アドレス変換を利用することで、インターネットからのSSHアクセスをLAN内のSSHサーバに転送することができる。SSHはTCPとUDPのポート22番を使用するので、この2つのポートをSSHサーバへ転送するように設定しよう。基本的には「変換元のポート」「変換先のIPアドレス」「変換先のポート（この設定はない機種もある）」を指定すればいい。例えばバッファローの無線LANルータであれば、ウェブブラウザからルータの管理画面を開くことができる。ここで「ネットワーク設定」-［アドレス変換］を選択すると、アドレスとポートの変換設定を登録するフォームが表示される。そうしたら「ポート変換テーブルの編集」をクリックして、プロトコル（TCP／UDP）とポート（22）を選択して「新規追加」をクリックすればいい。設定はルータごとに異なるので、詳しい手順は使用しているルータのマニュアルを参照してほしい。

インターネットから自宅のLAN内にアクセスするには　A



ルータのアドレス変換機能を使う

多くのブロードバンドルータでは、ウェブブラウザから管理画面を呼び出せる。「アドレス変換」というような項目が存在するので、SSH用のポートをローカルアドレスに変換してあげよう。

SSHサーバの設定をカスタマイズ

ここではSSHサーバの設定をカスタマイズする方法を紹介しよう。SSHサーバの設定ファイルは「/etc/ssh/sshd_config」だが、設定ファイルの編集を始める前にオリジナルのバックアップを取っておこう。バックアップファイルから書き込み権限を落としておけばより安全だ（**リスト10**）。設定を変更した後はSSHサービスを再起動（**リスト11**）して、設定を反映するのも忘れないように！またここでは代表的な設定の一部を紹介するが、さらなる設定については「man sshd_config」を参考にしてもらいたい。

List 10 設定ファイルのバックアップ

```
$ sudo cp /etc/ssh/sshd_config /
etc/ssh/sshd_config.factory-
defaults
$ sudo chmod a-w /etc/ssh/sshd_
config.factory-defaults
```

⬆オリジナルの設定ファイルをバックアップ、書き込み権限を剥奪する。

List 11 SSHサーバの再起動

```
$ sudo /etc/init.d/ssh restart
```

⬆設定ファイルの編集が終了したら、SSHのサービスを再起動しよう。

待ち受けるポートを変更しておこう！

SSHが使用するポート番号はデフォルトで22番に決まっている。そしてSSHはログインに成功すればシステムのシェルを取ることができるため、インターネットに向けてSSHの口を開けていると、ものすごい数の不正アクセスがやってくる。認証の記録は「/var/log/auth.log」に残るので、興味があったら見てみるといいだろう（**左ページコラム参照**）。これらの攻撃は22番ポートを決め打ちでやってくるので、ポートを通常使用されない番号に変更することで、攻撃の大部分を回避することができる。ポートを指定しているのは「sshd_config」ファイルのPort行だ。ここを任意のポート番号に変更することで、待ち受けポートを変更することができるぞ。インターネットにサーバを公開するならば、絶対にやっておきたい設定だ（**リスト12**）。なお、sshコマンドも暗黙で22番ポートへ接続へ行くため、22番ポート以外で待ち受けているサーバに接続するには「-p」オプションを使用してポートを明示する必要があることに注意しよう。

パスワード認証を禁止にしておく

パスワード認証は「でたらめなパスワードを打ち込み続け、偶然一致することを期待する」タイプの攻撃によって突破される可能性がある。先に紹介した公開鍵認証ならばこの手の攻撃では事実上突破できないため、パスワード認証は禁止してしまおう。パスワード認証を許可する設定は「sshd_config」ファイルの「PasswordAuthentication」行だ。デフォルトで有効になっているので、コメントアウトを外して「no」に変更しよう（**リスト13**）。ただしこの設定は、くれぐれも公開鍵を用いて正常にログインが可能なことを確認してから行うこと。公開鍵の登録が済んでいない状態でパスワード認証を禁止してしまうと、当然のようにSSHサーバにログインできなくなってしまう。もしもサーバが遠隔地にあったりした場合、手も足も出なくなっちゃうぞ！

特定のユーザのみのアクセスを許可する

デフォルトでSSHは、すべてのユーザのアクセスを許可するようになっている。これを特定のユーザのみに限定するには、「AllowUsers」行を追加するとよい。デフォルトで設定ファイルにこの行は存在しないため、末尾に自分で追記しておこう。「AllowUsers」には、ユーザ名を半角スペースで区切って複数列挙することができ、ここに名前の上がっているユーザに限りSSHでのログインが許可されるようになる（**リスト14**）。似たような機能に「AllowGroups」があり、名前の通りグループ単位でアクセス許可を与えることができる。

また逆に、特定のユーザやグループのみアクセスを拒否したい場合のため、「DenyUsers」や「DenyGroups」という設定もある。これらを複数設定した場合、判定はDenyUsers、AllowUsers、DenyGroups、AllowGroupsの順に行われる。

バナーの変更でメッセージを出す

SSHでログインした際に、端末に任意のメッセージ（バナー）を表示することができる。バナーは任意のテキストファイルで、デフォルトでは「/etc/issue.net」というファイルが指定されている。ただし設定は無効にされており、実際に「issue.net」の内容が表示されることはない。設定ファイルの「Banner」行のコメントアウトを外すことで、ログイン時に「/etc/issue.net」の内容を表示できるぞ（**リスト15**）。もちろん別の任意のファイルも指定できる。こういったファイルにサーバ管理者からユーザへの案内などを書いておくのもいいだろう。

クライアント側の設定を行う

SSHサーバに設定があるように、クライアントであるsshコマンドにも設定があるぞ。システム全体で共通する設定は「/etc/ssh_config」で、ユーザごとの個人的な設定は「~/.ssh/config」に記述することになっている。次からは「~/.ssh/config」に記載しておくと便利な設定を紹介しよう。

List 15 ログイン時のバナーを有効にする

```
#Banner /etc/issue.net
  ↓
Banner /etc/issue.net
```

⬆ここでは、コメントアウトを外すだけでよい。

List 14 特定ユーザのみSSHログインを許可

```
AllowUsers mizuno jkbys
```

⬆「AllowUsers」の後にアクセスを許可したいユーザ名を列挙しよう。

List 13 パスワード認証を禁止

```
#PasswordAuthentication yes
  ↓
PasswordAuthentication no
```

⬆行頭のコメントアウト（#）を外し、yesをnoに変更しよう。

List 12 SSHが待ち受けるポートを変更する

```
# Port 22
  ↓
Port 8022
```

⬆コメントアウトを外し、22から8022に変更

（クライアント側で実行）

```
$ ssh -p 8022 192.168.1.2
```

⬆192.168.1.2のポート8022へ接続

⬆/etc/ssh/sshd_configを開き、Port行にあるポート番号を変更する。

複数の鍵ペアを使い分ける

複数のSSHサーバを使用している場合、サーバごとに異なる鍵を使いたいという要求は当然あるはずだ。「ssh-keygen」コマンドを実行した際に鍵ファイル名を訊かれたことを思い出してほしい。作成時に任意のファイル名を指定したり、作成した鍵をリネームしたりすれば、鍵ペアはいくつでも作成することができる。さらにsshコマンド実行時に「-i」オプションで使用する秘密鍵ファイルを指定すれば、異なる鍵を使い分けることも簡単だ。だがそんな面倒なことをしなくても、configファイルに使用する鍵をあらかじめ列挙しておけば、sshコマンドは接続時に列挙されている鍵ファイルを自動的に使用してくれるのだ（リスト16）。

List 16 使用したい秘密鍵ファイルを指定

```
IdentityFile ~/.ssh/id_rsa
IdentityFile ~/.ssh/id_rsa.hoge
IdentityFile ~/.ssh/id_rsa.fuga
```

⬆使用したい複数の鍵ファイルを指定しておく。

サーバごとにポートを指定する

サーバのポート番号を変更するのは基本的なテクニックだが、サーバごとに使うポートを覚えておく必要があるし、接続時に毎回ポートを指定するのも面倒だ。そこでホストごとの使用ポートをあらかじめ指定しよう。まずconfigファイル内にHost（ホスト名）というセクションを作成する。これは次のHostキーワードが現れるまでの設定を、指定されたホストのみに限定する。つまりホスト固有の設定を行えるのだ。使用するポート番号はPortで指定できるので、Hostの中にPortを設定すれば、ホストごとの使用ポートを設定できるのだ（リスト17）。

List 17 ホストごとにポートを指定

```
Host hoge
Port 8022

Host fuga
Port 2022
```

⬆hogeに接続する場合にはポート8022へ、fugaに接続する場合にはポート2022へ自動的に接続する

サーバの多段接続を自動的に行う

LAN内で複数のSSHサーバがあり、これらのそれぞれに外部から接続したいが、外部から直接接続できるのはルータにアドレス変換が設定されている1台だけ、というのは家庭や企業でもよくあるケースだと思う。このような場合はまずゲートウェイになっているサーバにSSHでログインし、そこからさらにsshコマンドでLAN内のサーバにログインする必要があるが、こんな二度手間も、設定次第で自動化することが可能だ！

まず前段で説明したHostを使ってLAN内にあるホストの設定を作成し、そこへ「ProxyCommand」を設定しよう。「ProxyCommand」というのは「サーバに接続するために使用するコマンドを定義する」設定だ。（リスト18）は、「www.local」というインターネットから直接アクセスできないサーバへアクセスする際に、まずゲートウェイになっているSSHサーバ「gw.example.org」へsshコマンドで接続し、「gw.example.org」上でncコマンドを起動して「www.local」へ接続する、という意味だ。なお「%h」はsshコマンドに与えられたホスト名、「%p」はポート名にそれぞれ置き換えられる。またHostに指定するホスト名にはワイルドカードが使用できるので、家庭内のローカルドメインに属するホスト全てに対する設定を一括で行うことも可能だ。

List 18 多段接続を自動化

```
Host www.local
⬆ www.local は LAN 内にあり、インターネットから直接接続できないホスト
ProxyCommand ssh gw.example.org nc %h %p
⬆ ssh コマンドで gw.example.org に接続し、そこで nc コマンドを起動する
```

⬆ProxyCommandで多段接続を自動化する。

ポートを変えて不正な攻撃を避ける

インターネットなどTCP/IPで行われる通信では、IPアドレスで通信相手を特定している。そしてその通信相手のサーバ上で動作している複数のサービスのうち、接続するサービスを特定するために使われるのがポート番号だ。例えばウェブサーバは一般的に80番のポートを使用して通信を待ち受けており、80番ポートに接続しに行くことは、ウェブサーバのサービスに接続することを意味するのだ。ポートは0番から65535番までが存在し、0から1023番までは「ウェルノンポート」と呼ばれている。ウェルノンポートは一般的に利用されるポートであり、勝手に使用しないほうがよいだろう。また1024から49151は登録済みポート番号と呼ばれ、自由に使用できるポートは49152から65535までとされている。ただし他のサービスとバッティングしないウェルノンポート以外の番号ならば自由に使ってもそれほど問題はないだろう。「/etc/services」ファイルにサービス名とポート番号の対応表が用意されているので、ここに登録されていないポートの中から選ぶようにするといい。

ちなみに下はあるサーバの「auth.log」の一部を抜粋したもの。同一のIPアドレスから、ftpuser、testuserなど、システムに存在しそうなユーザ名で繰り返し侵入を試みているのがわかる。

auth.logに攻撃の痕跡が!

```
Jul 24 03:43:32 maya sshd[31112]: reverse mapping checking getaddrinfo for 212-95-xxx-yyy.local [212.95.xxx.yyy] failed - POSSIBLE BREAK-IN ATTEMPT!
Jul 24 03:43:32 maya sshd[31112]: Invalid user oracle from 212.95.xxx.yyy
Jul 24 03:43:40 maya sshd[31114]: reverse mapping checking getaddrinfo for 212-95-xxx-yyy.local [212.95.xxx.yyy] failed - POSSIBLE BREAK-IN ATTEMPT!
Jul 24 03:43:40 maya sshd[31114]: Invalid user mysql from 212.95.xxx.yyy
Jul 24 03:43:48 maya sshd[31116]: reverse mapping checking getaddrinfo for 212-95-xxx-yyy.local [212.95.xxx.yyy] failed - POSSIBLE BREAK-IN ATTEMPT!
Jul 24 03:43:48 maya sshd[31116]: Invalid user test from 212.95.xxx.yyy
Jul 24 03:43:56 maya sshd[31118]: reverse mapping checking getaddrinfo for 212-95-xxx-yyy.local [212.95.xxx.yyy] failed - POSSIBLE BREAK-IN ATTEMPT!
Jul 24 03:43:56 maya sshd[31118]: Invalid user testuser from 212.95.xxx.yyy
Jul 24 03:44:04 maya sshd[31120]: reverse mapping checking getaddrinfo for 212-95-xxx-yyy.local [212.95.xxx.yyy] failed - POSSIBLE BREAK-IN ATTEMPT!
Jul 24 03:44:04 maya sshd[31120]: Invalid user postgres from 212.95.xxx.yyy
Jul 24 03:44:12 maya sshd[31122]: reverse mapping checking getaddrinfo for 212-95-xxx-yyy.local [212.95.xxx.yyy] failed - POSSIBLE BREAK-IN ATTEMPT!
Jul 24 03:44:12 maya sshd[31122]: Invalid user ftpuser from 212.95.xxx.yyy
Jul 24 03:44:20 maya sshd[31124]: reverse mapping checking getaddrinfo for 212-95-xxx-yyy.local [212.95.xxx.yyy] failed - POSSIBLE BREAK-IN ATTEMPT!
Jul 24 03:44:20 maya sshd[31124]: Invalid user majordomo from 212.95.xxx.yyy
Jul 24 03:44:28 maya sshd[31126]: reverse mapping checking getaddrinfo for 212-95-xxx-yyy.local [212.95.xxx.yyy] failed - POSSIBLE BREAK-IN ATTEMPT!
Jul 24 03:44:28 maya sshd[31126]: Invalid user nagios from 212.95.xxx.yyy
Jul 24 03:44:36 maya sshd[31128]: reverse mapping checking getaddrinfo for 212-95-xxx-yyy.local [212.95.xxx.yyy] failed - POSSIBLE BREAK-IN ATTEMPT!
Jul 24 03:44:36 maya sshd[31128]: Invalid user ftp from 212.95.xxx.yyy
Jul 24 03:44:44 maya sshd[31134]: reverse mapping checking getaddrinfo for 212-95-xxx-yyy.local [212.95.xxx.yyy] failed - POSSIBLE BREAK-IN ATTEMPT!
Jul 24 03:44:44 maya sshd[31134]: User libuuid from 212.95.xxx.yyy not allowed because not listed in AllowUsers
```

⬆SSHの口をデフォルトの22番のままにしておくと、攻撃されやすくなるのがわかってもらえるだろう。

SSHでGUIアプリを動かしてみよう！

SSHはコマンドだけでサーバを操作するもの。そんなふうに考えていた時期が俺にもありました。というのは冗談としても、SSHにはX転送という機能があり、遠隔地のマシン上で起動したGUIアプリを、SSHを通して手元のPCに描画させることができる。

Ubuntuをはじめとするlinuxでは、ウィンドウシステムにXウィンドウシステムが採用されている。Xはクライアント-サーバモデルを採用しており、「Xクライアント」プログラムを「Xサーバ」が実際の画面に描画する。XクライアントというのはいわゆるGUIアプリケーションのことだと思っておいてほしい。Xクライアントと Xサーバは Xプロトコルで通信をしており、普通のUbuntuデスクトップでも1台のマシンの中で、クライアントとサーバが通信を行っているのだ。この通信はSSHを通して別のマシンへ転送できる。つまりSSHサーバ上でXクライアントが起動し、SSHクライアント上のXサーバがXクライアントのウィンドウを描画するというわけだ。遠隔地でクライアントが起動し手元でサーバが動くというあたりが、一般的なクライアント-サーバのイメージとは逆なのが特徴だ。

X転送を使用するにはサーバとクライアントの両方でX転送を許可する設定を行わなければならないが、Ubuntu同士であればデフォルトで許可されているため、特別な設定は必要ない。SSHでログインしたら、Firefoxのようなアプリケーションをコマンドから起動するだけでよい。サーバやクライアントのX転送を許可する設定部分を（**リスト19**）に示したので、Ubuntu以外の環境を利用する場合は参考にしてほしい。

SSHを応用してリモート操作自由自在

SSHで遠隔地のサーバに接続していると、当然出てくる要求のひとつが「ファイルをコピーしたい」だろう。「openssh-client」パッケージには「scp」や「sftp」というコマンドが同梱されており、これらを使えばSSH通信を経由してファイルのコピーをすることができるぞ。

scpは名前からわかるように、SSHな「cp」コマンドだ。使い方も「cp」コマンドと同じで、コピー元とコピー先のファイルを指定するだけという簡単さだ。コピー元やコピー先にSSHサーバのファイルを指定するわけだが、それには通常のファイル指定の前に「サーバ名:」をつけるだけでいい。「サーバ名::」の後には「/（ルート）」を起点とするフルパスか、サーバのホームディレクトリを起点とした相対パスで指定ができるぞ。「:」の後を省略した場合はホームディレクトリが指定されることになる（**リスト20**）。

ポート転送を使ってみよう！

81ページの図Aのネットワークでは、ウェブサーバである「192.168.1.3」にインターネットからアクセスすることはできない。もちろんルータのアドレス変換テーブルにウェブサーバの80番ポートを追加すればインターネットからアクセスすることは可能になる。しかし「自分だけはアクセスしたいけれど、インターネット向けに公開はしたくない」という場合も多いだろう。このようにLANの内部で動作しており、インターネットに公開されていないウェブサーバに外部からアクセスするにはどうしたらよいだろう？「SSHサーバにログインして、サーバ上でFirefoxを起動してX転送」というのももちろん一つの答えだ。だがSSHのポート転送機能を用いれば、簡易的なVPNのようにLAN内のサービスに外部からアクセスすることもできるぞ。ここではSSHサーバである「192.168.1.2」を利用して、通信の転送を行ってもらおう。

SSHコマンドの「-L」オプションは、「ローカルで待ち受けたポートを、リモートのポートへ転送する」ことを指定するオプションだ（**リスト21**）。このコマンドを実行すると、クライアントマシン上でSSHコマンドがポート8080の待ち受けを開始する。そしてこのポートへの接続は、SSHサーバを通して「192.168.1.3」のポート80へ転送される（**図B**）。つまりクライアント上で「http://localhost:8080」へアクセスすると、この通信はSSHを経由して「192.168.1.3:80」に送られるのだ。このようにSSH通信さえ許されているのならば、あらゆるポートをSSHを経由して転送できる。このような通信の仕方を「トンネリング」などと呼び、一時的にファイアウォールの内側へアクセスしたい時などに大いに活躍するテクニックだ。

なお「-L」オプションを使用しただけでは、SSHサーバにログインしてシェルが立ち上がってしまう。単純にバックグラウンドでポート転送を行いたいだけならば、「-f -N」オプションを併用すると便利だ。

また、GUIからバックグラウンドでSSHトンネルを作成できる『gstm』というツールもある。「apt-get」で追加できるので、興味があれば試してみよう。

List 19 X転送を許可する設定

```
（SSHサーバの /etc/ssh/sshd_config でX転送を許可する）
X11Forwarding yes

（sshコマンド実行時にX転送を許可する）
$ ssh -X 192.168.1.2

（SSHクライアントの ~/.ssh/config でX転送を許可する）
ForwardX11 yes
```

⬆ configファイルでX転送を許可しておけば、-Xオプションをいちいち指定する必要はないぞ。

List 20 scpコマンドでファイルコピー

```
$ scp example.txt 192.168.1.2:
⬆ example.txt を 192.168.1.2 のホームディレクトリにコピーする
$ scp 192.168.1.2:/etc/ssh/ssh_config .
⬆ 192.168.1.2 の /etc/ssh/ssh_config をローカルのカレントディレクトリにコピーする
```

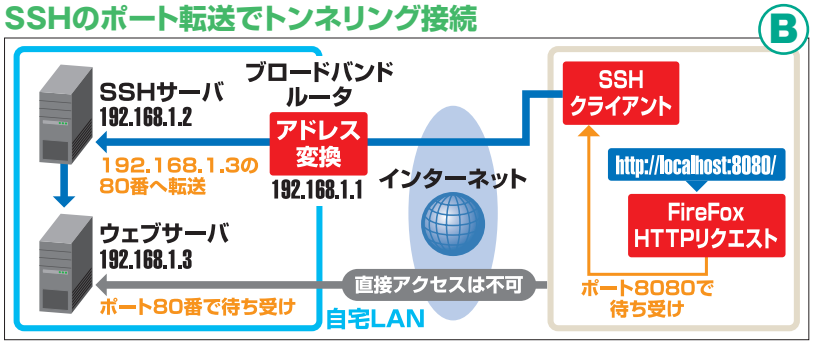
⬆ scpコマンドは、ホスト名とコロンにだけ注意すればcpコマンドと同じように使える。

List 21 SSHを用いたポート転送例

```
$ ssh -L 8080:192.168.1.3:80 180.9.xxx.xxx
$ ssh -f -N -L 8080:192.168.1.3:80 180.9.xxx.xxx
⬆バックグラウンドでポート転送のみを行う
```

⬆ポート転送を使いこなせば、sshだけでファイアウォール内側のあらゆるサービスにアクセス可能！

SSHのポート転送でトンネリング接続



List 22 iBus を起動しよう

```
$ export XMODIFIERS=@im=ibus
$ export GTK_IM_MODULE=ibus
$ ibus-daemon -d -x
$ gedit
```

⬆ログインしたら環境変数を設定して iBus を起動しよう。

Windowsから SSH接続！

SSHはセキュアな通信をするための規約であり、特定のアプリケーションを指すものではない。なのでSSHに対応したクライアントさえあれば、クライアントのプラットフォームを選ばずにUbuntuに接続することが可能だ。そして当然だがWindowsにもSSHクライアントは存在するぞ。Ubuntuマシンを別に1台用意する必要があるが、普段はデスクトップとしてWindowsを使い、Ubuntuの機能はSSH経由で呼び出すという使い方は、デュアルブートよりもずっと便利であると断言してしまおう。さらにWindows用のXサーバを導入すれば、X転送を使ってWindowsデスクトップ上でUbuntuのGUIアプリを使うこともできてしまうのだ。WindowsへのXサーバとSSHクライアントのインストールは、本誌Vol.2の「ネットウォーカーと自宅PCラクラク連携術」(http://ubuntu.asciimw.jp/elem/000/000/010/10113/ubumag-vol02-P52-57.pdf) を参考にしてほしい。この記事はWindows XPを対象に説明しているが、Windows 7でも同様の手順でインストールが可能だぞ。

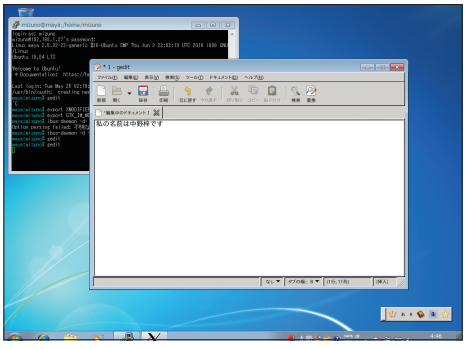
WindowsからUbuntuへログインしたら、環境変数にiBusを使用する設定を行おう。iBusのデーモンを立ち上げてからgeditやFirefoxを起動すれば(リスト22)、iBusを使用した日本語入力も可能だ。

WinSCPでファイル共有

Ubuntuで［場所］-［サーバへ接続］を実行すると、選択できるサービスの種類の中に［SSH］がある。ここでサーバ名、ポート番号、ユーザ名を入力してSSHサーバへ接続すれば、サーバのフォルダをマウントして、ファイルブラウザから操作することができる。前述の通りSSHはプラットフォームを問わないので、実はWindowsからも同様に「SSHごしにファイル操作」を行うことが可能なのだ。Windows用のクライアントは色々あるが、中でも有名なのが『WinSCP』。これはエクスプローラ風のGUIからSSHを経由してファイルをコピーできるアプリで、つまるところSambaのようなWindows共有サービスを別途インストールせずとも、SSHだけでWindows-Ubuntu間でのファイル共有が実現できるわけなのだ。

まず公式サイト (http://winscp.net/eng/docs/lang:jp) からインストーラをダウンロードして、WindowsへインストールをWinSCPの起動はデスクトップのアイコンから行える。設定項目は多岐にわたるが、サーバのアドレスとポート、ユーザ名とパスワードさえ設定すればとりあえず接続はできる。接続が正常に完了したら、ウィンドウ内にサーバのディレクトリツリーとファイルの一覧が表示されるだろう。

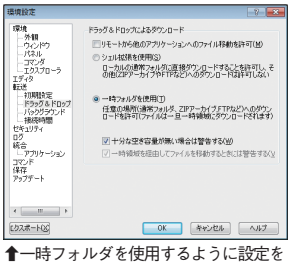
サーバへファイルをアップロードするには、エクスプローラからここへファイルをドラッグ&ドロップしてあげよう。同様にWinSCPからエクスプローラへファイルをドラッグすれば、サーバからのダウンロードができるという寸法だ。ただしドロップしたフォルダによっては、ファイルのダウンロードができないことがある。そんな場合は［表示］-［環境設定］-［ドラッグ&ドロップ］を開き、［一時フォルダを使用］を選択しよう。これでどのフォルダにもファイルをドロップできるようになるはずだ。



⬅Windows 7で「Xming + Putty」を使い、geditを起動してみた。iBusを使った日本語入力も利用できるぞ。

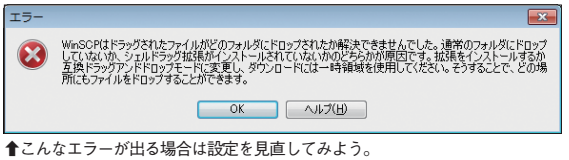
日本語入力も可能に!

一時フォルダを使用する



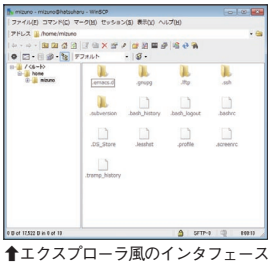
⬆一時フォルダを使用するように設定を変更しよう。これでどのフォルダにもダウンロードが可能になる。

ダウンロード時にエラー



⬆こんなエラーが出る場合は設定を見直してみよう。

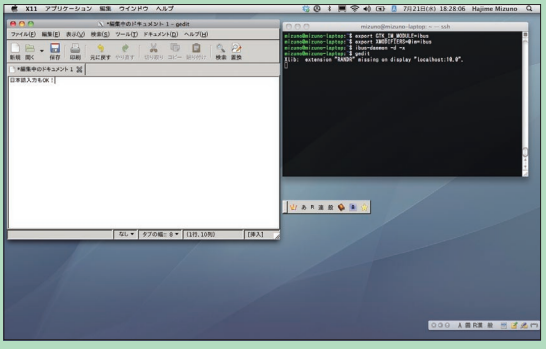
GUIでファイル操作



⬆エクスプローラ風のインタフェースの他、Norton Commander風の二画面モードも。

Macからも SSH接続だ！

Mac OS Xには、Xとターミナル、それに「ssh」コマンドが最初から用意されている。つまりMacは特別な準備を何もせずとも、UbuntuマシンにSSHログインし、さらにはGUIアプリも使用することができるのだ。「Finder」から［アプリケーション］-［ユーティリティ］-［ターミナル］を開き、「ssh」コマンドでUbuntuにログインしよう。Mac OS Xはデフォルトで X転送が禁止なので、X転送を使うならconfigファイルへ追記か［-X］オプションが必要だ。



⬅Macなら購入時の状態からXサーバとターミナルが用意されている。Ubuntuアプリを呼び出すのも簡単だ。

Macなら インストール不要!!