Focus

サイボウズのモバイルソリューション
外出先から利用できるグループウェア
「サイボウズ ポケット 4」

使いやすいWebグループウェアとして評判の高いサイボウズOffice 4に、PDAからPHSなどのデータ通信カードを使って接続できる「サイボウズ ポケット 4」が発売された。外出先からでもオフィスでPCの前にいるかのようにグループウェアを利用できる。PDAの限られた小さな画面でも快適に使え、データ通信量を節約するための工夫も随所になされている。

サイボウズ ポケット4の概要

●サイボウズ ポケット 4とは

サイボウズ ポケット 4は、Webグループウェアとして定評のあるサイボウズOffice 4に追加して利用するオプションモジュールである。単体で利用するものではなく、サイボウズOffice 4がすでに運用されている環境を前提に、サイボウズOffice 4に用意された各種アプリケーション／データにPDAでアクセスできるようにするためのものだ。

サイボウズOffice 4の機能をPDAによってフル活用できるようにすることで、サイボウズOffice 4の優れた機能に携帯性という性能を付加したと考えてよいだろう。これによって会社にいなければ得られない情報を、出張先などからいち早く取得し、アクションをとるといったビジネスのスピード化に対応するための製品である。

もちろん、PCのWebブラウザを前提としたユーザーインターフェイスと、PDAで利用可能なインターフェイスとは大きく異なる。フルカラーの高精細大画面が利用でき、マウスやフルキーボードを備えたPCとモノクロ液晶もまだ多く利用され、ごく小さな画面と数個のボタン程度しか利用できないPDAとで同じソフトウェアを利用するのはあまり簡単ではない。また、外出先などのモバイル環境でアクセスすることを想定すると、通信量を削減することも重要になってくる。時間やコストがかかりすぎては、「便利で使いやすいツール」とは言えなくなってしまう。

サイボウズ ポケット 4は、こうした要素に特化してチューニングされた、「PDA専用インターフェイス」だと考えればよいだろう。

●データはサイボウズOffice 4と共通

サイボウズ ポケット 4は、サイボウズOffice 4のオプションモジュールという位置付けなので、実際のアプリケーション機能はサイボウズOffice 4と同等となる。ただし、実装上はそれぞれ独自の実行ファイルを呼び出すようになっている。一方、当然だがデータは共通で、サイボウズOffice 4で入力したデータをサイボウズ ポケット 4で読み出したり、逆にサイボウズ ポケット 4から入力したデータをサイボウズOffice 4にアクセスして読み出すといった操作が可能だ。

サイボウズ ポケット 4では、アクセスに使われているデバイスを自動認識する、といった機能は含まれていない。このため、サイボウズOffice 4とサイボウズ ポケット 4の使い分けは、アクセスするURLを変更することで行なう。そのため、LAN内のPCからサイボウズ ポケット 4にアクセスしてPDA用のインターフェイスを利用する、といった使い方も可能である。

本来想定された利用法とは異なると思われるが、PCのWebブラウザでサイボウズ ポケット 4を利用すると、素っ気ないほどシンプルな画面に必要最小限の情報が表示されるので、これはこれでなかなか使いやすい。必ずしも「PDA専用」と考える必要はなく、小型軽量の新しいユーザーインターフェイスと捉えて、ユーザー側で活用方法を工夫してみても面白いのではないだろうか。

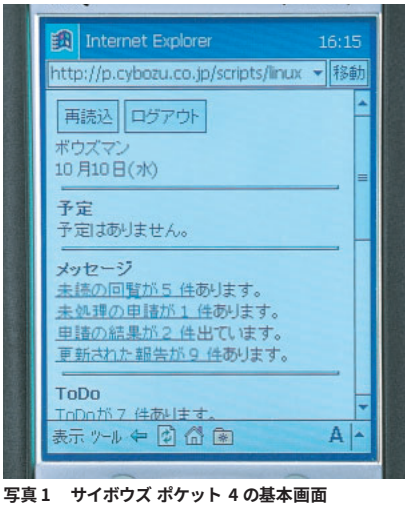


写真1　サイボウズ ポケット 4の基本画面

画面1　PCのWebブラウザでアクセスした場合のサイボウズOffice4の基本画面。PC版では一画面に豊富な情報がビジュアルに盛り込まれるが、サイボウズ ポケット 4では一転して簡潔さが追求される

サイボウズ ポケット 4 機能紹介

●PC版と機能は同一

実装面はともかく、利用上サイボウズ ポケット 4はサイボウズOffice 4のフロントエンドと見なすことができる。このため、利用できる機能についても、サイボウズOffice 4と同等と考えてよいが、唯一例外となるのはメールの添付ファイルだ。サイボウズOffice 4ではWebメールが利用でき、この機能自体はサイボウズ ポケット 4でも当然利用できるのだが、添付ファイルはサポートしていない。これは、PDAでは添付ファイルを開くことができるアプリケーションが備わっていない可能性も高いためで、仕方のないところだ。

このほか、機能面では同一だが、見た目の印象に関しては大きく異なっている。これは、サイボウズ ポケット 4がデータ量を最小限に抑えることを重視した設計になっているためだ。実際、データ量の削減についてはかなり徹底的なチューニングが行われている。たとえば、サイボウズOffice 4にPCでアクセスした場合には、アプリケーションの選択は画面上に表示されるアイコンをクリックすることで行うが、サイボウズ ポケット 4では、単純なテキストリンクが表示される。しかも、画面上には表示されないが、リンク先となるアプリケーション名はほとんどが1文字の名前となっている。これは、リンクを示すURL文字列が長くなると、それだけ転送するデータ量が増えてしまうからだ。

PDAは、PCに比べてメモリも少ないし、CPUの処理能力でも劣る。何より、モバイルでの通信を考えるとデータ量が増えるとそれだけ通信時間が長くなったり、通信コストが高くついてしまうことになる。同じ機能を実現することを重視する一方で、極力シンプルに実装することを目指している点がサイボウズ ポケット 4の最大の特徴といえるだろう。

●閲覧機能を重視した画面設計

サイボウズ ポケット 4では、特に情報を閲覧しやすいことを優先して画面設計が行われている。モバイル環境でもワイヤレス接続を利用することにより、機能面ではPCのWebブラウザを利用するのとほぼ同じ環境が実現できる。ただし、実際の利用を考えると、さすがにPCでアクセスするのとまったく同じ使われ方をするとは思えない面がある。もっとも目立つ点は、データ入力の困難さである。フルキーボードやマウスを備えたPCでは、長文の入力も別段苦にはならないが、PDAで文字入力するのはあまり楽なこととはいえない。これはもちろん、サイボウズの問題ではなく、PDA側の制約である。

このため、現実の利用ではPDAから情報を入力するよりも、素早く情報を閲覧することが主たる用途になると割り切って「見やすい画面」を意識して構成されている。

PDAの特性を考慮して、縦スクロールを多用する一方で、横スクロールは基本的に使わないなど、PDAの機能やインターフェイスを踏まえて最適化を行なった画面は、直観的に使えるわかりやすさと便利さを備えている。

写真2 ログイン画面

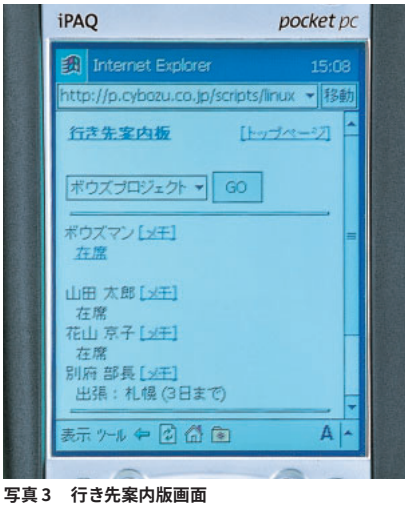


写真3 行き先案内版画面

写真4 週間スケジュールは表形式で表示される

システム構築の具体例

●サイボウズ ポケット 4 とセキュリティ

データの表示形式はごく一般的なHTMLを用いて記述され、特別な機能は使われていない。このため、ほとんどのWebブラウザが動くPDAから利用できる。通信に関しても同様で、モバイルアクセスのための特別な機能があるわけではなく、単にHTTPでアクセスするだけという構造である。したがって、システム構成としてもごく単純で、外部に公開しているWebサーバ上でサイボウズOffice 4とサイボウズ ポケット 4が稼働していればよいだけだ。

ただし、サイボウズOffice 4ではLAN内やイントラネットでの利用が多いと思われるが、モバイル環境での利用を想定したサイボウズ ポケット 4では、逆に社外からのアクセスが主となる可能性が高い。社外からのアクセスにインターネットを利用する場合、セキュリティ対策が不可欠となるが、サイボウズ ポケット 4自体には特別なセキュリティ機能は設けられていない。利用しているのがあくまでも一般的なインターネット標準であるため、どのようなセキュリティソリューションとも組み合わせて利用できるからだ。ファイアウォールなどを適切に利用し、安全な運用環境を構築するよう心がけたい。

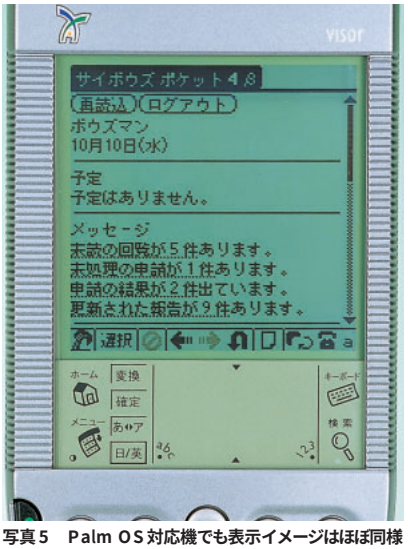


写真5　Palm OS対応機でも表示イメージはほぼ同様

●サイボウズ ポケット 4 の運用例

セキュリティを重視する場合、サイボウズに格納されたデータにインターネットから直接アクセスするのは禁止するのが一番確実な手法であろう。この方針とPDAからのモバイルアクセスを両立させるためには、独自のRASサーバを運用する手が考えられる。

PDAからどのような手段でリモートアクセスを実現するかにもよるのだが、一般的にはPHSデータカードを利用するのが便利だろう。CFカードタイプの製品が各キャリアから発売されており、ケーブルレスでスマートな接続が実現できる。これを使うのであれば、オフィスにISDN回線を用意し、PIAFS対応のTAを設置するだけで、PHS用のアクセスポイントを提供できる。機種にもよるが、TAでは接続先の電話番号に基づいたアクセス制御を行えるのが一般的なので、利用するモバイルユーザーが使用している電話番号からだけ着信を許可するようにすれば、セキュリティ面ではかなり安心できるだろう。

このほか、使用機種が限定されるが、サイボウズシンク4 for PalmOSやシャープのZaurusとの連係機能を用いて必要なデータをあらかじめPDAに格納してしまう手もある。この場合、データベースの必要な部分をあらかじめコピーしておき、リアルタイムで参照したい最新メッセージのみをサイボウズ ポケット 4から読み出す、という使い分けを行うことで、情報の機密性に応じてアクセス方法を変えることも可能だろう。

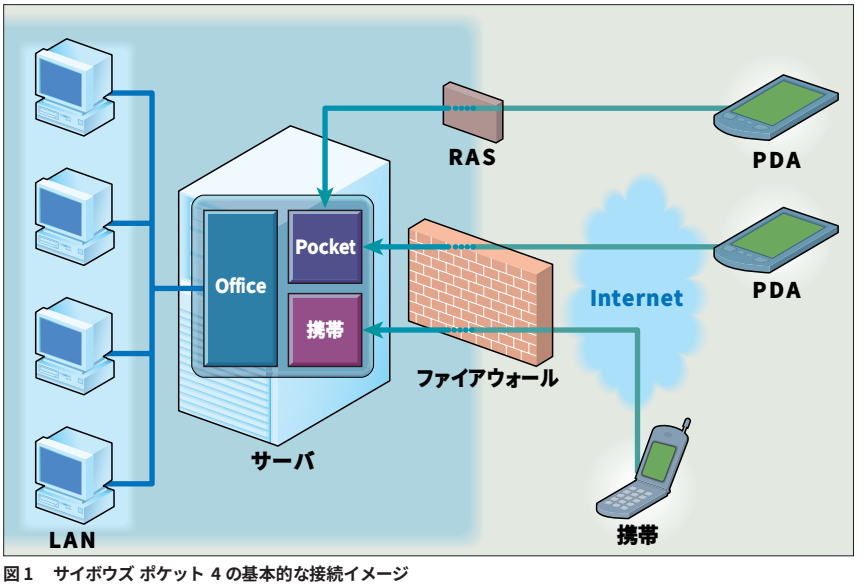


図1　サイボウズ ポケット 4の基本的な接続イメージ
サイボウズOffice4やサイボウズ ポケット 4はWebベースでのアクセスさえ確保されていれば利用できる。このため、インターネットからアクセスすることも可能だし、RASサーバを介してアクセスする構成も採れる。

Interview

サイボウズ ポケット 4 の開発のねらい

●デザイン上のポイント

サイボウズOffice 4に備わる、文書管理を除く全12個のアプリケーションを、なるべくもとの機能を損なわずにPDAで効率的に利用できるようにするためには、データ転送量など先に述べた技術的な工夫のほかに、インターフェイスのデザイン設計上の工夫が大変重要になる。この点について、サイボウズのプロダクトマネージャ・石谷雄一郎氏は「外出中に自分がこなさなければならない仕事の流れがひと目でわかる」ことを最大の目標に設計を行ったと語る。

具体的には、情報の優先順位に基づいて画面を配置している。ログイン後の初期画面では、

- **自分の予定**
- **自分が処理しなくてはいけない仕事**
- **ほかのメンバーから依頼された仕事**
- **自分が設定したそのほかの仕事**

が表示され、最後にアプリケーションメニューが表示される。特別な操作をしなくても、重要な情報がまず目に飛び込んでくるように工夫されているわけだ。使いやすさの最優先事項として「読みやすさ」を掲げ、逆に「書きやすさ」を犠牲にした部分もあるそうだが、ユーザーの利用形態を考えれば正解だと思われる。

対象ユーザーとその業務に関しては、「35歳前後の課長／主任クラスの方々に、その人たちが社内にとどまっていないと進まなかった仕事を、サイボウズとPHSというインフラを利用することで外出先でも滞りなく進むようにしてもらいたい。これによって、自分の仕事も進むし、社内の他のスタッフの仕事も滞留させないようにする」ことを狙ったという。

ただし、12アプリケーションをサポートしたことからもわかるとおり、機能面では可能な限り実装が行なわれている。同様にモバイル用途向けにリリースされた携帯電話向けのサイボウズケータイ4では必要最小限の情報が見られることを意図してコンパクトにまとめたが、「サイボウズ ポケット 4では、ちょっとした出張での利用にも耐えられるはず」とのことで、外出中でも業務を進められるように、という配慮が徹底されている。

●マルチプラットフォーム

サイボウズ ポケット 4は、「SJIS漢字コードに対応した、HTML 3.2対応のWebブラウザが動く機種」を動作対象としている。動作確認はPocketPC、Zaurus、Palmで行われているが、これ以外にも先の条件を満たす機種ならまず問題ないはずだ。さまざまな機種で利用できるようにするために表示はモノクロとし、機種ごとの画面サイズの違いなどはHTMLのレベルで吸収するようになっている。

写真6　コンパックのPDA iPAQとC@rd H" 64

接続に関しても特に機種やキャリアに依存するような処理は行っていないため、どのような環境でも利用可能だが、現実的にはRASによるイントラネット直通の接続が主流だと想定しているという。

●今後の展開

Web Serviceなどが話題を集めていることからもわかるとおり、今後はインターネットを利用したサービスがますます重要な存在になっていくだろう。サイボウズでも当然、こうした展開を視野に入れて次期バージョンの検討を行っている。

具体的には、ASPサービスへの展開や、SOAP／UDDIとは限らないにしても何らかの形で外部からサイボウズの機能を呼び出せるようなインターフェイスを用意する、といった可能性もある。

（渡邉利和）

サイボウズ株式会社
http://cybozu.co.jp/products/pt4/

サイボウズ株式会社
企画部プロダクトマネージャ
石谷 雄一郎

DCC／CG業務向け Linuxワークステーション hp Linux workstation x2000／x4000

日本ヒューレット・パッカードは、Xeon、Pentium 4搭載のLinuxワークステーション「hp Linux workstation x2000／x4000」の2機種を発売する。

製品は、x2000がCPU Pentium 4に、最大2Gバイトのメモリを搭載、Intel 850チップセットを採用しているのに対して、x4000は、CPUがデュアル対応のXeonで、最大4Gバイトのメモリを搭載、Intel 860チップセットを採用している。Linuxは、プリインストールされ、DCC（デジタル・コンテンツ・クリエーター）、CG向けのワークステーションとして最適に設定した状態で出荷される。

いずれも高度なOpenGL 3Dパフォーマンスを提供するためにATI Fire GL2、ATI Fire GL4などのグラフィックス・カードに対応した。ATI Fire GL4を使用すればデュアル・ヘッド（モニタ2台）にできる。同時に、XFree86やX Serverにヒューレット・パッカード社独自の拡張を施し、色階調の定義、カラー・テーブルおよび簡単にモニタのガンマ補正ができるX Server APIを作成した。

また、ソフトウェアベンダーとも協力し、順次アプリケーションの検証作業を進めている。現段階で、Avid社 の Softimage XSI 2.0、 Alias｜Wavefront社の Maya 3.0、 NothingReal社の Shakeの3製品の動作検証が完了している。

製品には、ハードウェア3年間翌日オンラインサポートとLinux OSの1年間の電話支援サポートがつく。モニタは別売。

発売日	2001年12月1日
発売	日本ヒューレット・パッカード株式会社
TEL	0120-081565
価格	60万8000円～
URL	http://www.jpn.hp.com/go/workstation/

VA Linux Systemsジャパン、SourceForgeパイロットユーザー版の提供を開始
2001年10月23日

VA Linux Systemsジャパンは10月22日、ソフトウェア共同開発支援システム「SourceForgeパイロットユーザー版」の提供を開始した。同時に、アイアイジェイテクノロジー（以下IIJ-T）と同システムの導入契約が成立したことを発表した。

VA Linux Systemsジャパンの「SourceForge」は、米VA Linux Systemsが運営するオープンソフトウェア開発支援サイトSourceForge.netのシステムをベースとし、独自の導入支援サポートなどが追加されている。

「SourceForge」には、「SIエディション」と、「エンタープライズエディション」の2種類があるが、今回、IIJ-Tに提供されるのはSIエディション。SI向けの操作性が強化されており、メールによるバグ登録などが行える。IIJ-Tでは、同システムの導入によって業務の効率化に取り組んでいきたいとしている。

近日中には、エンタープライズエディションのパイロットユーザー版も提供される予定。

VA Linux Systemsジャパン
(http://www.valinux.co.jp/)
アイアイジェイテクノロジー
(http://www.iij-tech.co.jp/)

OSDL、Linuxシステムのパフォーマンスを検証するサービスを提供
2001年10月18日

米Open Source Development Lab（以下OSDL）は、オープンソース開発者向けに、Linuxシステムのパフォーマンスをさまざまなハードウェアやテスト方法で検証するサービス「STP（Scalable Test Platform）」の提供を開始した。

「STP」サービスでは、ユーザーがOSDLのテストラボに送ったカーネルのパッチやアップグレードと指定したLinuxカーネルを、指定された構成のハードウェアでコンパイルし、ユーザーが選択したパッケージとともにインストールしてパフォーマンスを検証する。検証結果は、ユーザーにメールと専用Webページで提供され、参考データとしてOSDLのサーバに保存される。ユーザーは検証結果をOSDLのWebサイトに掲載して一般に公開することも選択できる。

OSDLのテストラボは、1Wayから16Way構成の32ビットおよび64ビットのCPU、2Tバイトのストレージなどが利用可能になっており、ユーザーは、Webサーバやファイルサーバ、データベースシステムなどについて、さまざまな測定方法、カーネルツリー、ハードウェア構成などを選んで検証することができる。

Open Source Development Lab
(http://www.osdl.org/)

ノーザンライツとミラクル・リナックス、クラスタリングソフトLifeKeeperで提携
2001年10月18日

ノーザンライツコンピュータとミラクル・リナックスは10月18日、米SteelEye Technologyのクラスタリングソフト「LifeKeeper for Linux」の国内投入において、共同で検証およびマーケティングを行うと発表した。

ノーザンライツが提供する「LifeKeeper for Linux」とOracle8i／9iとの接続検証を行い、Linuxクラスタリングシステムによる安定性、信頼性の高い企業向けソリューションを提供する。

ノーザンライツコンピュータ
(http://www.nlcomputer.com/)
ミラクル・リナックス
(http://www.miraclelinux.com/)

LPI-J、Linux技術者認定試験レベル2日本語版を11月28日より開始
2001年10月12日

特定非営利活動法人LPI-Japan（以下、LPI-J）は、「Linux技術者認定試験」（以下、LPIC）レベル2の日本語試験を11月28日より開始する。

LPI-Jは、米国を中心に世界共通基準のLinux技術者認定試験を推進しているLPIの日本法人で、“LPIC日本語試験”を実施している。LPICはレベル1からレベル3まであり、各レベル2科目の試験が行われる。現在LPI-Jが実施しているのは、LPICレベル1の日本語版（101、102の2科目）、レベル2ベータ（1科目のみ）の英語版試験（10月末まで）。

LPICの位置づけについて、LPI-Jの広報担当者によると、「LPICは各ディストリビューターが行う資格試験と異なり、あらゆるディストリビューションに共通したLinuxのもっとも基礎となる知識をテストするもの」であるとしている。

実際の試験は、米NCS Pearsonの電子テストサービス事業部門であるVUEが行い、国内43カ所の公開テストセンターで実施される。日本語版試験はレベル1（101、102）、レベル2（201、202）とも1科目につき1万5000円。なお、現在公開されているレベル2ベータ英語版は84ドル。

LPI-Jでは、来年以降にレベル3日本語版試験を開始する予定。また、今後専門学校などの教育機関がLPIC向けコースを設置する際のサポートなども行ってゆくとしている。

LPI-Japan（http://lpi.or.jp/）

日本IBMとシチズンがLinuxウォッチ「WatchPad」を共同開発
2001年10月11日

日本IBMとシチズン時計は、Linuxを搭載した腕時計型コンピュータ「WatchPad」を協同で開発することを発表し、試作機を公開した。

「WatchPad」は、これまでIBMリサーチが中心に開発を進めてきた、Linux搭載腕時計型コンピュータ「スマート・ウォッチ」や「Linux Watch」をベースに、日本IBMが全体の概念やハード／ソフトウェアを含めたアーキテクチャとシステム設計、OSの実装を行い、シチズンが外装を含めたパッケージング設計、表示素子や入力デバイスなどの部品設計を行っている。

これまでの「Linux Watch」はカーネル2.2とX11R6の組み合わせであったが、今回はより小さなリソースで動作させるた

め、組み込みデバイス向けのウィンドウシステムで、オープンソースで開発されている「Microwindows」を採用している。また、これまでのローラー型の入力デバイスに代わって搭載されたリューズ型入力デバイスは、防水性を高めるためにシチズンの発案で搭載された。

両社では「WatchPad」の使用例として、ノートPCや携帯電話を鞄の中にしまったままで、Bluetooth経由で操作したり、指紋による個人認証機能を用いてホテルや空港のチェックインを自動的に行うといったことを想定している。今後、シチズンは製品化も視野に入れた研究開発を行うとしており、日本IBMはシチズンに対する技術協力の提供や、大学などの研究機関へハードウェアや開発キットなどを提供する。

日本IBM（http://www.ibm.com/jp/）
シチズン（http://www.citizen.co.jp/）

日本Pythonユーザー会（PyJUG）のホームページ開設

2001年10月4日

フリーのオブジェクト指向プログラミング言語「Python」のユーザー組織である「日本Pythonユーザー会（PyJUG）」のホームページが開設された。

運営はPyJUGの下部組織「PyJUG網元衆」が行い、サーバやネットワークリソースは一件楽着インターネットサービスが提供する。Pythonで開発されたアプリケーションサーバ「Zope」が利用されている。

ホームページ内には、Pythonの基礎的情報、技術解説、活用事例が用意されているほか、Pythonによる開発を支援するソフトウェアの配布も行うという。さらに「オープン記念」として、Pythonメーリングリスト「Python-ml-jp」の全文検索ページも開始された。

PyJUG（http://www.python.jp/）

米Sun Microsystems、「StarOffice 6.0 Beta Software」を公開

2001年10月3日

米Sun Microsystemsは10月2日、「StarOffice 6.0 Beta Software」（アジア地域では「StarSuite 6.0」）を公開した。旧バージョンからの主な変更は以下のとおり。

- XMLファイルのサポート
- 日本語、韓国語、中国語（簡体字、繁体字）サポート
- Microsoft Officeとの互換性強化
- ヘルプの刷新
- 統合デスクトップを削除し、GNOMEなどの統合デスクトップ環境との親和性を強化

「StarOffice 6.0 Beta Software」は、Solaris、Linux、Windows 95／98／Me／NT／2000で動作する。Linux版の動作環境は以下のとおり。

- カーネル2.2.13以上
- glibc 2.1.2以上
- 800×600以上の解像度、256色以上のX Server
- CPU Pentium互換
- HDD 250Mバイトの空き容量
- メモリ 64Mバイト

バイナリのダウンロードは米Sun MicrosystemsのWebサイトから可能で、すべてダウンロードすると133Mバイト程度になる。ソースコードはOpen Office.orgにて入手可能だ。

米Sun Microsystems
（http://www.sun.com/）
OpenOffice.org（http://www.openoffice.org/）

専門家がLinuxのウィルスの危険性を警告

2001年9月28日

コンピュータウィルスに関する国際会議「Virus Bulletin（ウィルスブリテン）2001」が9月27日（現地時間）、チェコのプラハで始まった。

初日には、会議を主催するVirus Bulletin誌編集者のヘレン・マーティン氏の開会宣言に続き、企業におけるウィルス対策などが中心のCorporate Streamと、技術的なテーマを取り上げるTechnical Streamの2つのトラックに分かれて、14の発表が行われた。

Virus Bulletinの常連発表者という米マイクロソフト社のランディ・エイブラムス氏は、Changes In Retail Virus Scanning 1998-2001と題して、同氏が'98年に行った小売りソフトにおけるウィルスの検知に関するレポートのアップデートを行った。

エイブラムス氏によれば、この4年間にハードウェアの高速化が進み、より進んだ処理が可能になったが、その間にウィルスの発生量も増え、ウィルス情報の更新もより頻繁に行うことが必要となっているという。

コンピュータ・アソシエイツ社（オーストラリア）のヤコブ・カミンスキー氏はNot So Quiet On The Linux Front: Linux Malware IIというタイトルで、Linuxプラットフォームにおけるウィルスの危険性についてレポートした。カミンスキー氏はまず、Linuxのバイナリベースのウィルスリストを示し、2000年6月には13であったものが、半年ごとに50%ずつ増加し、2001年1月には31になっているというデータを紹介した。続いてそのウィルスの中からいくつかを特徴とともに紹介し、Linuxのカーネルライブラリに感染するものも登場しているとした。

また、バイナリ以外にも、27のシェルスクリプトをパックしたRamen、Admw0rm、Lion、Adoreなどいくつものワームが登場しているという。

米ボーイング社のジャネット・ジャービス氏は、企業のウィルス対策管理者としての立場から、企業においてどのような方針でウィルス対策を行うべきかを発表した。

Virus Bulletin 2001
（http://www.virusbtn.com/vb2001/）
Virus Bulletin（http://www.virusbtn.com/）

Hardware　発売日　2001年9月17日

Itanium搭載サーバ HA8000-ex／780

URL　http://www.hitachi.co.jp/ha8000/

発売	株式会社日立製作所
TEL	0120-2580-12
価格	475万円〜

日立製作所は、日立アドバンストサーバHA8000-exシリーズでItaniumプロセッサを最大4個搭載可能な「HA8000-ex／780」を発売した。

HA8000-ex／780は、CPUにItanium 800MHzを採用し、各CPUごとに2Mバイト／4Mバイトの3次キャッシュを使用することで高性能化を実現している。メモリは最大32Gバイトまで拡張でき、64ビットアドレッシングを活かした大規模アプリケーションを実行できる。

HDDには73Gバイト Ultra160 SCSI HDDを使用（最大2基146Gバイト搭載可能）。RAID1に対応することで信頼性の向上を図っている。OSはRed Hat Linux 7.1 for Itanium Processorに対応している。

Hardware　発売日　2001年9月19日

A3ワイド・タンデムカラーレーザープリンタ Phaser 7700

URL　http://www.phaser.co.jp/

発売	富士ゼロックスプリンティングシステムズ株式会社
TEL	03-3448-3040
価格	99万8000円

富士ゼロックスプリンティングシステムズは、4連タンデムカラーレーザーエンジンを採用したカラープリンタ「Phaser 7700」を発売した。

Phaser 7700はPhaserシリーズの最高機種で、グラフィック向け機能を強化している。A3ワイドサイズ（320×450mm）、Adobe PostScript 3に対応し、メモリ128Mバイト、100Base-TX、内蔵HDD、カラー補正機能などを標準で装備している。印刷速度は、A4横、連続印刷でカラー／モノクロともに毎分22枚を実現した。500MHz PowerPC G4相当のプロセッサを搭載して、高速処理を可能にしている。

また、オフィス用途としてオフセット機能つきの自動両面プリント、丁合、保存プリント、機密文書をパスワードで保護するセキュアプリントなどの機能や最大3500枚まで拡張できる給紙トレイやステープラやスタッカ処理用のフィニッシャを装着することで大容量のオンデマンド印刷にも対応している。

Hardware　発売日　2001年10月1日

情報家電・端末制御機能をもつLinux BOX L-BOX

URL　http://www.nttcom.co.jp/

発売	NTTコムウェア株式会社
TEL	043-211-2404
価格	要問合せ

NTTコムウェアは、高速・大容量通信に対応した、多目的Linux BOX「L-BOX」を発売した。

L-BOXは、ネットワークと情報端末のインターフェイスとして利用する製品で、ADSLやCATVなどの常時接続用ファイアウォールマシンや、情報家電や情報端末に対する制御ボックスになる。PCカードやモバイルカードをL-BOXのカードスロットに差し込めば簡単に無線LANやモバイル環境でも利用できる。内部OSにLinuxを採用しているので、アプリケーションの拡張が容易にできる。

i486相当128MHzのCPUを使用し、16Mバイト（最大80Mバイト）のメモリを装備、基本実装4Mバイトのマイクロディスクモジュール（最大32Mバイト）とオプションのコンパクトフラッシュの外部記憶装置がある。筐体のサイズは、119mm（H）×67mm（W）×98（D)mm。

Hardware　発売日　2001年10月4日

プラグ＆プレイで実現する次世代VPNソリューション SSH Complete VPN

URL　http://www.ipsec.co.jp/

発売	SSHコミュニケーションズ・セキュリティ株式会社
TEL	03-3459-6830
価格	98万円（SSH VPN Gateway）／78万円（SSH Central Manager 5ユーザーライセンス）

SSHコミュニケーションズ・セキュリティは、プラグ＆プレイでインターネットセキュリティ環境を実現するVPNソリューションの新製品「SSH Complete VPN」を発売した。

SSH Complete VPNは、SSH VPN Gateway（サイト間の接続用VPN装置）、SSH Sentinel（モバイルユーザー用VPNクライアントソフトウェア）、SSH Central Manager（VPN管理ツール）の3つのコンポーネントで構成されている。

SSH VPN GatewayとモバイルSSH Sentinelは、設置時に別に送られてくるスマートカードを差し込むだけですべての初期設定が完了する。いったん、インターネットに接続されると同一VPN内でIPSecによる暗号化通信が行われ、各地に点在する企業ネットワークを安全にSSH Central Managerから集中管理できるようになる。

端末側は、スマートカードに記録してある秘密鍵が合わなければ作動しないので、スマートカードを別に送ることで設置前の機器の配送ミスや盗難等による情報漏洩の危険性を防いでいる。

Hardware　発売日　2001年10月9日

DVD-RAMドライブ搭載小型ワークステーション Vectran 2000GLM-LNX

URL　http://www.arcbrain.co.jp/

発売　株式会社アークブレイン
TEL　03-3375-8968
価格　37万8000円

アークブレインは、ミニタワーにIntel Pentium 4 2GHzを搭載したVectoran 2000GMLシリーズを発売した。

今回発売されたのは、Windows 2000 ProfessionalをプリインストールしたW2PとRed Hat Linux 7.1をプリインストールしたLNXの2機種。主な仕様は、Pentium 4 2GHz、メモリ1Gバイト（最大1.5Gバイト）、UltraATA/100対応80GバイトHDD、DVD-RAM、100Base-TXなど。ビデオカードにGeForce3チップを採用したELSA製のGLADIAC 920を搭載し、3Dグラフィックを用いたプレゼンテーションなどのマルチメディア環境に対応した設計になっている。筐体のサイズは329mm（H）×136mm（W）×340mm（D）。

OSについては、上記2製品のほか、Turbolinux、FreeBSD、Solaris 8などに個別に対応することもできる。

Hardware　発売日　2001年10月31日

ファイアウォール・アプライアンス Symantec VelociRaptor ver. 1.1

URL　http://www.symantec.com/

発売　株式会社シマンテック
TEL　03-3476-1426
価格　72万円（50ノードライセンス）～

シマンテックは、同社初のハードウェア製品となるファイアウォール機器「Symantec VelociRaptor ver. 1.1」を発売した。

製品は1Uラックマウントサイズの筐体にLinuxベースのOSとファイアウォール、VPNソフトをプリインストールしたもので、これを用いて各地に分散したマシンや内部ネットワークを、インターネットなど外部ネットワークを用いて安全に接続することができる。

ファイアウォールには、パケットフィルタリング方式のほかに、バッファオーバーフローやバックドアコマンドなどのアプリケーションレイヤーレベルでの攻撃を検出できるアプリケーションプロキシ方式を採用した。

管理はWindows 2000／NT上にインストールされるSymantec Raptor Management Consoleを使用してローカル、リモートに設置した複数のVelociRaptorを一元管理する。

Hardware　発売日　2001年11月1日

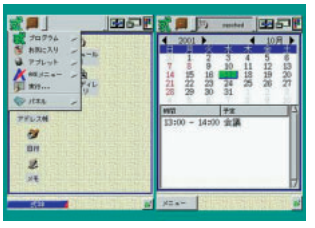
iPAQ上でLinuxを稼動 Melon

URL　http://melon.10art-ni.co.jp/

発売　株式会社テンアートニ
TEL　—
価格　9800円

テンアートニは、コンパックコンピュータのPDA iPAQ上で利用できる、コンパクトフラッシュカード型Linuxキット「Melon」を発売した。

Melonは、コンパクトフラッシュカード（容量64Mバイト）に、オープンソースのディストリビューションFamiliarをベースとしたLinuxを組み込み、PDAに装着するだけで既存の環境を書き替えずにLinuxを起動できるようにした製品。GUIには、同社が業務提携したアックス社の「式神（しきがみ）」を採用し、手書き文字認識による日本語入力環境を装備している。このためスケジューラなどのPIM機能を直ちに日本語で利用できる。



簡単にPDA用のLinux環境が作れるので、Linux対応PDAのアプリケーションの開発にも役に立つだろう。対応しているコンパックのPDAは、iPAQ Pocket PC H3630とH3660の2機種。

Software　発売日　2001年9月17日

効率的文書管理を実現するナレッジバンクサーバ Kacis Cabinet Ver.2.0

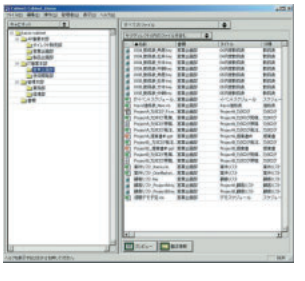
URL　http://www.mvi.co.jp/

発売　株式会社メディアヴィジョン
TEL　03-3222-3908
価格　38万円（10ライセンス）～

メディアヴィジョンは、カシスと共同開発したナレッジバンクサーバ「Kacis Cabinet Ver.2.0」を9月17日より発売した。

Kacis Cabinet Ver.2.0は、Word、一太郎、Excel、PDFなどのあらゆるドキュメントを分類整理し、自由な並べ替え、検索、再利用を行うことによって、ドキュメントの有効利用を可能にするサーバ・クライアントシステム。

Kacis Cabinetは、キャビネットカードにファイルの分類事項を記入、登録してファイルを一元管理する。ファイルはキャビネットカードに記入した情報をもとに、部門別、製作者別、日付別、ジャンル別など目的に応じたフォルダ分類を自由に行える。全文検索、キャビネットカードの検索、AND検索、OR検索などの検索機能によって柔軟なファイル検索が可能。Webブラウザからのファイルの検索、登録、ダウンロードもできる。対応OSはTurbolinux Server 6.5。



Software　発売日　2001年9月27日

高性能サーバ向け仮想化ソフトウェア VMware ESX Server

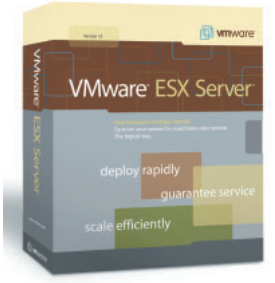
URL　http://www.networld.co.jp/

発売	株式会社ネットワールド
TEL	—
価格	37万5000円（1-2CPU）～

ネットワールドは、高性能サーバ向け仮想化ソフトウェア「VMware ESX Server」を発売した。

ESX Serverは、エンタープライズ規模のサーバを集約し、それをパーティショニングする仮想化ソフトウェア。1台のマシン上で複数のサーバアプリケーションを実行するという点はこれまでのVMwareと同じだが、ESX Serverは、サーバハードウェア上で直接動作するためホストOSを必要としない。そのため、ハードウェア資源をVMwareが完全に管理できるので、複数の仮想マシンに対してきめ細かいコントロールが可能になった。

ESX Serverには、起動と管理を行うためのLinuxベースのコンソールOSが組み込まれており、コンソール画面か、Webブラウザから管理タスクを実行する。各アプリケーションは、このコンソールOSではなく、仮想マシン上で動作する。Intelアーキテクチャのマシン上で稼動し、最大8プロセッサのサーバマシンに対応できる。



Software　発売日　2001年10月15日

Linuxに対応した不正アクセス監視ツール Intruder Alert 3.5.1J

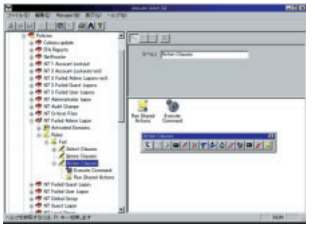
URL　http://www.elnis.com/

発売	日新電機株式会社
TEL	03-5821-5914
価格	74万円～

日新電機は、シマンテック社製不正アクセス監視ツール「Intruder Alert」を機能強化し、「Intruder Alert 3.5.1J」として販売する。

Intruder Alertは、サーバなどのマシンに常駐して監視するホストベースの不正アクセス監視ツールで、Webページの書き替えやバックドア、トロイの木馬による不正アクセスを検出できる。監視対象となるマシンにエージェントソフトを常駐させ、各種ログをモニタし、マネージャソフトがログインやファイル操作の中から、不正もしくは不審なイベントを抽出し、警告表示、ログの保存など、必要な措置を自動実行する。

商用UNIX、Windows NTをはじめ、26種類以上のOSに対応し、1台のマネージャで100台までのエージェントを管理できる。3.5.1Jで、新たにRed Hat Linux 6.2に対応した。また、Webサーバなど外部サーバのログと連携してWebサーバへの不正アクセスへの対応もできる。



Software　発売日　2001年10月中旬

WebSphere対応XML＆JSPベースの販売管理ソフト ClassCat WebSales 1.2 Petit for WebSphere（DB2 version）

URL　http://www.azi.co.jp/

発売	クラスキャット ネットワークス株式会社
TEL	—
価格	7万5000円

クラスキャット ネットワークスは、Webベースの販売管理ソフトウェアClassCat WebSales 1.2 Petit for WebShpere（DB2 version）を発売した。

製品は、Webサーバ上で動作する販売管理パッケージソフトウェアで、ブラウザで受注データを入力したり、問い合わせができる。管理データをExcel形式に直接出力できるので市販している財務会計ソフトウェアとの連携も容易だ。XML技術を用いてデータ処理の多くをローカルコンピュータ側で済ませるので、サーバ側の負荷が減り、速度の遅いネットワークでも使用できる。

主な機能は、受注管理、顧客・商品管理、システム管理／ユーザー管理など。動作環境はIBM WebSphere Application Server V4.0とIBM DB2 UDB V7.1が必要なほか、OSは、LinuxとWindows 2000に対応している。

Software　発売日　2001年10月22日

Red Hat Linux 7.2で動作するシステム管理製品 Net-DMS for Red Hat Linux Ver1.5

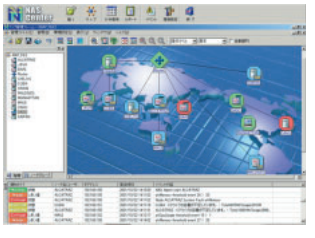
URL　http://www.nuritelecom.co.jp/

発売	ヌリテレコム株式会社
TEL	—
価格	98万円～

ヌリテレコムは、システム運用管理製品「Net-DMS for Red Hat Linux Ver1.5」を発売した。

Net-DMSは、管理対象となるPCに導入するNASエージェントと、エージェントが監視している情報を集中管理するマネージャサーバそして管理者が利用するコンソールで構成されている。管理はシステムコンソールの管理用アイコンと、イベントコンソールで行う。

Net-DMSでは各PCのメモリ、CPU、ディスクのしきい値、ネットワークインターフェイスの監視、Windowsマシンの遠隔制御などのほか、企業の組織構成をもとにハードウェアを管理でき、組織ごとの資産一覧レポートの出力やソフトウェア配布が行える。マネージャサーバにRed Hat Linux 7.2とRed Hat Databaseを採用している。シンプルに必要な項目だけを管理できる製品となっている。



Software　　発売日　2001年10月24日

Oracleデータベース Linux版
Oracle9i Database Standard／Enterprise Edition for Linux

URL　http://www.oracle.co.jp/9i/

発売	日本オラクル株式会社
TEL	03-3511-5331
価格	3万2000円（5ユーザー）～

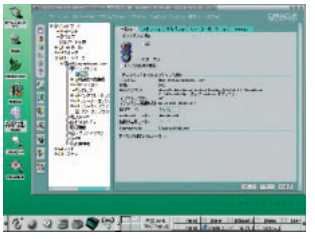
日本オラクルは、次世代のOracleデータベース「Oralce9i Database」のLinux版を発売した。

今回発売されるのは、ワークグループや、部門単位での使用を想定したStandard Editionと企業レベルでの使用を想定したEnterprise Editionの2種類。

8iで培ったインターネットデータベースとしての実績を継承しつつ、Oracle9i Real Application Clustersの採用による可用性、パフォーマンスの強化、Oracle9i Data Guardによる事故等の計画外停止に対する保護機能強化、Oracle Internet File Systemによる多様なデータへの対応などが施されている。

特にLinux版では、Linuxカーネル2.4と組み合わせることで、最大64Gバイトの大容量メモリを実装可能にし、最大64Gバイトのデータファイルを作成できるLFS（Large File System）に対応した。

Linux OSは、MIRACLE LINUX Standard Edition V2.0が対応している。



Software　　発売日　2001年10月31日

OSの混在環境に対応、複数台のサーバを集中的に運用管理
HDE Center

URL　http://www.hde.co.jp/center/

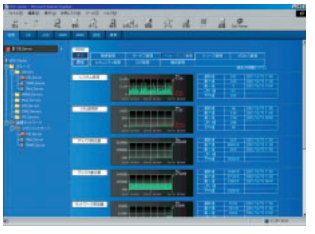
発売	株式会社ホライズン・デジタル・エンタープライズ
TEL	03-5738-5410
価格	72万円（5ノード管理）～

ホライズン・デジタル・エンタープライズは、OSの混在する複数のサーバを集中的に運用管理するためのソフトウェア「HDE Center」を発売した。

HDE Centerの主な機能は、サービス、プロセス、パフォーマンス、リソース、セキュリティ、ログ、障害、ハードウェア、ソフトウェア構成の管理と管理項目検索機能など。

監視項目をひと目で確認でき、発生した障害は自動的にメールで通知する。メールはその障害の度合いに応じて適切な通知先を指定できる。

OSの異なる複数サーバの管理で避けられない、各OSの学習にかかる時間や作業の煩雑化による人為的なミスを軽減する。対応するOSはRed Hat Linux 6.2J／7.1、Turbolinux Server 6.1／6.5、Miracle Linux Standard Edition Version 1.1、Solaris 8。その他のOSにも順次対応予定。



Software　　発売日　2001年11月2日

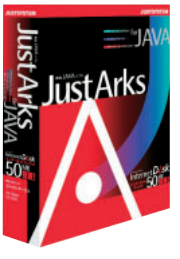
マルチプラットフォーム対応Java製オフィスソフト
Just Arks

URL　http://www.justsystem.co.jp/ark/

発売	株式会社ジャストシステム
TEL	03-5412-3939
価格	1万5000円

ジャストシステムは、すべてJava言語で開発したマルチプラットフォーム対応のオフィスソフト「Just Arks」を発売した。

Just Arksは、文書の論理構造を考えるのに適したアウトラインプロセッサ機能を持ち、HTMLや既存文書を自動的に解析しプレゼンテーションに適したデータを構築してくれる本格的プレゼンテーションソフト「PrezArk」、ビジネス利用を想定して開発され、約160種類の関数をサポートしているマルチシート型表計算ソフトの「CalcArk」、インターネットドキュメント互換を実現するため、いち早くXHTMLに対応したほか、日本語、韓国語、フランス語などを混在できるワードプロセッサ「一太郎 Ark1.1」、プラグインソフト、インターネットディスク使用権50Mバイトで構成されている。標準データ形式にXHTMLを採用している。



Software　　発売日　2001年11月12日

ネットワーク・バックアップ・ソフトウェア
NetVault 6.5 Linux

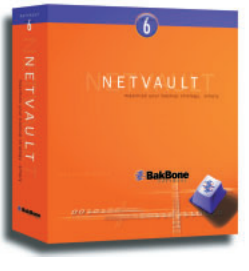
URL　http://www.bakbone.co.jp/

発売	バックボーン・ソフトウェア株式会社
TEL	03-5908-3511
価格	15万円～

バックボーン・ソフトウェアは、OSが混在する環境でのネットワーク・バックアップ、リストアを可能にしたバックアップ・ソフトウェアNetVault 6.5をリリースした。

NetVaultを使うと、複数のOSが混在するネットワーク上に分散するマシンのデータバックアップを一括して管理できる。NetVaultは、それぞれのOSやアプリケーション、マシンに対応したモジュールをインストールするので、拡張性に優れている。

6.5では、障害が発生したシステムをリモートから迅速に復旧するためのディザスタ・リカバリ・オプション、リアルタイムでのファイルバックアップを可能にしたオープン・ファイル・マネージャ・オプション、GUIの日本語化、クラスタリングサポートなどの機能が追加されている。製品は10月10日Solaris（SPARC）用モジュールから順次出荷を開始し、Linux用モジュールが出荷されるのは11月12日から。



日刊アスキー Linux Business Report

OSDLジャパンラボ開設

http://www.linux24.com/

10月24日、LinuxWorld Conference & Demo /Tokyo 2001が行われている新宿NSビルにて、OSDL（Open Source Development Lab）のラボを日本で開設する旨の記者発表会が行われた。

OSDL自体はオープンソースソフトウェアの研究支援機関なので、ビジネスという枠では語れないかもしれない。しかしながら、後述するようにOSDLのスポンサーには日米の大手IT企業が名を連ね、IT発展という観点から、Linuxの基礎研究がいかに重要視されているかを伺わせている。

今回は、横浜に開設されるOSDLジャパンラボの詳細と、OSDLジャパンラボのディレクターに就任する高澤真治氏のお話を紹介したい。

コミュニティを支援するOSDL

OSDLは、米国オレゴン州ポートランドに本拠を置き、エンタープライズ／高性能サーバ向けLinuxプロジェクトや、Linuxベースのオープンソースソフトウェア開発プロジェクトを支援している非営利団体（NPO）だ。ラボのディレクターはTimothy Witham氏。具体的な活動としては、高価で大規模な実験環境を用意できない開発コミュニティに対し、ハードウェア環境の提供をはじめとした各種サポートを行う。活動はオンラインが中心であり、各コミュニティはたとえばネットワーク経由で機材を使うことができる。最近の動きとしては、OSDLに送付したカーネルのパッチやアップグレードを、指定したコンピュータ上でコンパイルしてパフォーマンスを検証する「STP（Scalable Test Platform）」などが開始されている。

これらOSDLで支援されたプロジェクトは、オープンソースの開発モデルに則り、誰でもその成果物を利用できる形で公開される。一企業の利益になるようなプロジェクトの支援は行われない。あくまでもオープンソースのプロジェクト支援の機関なのだ。

運営資金は、**表1**に記した企業からの提供によってまかなわれている。現在2400万ドルの資金を集めているというから、30億円程度の資金が、エンタープライズ向けのLinux技術に投資されているわけだ。NEC、富士通、日立製作所、三菱電機、ミラクル・リナックスといった、日本企業が名を連ねている点にも大いに注目したい。

横浜に2番目のラボを開設

発足して1年近くが経過したOSDLが、2番目に開設するのが横浜のOSDLジャパンラボだ。OSDLは基本的にバーチャルなラボなので、どこにハードウェアがあっても構わないのだが、カーネルなどコアな部分の開発はシステムのリブートなども伴うし、言語、地域的な制限も出てくるため、日本の有志がOSDLのボードメンバーに訴えて開設となった。OSDLジャパンラボは、日本ばかりではなく、アジア・太平洋地域のオープンソースプロジェクトの支援も行う。アジア各国のプロジェクトは、OSDL本部が受け付けたあと、状況によってOSDLジャパンラボを使用する場合もあるという。基本的にジャパンと米国は連携して動くようで、たとえば日本にない機材が米国にある場合は、米国の機材を使用することも可能だ。バーチャルなラボなので、こうしたことも当たり前に行われるのだろう。

OSDLジャパンラボの施設は、神奈川県横浜市保土ヶ谷区の「横浜ビジネスパーク（YBP）」に設立される。YBPは、野村総研やソニーの研究所も入居しているが、OSDLジャパンラボは現在新築中の建物に入居することになっている。床面積は400平方メートル。面積の半分をサーバルームにし、残り4分の1をオペレーションルーム（3室）に、ほかのスペースをミーティングルームとスタッフルームにする予定。

肝心のハードウェアは現在選定中だという。考えられているスペックは、32ビット、64ビットのIAマシンをそれぞれ2Way～8Way構成とし、バリエーションをそろえていく。ただ、64ビットの8Wayはまだまだ高価なので、発表会では不確定要素となっていた。こうした機材の調達方法は、OSDLが欲しいスペックを公開し、各企業に対して分散して購入する手順となる。購入に関しては、各プロジェクトのリーダーとも相談を行う。イメージ的には入札に近い方法といえるだろう。

スタッフは、ディレクターである高澤氏のほか、常駐SEが3名。各プロジェクトには当然コミュニティの人間が参加するわけだから、常駐の3名のみがOSDLジャパンラボで作業を行うわけ

ではない。この3名はインフラ周りや機器関連のスケジュール調整なども行うという。高澤氏は、OSDLジャパンラボにはカーネルに強い人間がいないので、コミュニティに相談してカーネルの専門家を確保したいとしている。

気になる運営資金だが、これは本家米国のOSDLからの支援金のみが唯一の収入源だ。その金額は約数億円。このうち半分をハードウェア購入費用にし、4分の1が家賃、残り4分の1が人件費となる。OSDLのスポンサーに日本企業が多く存在するため、直接OSDLジャパンラボへの支援がありそうな印象を受けるが、OSDLジャパンラボ自体は集金活動は行わない。よって、日本企業は今まで通り、米国のOSDLに寄付を行うという形になる。もちろん営利活動もいっさいしないので、Tシャツの販売や広告掲載などは行われない。

OSDLジャパンラボへのプロジェクトの募集は、現時点ではOSDN本部への英語による申し込みのみだが、11月末を目安に日本語での受け付けも開始する予定だ。現在すでに8つのプロジェクトが計画されており、5つのプロジェクトは発表会で公表された（**表2**）。高澤氏によると、今後4〜5件のプロジェクトを、1〜2年のスパンで運営していきたいそうだ。ただ、プロジェクトによっては比較的簡単なものもあるので、そうしたものも含めると、さらに多くのコミュニティがOSDLジャパンラボを利用することになるかもしれない。

プロジェクトの発動は、まずはリクエストフォームによる応募をし、プロジェクトのリーダーがハード・ソフトの要件をOSDLに伝達する。OSDLは研究結果の公開が前提なので、プロジェクトは機密保持契約を結ばないことを了承しなければならない。こうしてできた成果物は、途中結果がレポートされるほか、Linuxディストリビューターなどに転送され、次世代のLinuxをより高度なものへと進化させていくという流れだ。

●

OSDLジャパンラボの設立自体が、ビジネスに即直結するとは考えにくいが、その研究結果が万人に公開されるなど、業界全体が得る利益は限りなく大きい。なお、OSDLジャパンラボでは、OSDLの準会員であるOSDLアソシエーツの日本語メーリングリスト開設も予定しているそうなので、興味のある方は今後の動向に注目だ。

米Caldera International
米Computer Associtates International
米Covalent Technologies
米Dell Computer
富士通
日立製作所
米Hewlett-Packard
米IBM
米Intel
米Linuxcare
米LynuxWorks
ミラクル・リナックス
三菱電機
NEC
米Red Hat
米SGI
ドイツSuSE Linux
米Turbolinux
米VA Linux Systems

表1　OSDLのスポンサー企業

IPv6
SCore
Namazu
PostgreSQL
Linux Super Page

表2　OSDLジャパンラボで計画中のプロジェクト

Column

インタビュー
OSDLジャパンラボディレクター
高澤真治氏

OSDLジャパンラボの統括責任者として、日本Linux協会副会長／日本Linux協会副会長／LPIジャパン理事／Linux World Japanアドバイザリーボードメンバーの高澤真治氏が就任した。同氏は元大手コンピュータメーカーに勤務していたが、OSDLジャパンラボ設立により退職。オープンソースの発展に寄与することとなった。

――企業が非営利団体のOSDLのサポートをするのはなぜか？

高沢氏：Linuxの世界では貢献することで技術が進化して、その上に応用が見えていくということだろうと思う。いずれにせよ、成果物はオープンになって、企業もたとえばApacheやSambaでビジネスを行っている。それを還元するという意味もある。日本の企業も長い目でビジネスプランを考えているのではないか。Linuxは基礎技術として注目されているから、コントリビュートすることに日本企業は気を遣っている。たとえばSCoreプロジェクトもNEC、富士通がスポンサーだが、動作するのはNECや富士通のUNIXではなくLinuxだ。今後は、巡り巡ってお金が回って来るというスタイルに変わってくると思う。

――今後、ハードウェアの追加などは考えているのか？

高沢氏：リクエストが出た時点で、よほどの金額でない限り買う。全体の予算の半分がハードウェア費用だが、初期費用はその中の半分。残りはバッファにしている。だがたとえば5億円の買い物は無理なので、本部との相談になる。本部がやるべきプロジェクトだと判断すれば、本部が買うかジャパンが買うかするだろう。

――共同研究や補助金など、政府機関との関係は？

高沢氏：大学関係の方々は、もともと先端の研究もされているので現在も入っていただいている。しかしそれ以外では今のところコンタクトは取っていない。プロジェクトを誘致していただけるのであれば、会いに行きたい。ただ、補助金などお金が流れるというのは難しいだろう。

LinuxWorld Conference & Demo/Tokyo 2001

October 24-25, Tokyo

秋のLinuxイベント「LinuxWorld Conference & Demo/Tokyo 2001」が、10月24日～25日の両日、新宿NSビルのNSイベントホールで開催された。デモ会場で発表された製品を中心に速報レポートをお伝えする。

10月24日と25日に新宿NSビルのNSイベントホールで「LinuxWorld Conference & Demo/Tokyo 2001」が開催された。

5月末（Expo）と10月末（C&D）の年2回開催が定着してきたLinuxWorldだが、今回は会場を東京・有明から交通の便が良い新宿に移し、都心で働くビジネスマンが来場しやくなった。来場者数も昨年より増えて、2日間合計で2万名弱の方々が訪れた。

基調講演はオラクルの新宅正明氏

最初の基調講演は、日本オラクル代表取締役社長・新宅正明氏によって、“「e-Japan戦略」を実現するオラクルの役割 ～Linuxの現状と今後の展開～”というテーマで行われた。

IT景気が落ち込んだ米国のエンジニアの動向を挙げ、中国、インド、ベトナムの技術者たちが、米国企業から受注がなくなり、日本市場にアプローチしてきているとし、すでに成長しているモバイルや今後発展するブロードバンド分野で、日本の技術が羽ばたくチャンスであると述べた。

そして、e-Japan構想によって国や自治体のシステムとして、オープンアークテクチャの採用が進むことで、大手企業の閉じたシステムではなく、中堅・零細企業にもチャンスが広がるとの展望を示した。

コンファレンス＆セミナー

基調講演終了後は、同会場を3つに区切り、「Business Track」「Technology Track」「1 Day Track」のそれぞれで、セミナー／パネルディスカッションが開催され、多くの聴講者を集めていた。

「1 Day Track」では、24日に「Oracle Linux Summit 2001 Autumn」が行われ、LinuxWorldに合わせるように発売された、Oracle9i for LinuxとMIRACLE LINUX V2.0によって実現するシステムを紹介した。

また、25日には「Debian Conference in Tokyo, Japan 2001」が開催され、Debian GNU/Linux独自のパッケージ管理dpkg／aptツールやDebianのWebページに関する解説などが行われた。

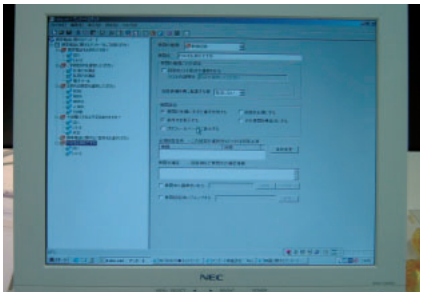
少なかったディストリビューターの参加

展示会場では、約40社（団体）が出展していた。NECからは、低価格なItaniumプロセッサ搭載の新製品として、CPUを4基搭載可能なサーバ「Express5800/1040Xa」と、2基搭載可能なワークステーション「Express5800/59Wa」が発表された。

また、目立っていたのはアプライアンスサーバで、日立「HA8000-ieシリーズ」、ノーザンライツコンピュータ

パネルディスカッション「Linuxのホンネ ～Linux B2Bシステムの現場から～」では、LinuxでB2Bを構築・運用している4名の方々を招いて、Linuxを選んだ理由や実際の運用について話し合われた。

NTTコムウェアが参考出品した「Intelligent Memory Server」は、32Gバイトのメモリを内蔵し、データ領域をディスクの代わりにメモリで処理することで高速データ処理を実現する。



NECのインターネットマーケティングソフト「ACTIVECR」は、質問と回答を入力していくだけで簡単にWebアンケートを作成することができ、集計やダイレクトメールの送信なども行える。

韓国のPalmPalm Technologyは、同社の組み込み向けStrongARMベースのプロセッサに最適化されたLinuxディストリビューション「Tynux」と、開発キット「Tynux Box II」を展示していた。

「NL Server 1500」、VA Linux Systems「VA Linux 1222／2252」、ランス「Lx-Server」、AIMCOM「InetBase IT Server」といった製品が展示されていた。

組み込みLinux関連では、開発ツールやキットとして提供される製品として、パームパームテクノロジーの「Tynux Box II」やメガソリューションから米REDSonicの「REDICE-Linux」、レーザーファイブの「L-Card＋」「L-Board」などがあった。

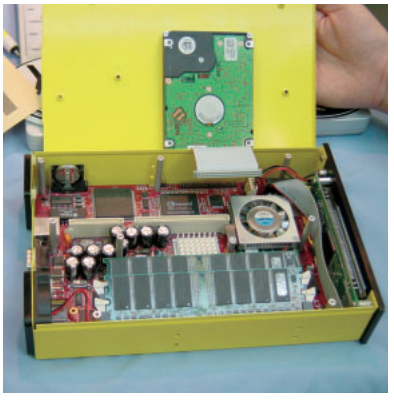
Linuxに関するITエンジニアの教育やセミナーも盛んになってきており、びぎねっとや、ソリッドスキーム、リナックスアカデミーがブースを出していたほか、オープンなLinux技術者レベル認定試験を行っているLPI-Japanでは、特設会場にてLPI認定割引試験を開催していた。

今回出展していた大手Linuxディストリビューターは、ミラクル・リナックスとレーザーファイブの2社だけで、レッドハットやターボリナックス ジャパン、デジタルファクトリなどが出展していないこともあって、少しさびしかった。ミラクル・リナックスは、発売されたばかりのMIRACLE LINUX V2.0を、レーザーファイブは、8月発売のLASER5 Linux 7.1のデモを行っていた。

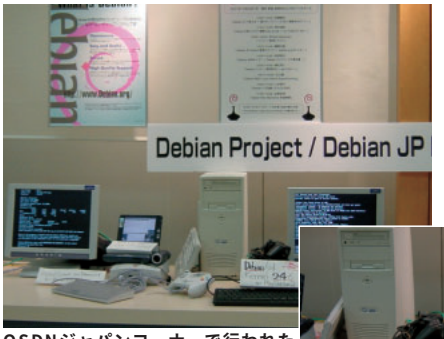
LinuxWorld会場内で開催された「Debian Conference in Tokyo, Japan 2001」で行われたセミナー風景。

ランスの2UラックマウントタイプのNASアプライアンスサーバ「Lx-Server」は、内蔵する「KEGON」という独立したワンボードPCがシステムの状況を監視し、前面のLCDモニタに表示、トラブル時の自動制御を行う。

日立ソフトのネットワークアプライアンス「ILIOS IS200」は、195（W）×148（D）×80（H）mm、2.3kgと小型ながら3.5インチHDD内臓で、100BASE-Tポートを2ポート備え、Red Hat 7.1も動作確認済み。



アミュレットは、米Terra Soft SolutionsのPowerPC対応ディストリビューション「Yellow Dog Linux 2.1」を販売開始した。CD-ROMドライブサイズPowerPCマシン「briQ」を4ノードのクラスタ構成で接続し、レンダリングを行っていた。



OSDNジャパンコーナーで行われたDebian Project／Debian JP Projectの展示では、カーネル2.4ベースのDebian GNU/Linuxを、PlayStation 2やDreamcastで動作させているデモが行われていた。

Products

カードサイズの組み込み用超小型Linuxサーバ

FutureNet MA-100

Linuxが備えているネットワークサーバ機能を利用すると、PCに限らずさまざまな機器同士を結びつけて連携したシステムを構築できる。FutureNet MA-100はコンパクトに凝縮したサイズに、RS-232CとLANポートを装備し、各種ネットワークサービスを内蔵した組み込み用Linux製品だ。

製品名	FutureNet MA-100
価格	3万4800円（基板単体100個購入時）
問い合わせ先	センチュリー・システムズ株式会社 TEL 0422-37-8911 http://www.centurysys.co.jp/

Linuxを使用したマイクロアプライアンスサーバやブロードバンドルータを開発しているセンチュリー・システムズから、カードサイズの超小型Linuxアプライアンスサーバ「FutureNet MA-100」（以下MA-100）が発売されている。

MA-100は、基板サイズが54mm×80mmとクレジットカード並の大きさで、その上にCPU、メモリ、各種インターフェイスを搭載した組み込み用カードだ。製品はOEM向けなので、1ロット100台以上で販売される。**写真1**のようなケースに入れたものと、ケースを除いた基板のみの形態が用意されている。

CPUは、RISCプロセッサコアに周辺機能をワンチップに組み合わせたモトローラのPowerPC 855T 50MHzを採用している。4Mバイトのフラッシュ ROMにLinuxが書き込まれており、電源を入れると起動する。RAMは16Mバイトで、そのうち4Mバイトは RAMディスクとして利用している。

外部インターフェイスには、10/100BASE-T対応のネットワークインターフェイスを1ポート、RS-232Cを1ポート備えている。基板の裏側には、CFカードスロットがあり、コンパクトフラッシュかIBMのMicrodriveを装着して使用できる。

多くのネットワークサーバ機能を標準装備

MA-100には電源スイッチが存在しないため、ACアダプタを接続するとすぐにLinuxが起動する。PCとの接続は、LANまたは、クロスケーブルを使ってRS-232Cポートで行う。LANの場合、本機のIPアドレスの初期値は、192.168.253.254になっているので、PCを同一セグメントの別のア

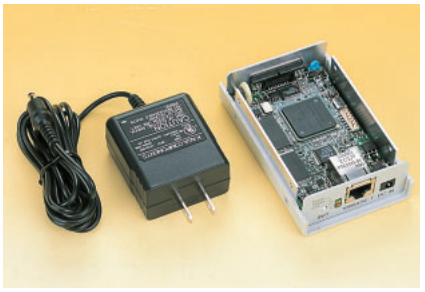


写真1　ケースとACアダプタ
小型ケースはプラスチック製で、たばこの箱より少し大きいくらいのサイズだ。ファンレスなので無音で使用可能。

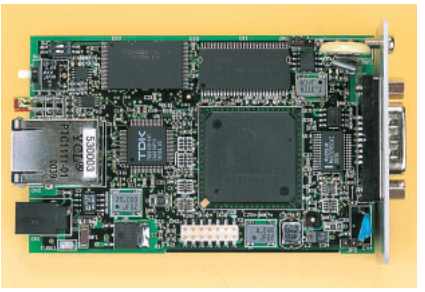


写真2　マザーボードのCPU装着面
モトローラのCPU PowerPC 855T、メモリ、各種インターフェイス、RTC用バッテリなどが装着されている。

写真3　マザーボードのCFカード装着面
コンパクトフラッシュではなく、1Gバイトの Microdriveを装着すれば大容量のデータを格納できる。

ドレスに設定してつなぐ。RS-232Cの場合は9600bpsで接続する。

インストールされているネットワークサーバ機能は、**表1**のようになっている。ストレージオプションを装着して、ファイルシステムを作成し、アーカイブされているソフトウェアを展開することで、メールサーバ、DNSサーバ、FTPサーバ、PPPサーバが使用可能になる。標準のフラッシュROMに、BashやApacheといった数百Kバイトの大きなサイズのプログラムとともに、圧縮されてはいるものの追加のプログラムも入っているとは驚きである。

各サーバ機能の設定を変更するために、PC上のターミナルエミュレータ（telnet）プログラムからrootでログインして、エディタ（aee）で設定ファイルを直接編集することができる。

たとえば、DHCPサーバでは/etc/dhcpd.confを、メールサーバでは/etc/sendmail.cfを編集することになる。そして、HTMLファイルを、/home/httpd/html/ディレクトリ以下に置くことで、簡単にWebサーバとして利用することができる。

また、ipchainsを用いて、パケットフィルタやIPマスカレード機能を利用することも可能だ。

MA-100に搭載されているOSは、PowerPC用のLinuxディストリビューションであるLinuxPPC Release 5のサブセットである。カーネルは2.2.13で、本体上のフラッシュROMには、容量が少ないため必要最小限のプログラムしかインストールされていない。ユーザーがプログラムを追加する場合には、PowerPC用のクロスコンパイラ環境を構築するか、PowerPCで動作するLinuxシステム上のgccで開発して、MA-100にFTPなどで転送すればよい。

用途としては、RS-232Cで接続した計測器などを通じてデータの収集を行い、LAN経由でPCからアクセスすることや、遠隔地に置いたMA-100にダイヤルアップで接続して、LANにつながっている機器の監視を行うことが考えられる。

/etc以下の設定ファイルの変更には、Linux（UNIX）のネットワーク設定の知識が必要なので、一般の人がMA-100は使いこなすのはむずかしいかもしれない。しかし、多くの機能が盛り込まれており、これらを応用することでさまざまな用途での利用が可能だろう。

	標準状態	CFカード／Microdrive使用時
telnetサーバ	セットアップ済み	セットアップ済み
Webサーバ（Apache）	セットアップ済み	セットアップ済み
DHCPサーバ（dhcpd）	セットアップ済み	セットアップ済み
DHCPクライアント	セットアップ済み	セットアップ済み
メールサーバ（sendmail）	アーカイブ状態	セットアップ済み
DNSサーバ（bind 8.2.2）	アーカイブ状態	セットアップ済み
FTPサーバ（wu-ftpd）	アーカイブ状態	インストール済み
PPP（pppd）	アーカイブ状態	インストール済み

表1　動作可能なネットワークサーバ機能

CPU	PowerPC 855T 50MHz
ROM（Flash）	4Mバイト（最大8Mバイト）
RAM	16Mバイト（最大32Mバイト）
シリアルポート	RS-232C×1ポート
イーサネット	10／100BASE-T×1ポート
拡張インターフェース	CFカードスロット（Type I／Type II）
LED	Power、ステータス、Link（LAN）の計3つ
OS	Linux（カーネル2.2.13、glibc 2.1.2）
ストレージオプション（システム、データ用）	コンパクトフラッシュカード Microdrive
基板サイズ	54mm×80mm（突起物を除く）
ケース外形寸法（mm）	61（W）×99（D）×28（H）
電源	DC 5V（ACアダプタ使用）

表2　FutureNet MA-100の主な仕様

Red Hat Linuxにも対応したネットワーク対応データベースのサーバ版

ファイルメーカーServer 5.5

WindowsやMacからGUIの操作で手軽に使えるデータベースソフトとして評価の高いファイルメーカーPro 5.5を使って、複数のユーザーでデータベースを共有することを可能にするネットワークサーバ専用ソフトウェア。

製品名	ファイルメーカーServer 5.5
価格	12万9000円
問い合わせ先	ファイルメーカー株式会社 TEL 03-5977-7256 http://www.filemaker.co.jp/

Macintosh用のデータベースソフトとして有名な「FileMaker」だが、現在ではGUIでの操作が簡単に行えるリレーショナルデータベースとして、Windows用も発売されている。

7月に発売された最新版の「ファイルメーカーPro 5.5」では、Windows 2000やMac OS Xネイティブへの対応や、ODBCを介して他のデータベースとデータ交換が図れるようになり、使い勝手を向上させている。

ファイルメーカーPro 5.5自体でも、LANで接続したPC同士でデータベースの共有を行うことは可能だが、多数のユーザーが同時にアクセスするのには適していない。そのため、専用サーバにインストールして、高速に効率的にワークグループ内で共有データベースを利用できるようにした製品が「ファイルメーカーServer 5.5」である。

LDAPディレクトリサービスやWindows認証が利用可能

ファイルメーカーServer 5.5（以下Server）は、Windows NT／2000、Mac OS／OS X、Red Hat Linuxといった幅広いプラットフォームで動作する。

Serverの管理は、ファイルメーカーPro 5.5から行うことができる。ゲストユーザーの接続状況などを監視できるほか、接続の強制解除やゲストへのメッセージの送信などが可能だ。

ネットワーク上で別のセグメントのサーバを指定する際に便利なように、LDAP対応のディレクトリサービスに登録することができる。データベースにアクセスできるユーザーを、Windows認証を利用してServerの所属するドメインのユーザーに限定することができる。

また、ファイルメーカーPro 5.5用に作成したプラグインファイルを、Serverのプラグインアップデート用フォルダに置いておくだけで、クライアントが接続した際に自動的にダウンロードする機能を備えている。

インストールと設定

ファイルメーカーServer 5.5 Linux版（以下Server Linux版）は、RPMファイルで提供されるので、インストールは他のLinuxプログラムと同様に、rpmコマンドかGnoRPMで行う。

インストールできたら、まず製品パッケージに含まれているインストールキーを利用して、レジストレーションの作業を行う。

プログラムはデーモンとして登録されるので、

```
# /etc/rc.d/init.d/fmreserved start
```

として起動する。Linuxシステムの起動時にServer Linux版をスタートさせるには、サービス設定プログラムであるntsysvを利用して、httpdなどと同様にチェックするだけだ。

各種環境設定は、/etc/fmserver.confファイルを直接編集する。このファイルは、日本語でコメントが丁寧に書かれているので、それを見ながら書き直せば設定を間違えることは少ないだろう。

設定できる項目を簡単に紹介すると、同時アクセス可能なゲスト数、同時共有可能なファイル数、キャッシュサイズ、共有ファイルディレクトリ名の指定、リモート管理の許可／パスワード、ログファイル名／最大サイズ、LDAPの設定などである。

なお、同時接続可能なゲスト数は250人まで、同時に開けるファイル数は125ファイルまでと、多数のWindowsやMacintoshクライアントをサポートすることが可能だ。

共有データベースの転送はFTPやSambaを利用

ファイルメーカーServer 5.5は、クライアントとしての機能は持っていないので、データベースの設計は、ファイルメーカーPro 5.5を使用して作成しておく。ウィザードで対話式に指定したりテンプレートを利用す

ることで、楽に設計できるだろう。属性の変更やサイズの調整などの設計変更もGUIを使って簡単に行える。

さて、完成したデータベースファイルをServer Linux版で共有するには、MOやCD-Rなどのメディアか、FTPやSambaを使用してファイル転送する必要がある。先ほどのfmserver.confでは、共有ファイルディレクトリの初期値として/var/fmserverが設定されているので、その下にデータベースファイルをコピーする。

ファイルメーカーPro 5.5のファイルメニューから、［共有ファイルを開く］を選ぶと**画面1**のように、共有ファイルのダイアログが表示される。

ここで、「FMServer」と表示されているのが、Server Linux版のインストールされたマシンを表している。画面1の下にある「ローカルホスト」を選ぶとブロードキャストによって、「ディレクトリサービス」ならばLDAPによって、「ホスト指定」だとIPアドレスかドメイン名の指定によってサーバを探して表示する。

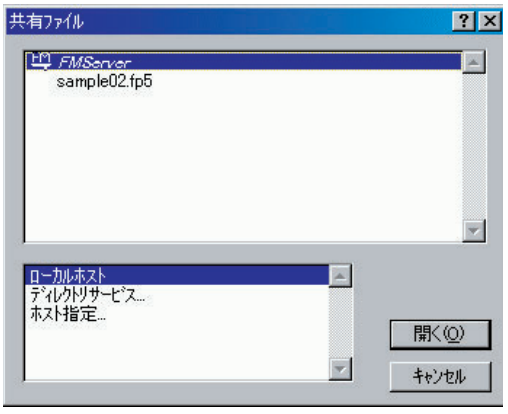
「FMServer」をダブルクリックすると、サーバリモート管理プラグインがダウンロードされ、ファイルメーカーPro 5.5のウィンドウ内に**画面2**のような「サーバリモート管理」が表示される。ここで、データベースファイル（ここではsample02.fp5）を選んで［ファイルを開く］をクリックすることで、クライアントからデータベースの共有が可能になる。あとは、もう一度先ほどの画面1で、sample02.fp5のように表示されているデータベースファイルを開けば、サーバ上のデータベースを使って起動する（**画面3**）。

なお、リモート管理では、サーバ上の負荷をリアルタイムに監視することができるほか、ユーザーのアクセスやサーバの状況を記録したログファイルの閲覧も可能だ（**画面4**）。

Serverを利用することで、インデックス処理の高速化やキャッシュファイルの利用、マルチスレッドのサポートによって複数作業の並行処理も効率よく行える。

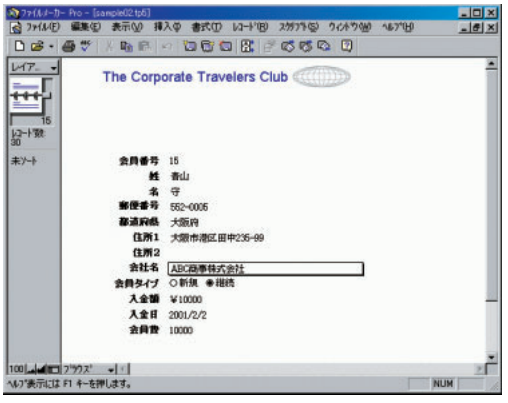
WindowsとMacintoshのどちらでも同じ操作方法でデータベースを使えるようするには、Webブラウザを用いることが多いが、JavaScriptなどを駆使しても専用ソフトの使い勝手には及ばないところがある。

その点、ファイルメーカーPro 5.5ならば、アプリケーションの開発が容易なうえ、どちらのプラットフォームでも同じデータベースファイルが利用可能だ。Serverを併用すれば、複数ユーザー利用時のパフォーマンスの低下も防ぐことができる。

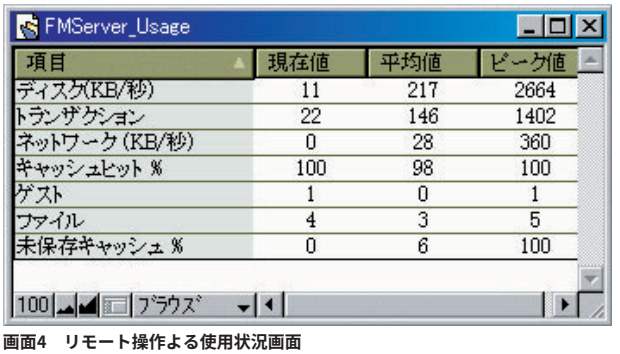


画面1　共有ファイルを開くダイアログ
共有可能なファイルを表示する。ブロードキャストやLDAP、ホスト名によって指定したファイルメーカーServerに接続する。

画面2　サーバリモート管理画面
ファイルメーカーPro 5.5上からファイルメーカーServer 5.5をリモート管理できる。共有DBの指定を行う。



画面3　サンプルを利用して共有したDBの表示
ファイルメーカーPro 5.5付属のサンプルデータベースを、ファイルメーカーServerの共有ファイルとして読み込んだ。



画面4　リモート操作よる使用状況画面
使用状況ボタンを押すと、ファイルメーカーServerのパフォーマンス状態が表示される。

Distribution

新着ディストリビューション

Red Hat Linux 7.2

ついにジャーナリングファイルシステムを採用。他の有力ディストリビューターが次々とジャーナリングファイルシステムを採用していくなか、Red Hat Linuxはかたくなにext2を採用し続けていた。今回、Red Hat Linuxが満を持して採用したのは、ポピュラーとはいえないext3だった。Red Hat Linuxがext3を採用した経緯に迫ってみる。

SuSE Linux 7.3

完成度はトップクラス。ドイツ産ディストリビューションのSuSE Linuxは、新しいパッケージを採用するタイミングでは、業界1、2を争う早さだ。そして、新しいパッケージによってもたらされる新機能を、設定ツールを収録することで誰でも使えるようにしている。新機能と設定ツールの充実度、これがSuSE Linuxの完成度が高いと言われる所以である。

Mandrake Linux 8.1

お洒落な風体でも中身は最新仕様。Mandrake Linuxのインストーラやデスクトップのデザインは、フランス産のディストリビューションらしくお洒落さと遊び心が満載だ。もちろん、外見だけではなく収録されるコンポーネントの新しさでもMandrake Linuxは十分に目立っている。

MIRACLE LINUX Standard Edition Version2.0

エンタープライズ仕様。オラクルデータベースのプラットホームを自認するMIRACLE LINUXは、カーネル2.4のパフォーマンスをフルに引き出すためにチューニングが施されている。バイナリはPentium Pro以上に最適化するなど完全サーバ仕様である。

LASER5 Secure Server6.9

クラッカーお断り。インターネットサーバに特化したLASER5 Secure Server6.9は、コンパイラやライブラリのレベルから、サーバプログラムを構築している。バッファオーバーフローの心配がない完全セキュア仕様だ。

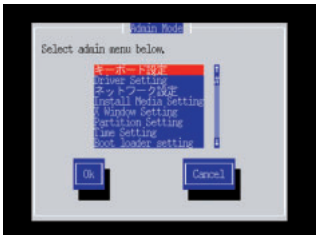
Distribution ▶▶▶

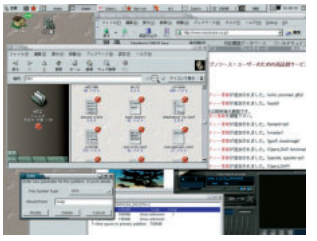
「ARMA aka Omoikane GNU/Linux 2.0」12月中旬発売予定

オモイカネは、パッケージシステムとして定評のあるDebian GNU/Linux互換である「Omoikane GNU/Linux 1.1」の次バージョンを12月中旬に発売する予定だ。今バージョンからシリーズ名がARMAになり、商品名称は「ARMA aka Omoikane GNU/Linux 2.0」である。

前バージョンまでは、ダウンロードでのみ提供されてきたが、ARMAは商用・非商用にこだわらない視点から開発され、デスクトップおよび小さいサーバ向けの製品としてショップでパッケージ販売される。ARMA aka Omoikane GNU/Linux 2.0では、リコーフォントなどの商用ソフトウェアがバンドルされ、標準デスクトップ環境には、GNOME 1.4＋Nautilusが採用される予定である。ほかにも、GUIでのインストール、統一された管理ツール（画面左）でインストール後のシステムメンテナンスが可能となるなど期待できる内容である。

オモイカネ（http://www.omoikane.co.jp/）





「OpenLinux 64 Release 3.1」発売開始

カルデラは10月15日にIA-64対応「OpenLinux 64 Release 3.1」の発売を開始した。カーネル2.4.5、XFree86 4.1.0、KDE 2.1.1、gcc 3.0などを基本システムに採用している。

インストール時に必要のないポートを閉じることができるほか、不正アクセスをチェックするソフトが統合されるなどのセキュリティ対策がとられている。製品マニュアルのHTMLドキュメントをローカルまたはリモートから参照できるDocview、サーバ管理ツールWebmin、64ビット対応JavaやRAID設定ツールなどが搭載されている。

現在ベータ版が公開されている「OpenLinux 3.1.1」は12月より発売が開始される。基本システムには、カーネル2.4.9（または2.4.10）、XFree86 4.1、glibc 2.2.4、gcc 2.95、KDE 2.2.1などが採用される予定だ。

カルデラ（http://www.jp.caldera.com/）

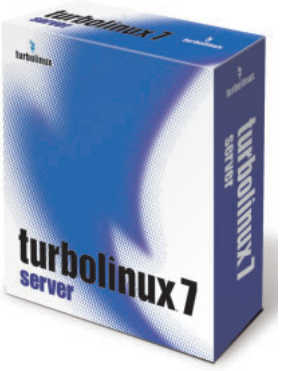
「Turbolinux 7 Server」12月7日発売予定

Turbolinux 7 Serverの基本システムには、カーネル2.4.9、glibc 2.2.4、XFree86 4.1.0、gcc 2.95.3などを採用し、ext3、ReiserFSはもとより、IBM JFS1.0にも対応する。

最大32個のCPUを搭載するコンピュータシステムをサポートし、ジョブを並列処理させることで大幅なパフォーマンスの向上を実現する。

ユーザーフレンドリーなWebベース設定ツール「Webmin」を採用するほか、複数のドライブとパーティションを仮想的に1つのドライブとして扱うLVMや、複数のハードディスクを1台のディスクのように使用して信頼性や処理速度を高めるRAIDの設定で、グラフィカルな独自ツールを採用するなど、より便利にサーバ構築やシステム管理ができるようになる。

商用ソフトウェアには、仮想的に複数のテープライブラリやスロット、ドライブを作成するバーチャルライブラリ機能により、テープ装置がなくてもバックアップが可能な「NetVault 6.5 Turbolinux Edition」、「リコーTrueTypeフォント」がバンドルされる予定だ。本体価格は3万9800円。



ターボリナックス ジャパン（http://www.turbolinux.co.jp/）

Progenyが独自パッケージの開発を中止

Debian GNU/Linuxの独自バージョンの開発を進めていた米Progeny Linux Systemsは、10月15日をもって独自開発を中止し、コンサルティングサービスに専念することになった。

Progenyは、本家Debianのアップデート間隔が、1年から1年半と他のディストリビューションに比べて遅いことにしびれをきらして独自開発を始めた。Progenyは、ほかのディストリビューションが持っている機能を追加する形でインストーラ、アップデートソフト、GUIを強化した。しかし、2度目のバージョンアップ開発時に、Debianプロジェクトの方向性と相反する問題に直面したため、自社開発を断念してDebianプロジェクトに戻ることになった。Progeny製ディストリビューションの大部分は、Debianプロジェクトに引き継がれた。

米Progeny Linux Systems（http://www.progeny.com/）

Debianプロジェクト（http://www.debian.org/）

Red Hat Linux 7.2

Linuxの代表的なディストリビューションであるRed Hat Linux（以下、Red Hat）の最新版、Red Hat Linux 7.2が10月31日に発売された。

Red Hat 7.2には、ProfessionalとDeluxeの2種類のパッケージが用意されている（**表1**）。

Red Hat 7.2は、基本コンポーネントにカーネル2.4.7、glibc2.2.4、XFree86 4.1.0を、デスクトップ環境にGNOME 1.4.0とKDE 2.2を採用している（**表2**）。

標準ファイルシステムext3

Red Hatは新しい機能を積極的に取り入れることが特徴だが、ことファイルシステムに関しては、ext2を標準ファイルシステムとして採用し続けてきた。しかし、今回満を持して、標準ファイルシステムとして、ext3を採用してきた（**画面1**）。

製品名　Red Hat Linux 7.2
価格　1万2800円（Deluxe）
4万2800円（Professional）
問い合わせ先　レッドハット株式会社
03-3257-0411
http://www.jp.redhat.com/

ext3は、システムの変更を逐一記録するジャーナリング機能を備えており、システムが不正シャットダウンされた際でも、すぐにシステムを復旧できる。

さて、ext3以外にLinuxで使えるジャーナリングファイルシステムReiserFS、JFS、XFSは、Bツリーという方式でシステム内にあるファイルなどの情報を管理している。このため、大量のファイルが存在するシステムでは高いパフォーマンスを発揮するのだが、ext3はその機能を備えていない。

にも関わらず、Red Hatがext3を採用したのは、多くのシステムではext2が使われているので、ext2との互換性や、ext2からの移行のしやすさを考慮した結果だろう。事実、Red Hat 7.2のインストーラは、ファイルシステムをext2からext3に変換しつつ、古いバージョンのシステムをアップグレードできるように作られている。

標準ブートローダにGRUBを採用

これまで標準のブートローダだったLILOはGRUBに変更された。

LILOがハードディスクの絶対位置で記憶したカーネルにアクセスするのに対し、GRUBはファイルとしてのカーネルにアクセスする。

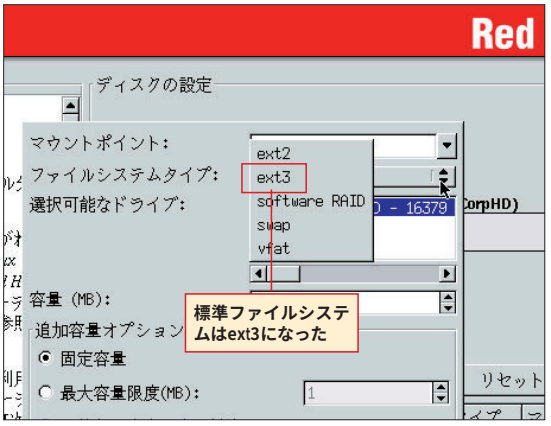
このため、LILOではディスク内でのカーネルの位置が変更された場合にLILOの再実行を忘れると、そのシステムではLinuxが起動しないという状況に陥る。その点、ファイルシステムを理解するGRUBは、カーネルさえディスクの中に残っていれば、システムを起動できるので、カーネルを頻繁に入れ替える上級者にはもちろん、起動のトラブルに柔軟に対応できるという意

	Deluxe	Professional
価格	1万2800円	4万2800円
サポート	Web 30日間 Red Hat Network 90日間	Web 60日間 Red Hat Network 180日間 電話 30日
マニュアル	インストレーションガイド 入門ガイド	インストレーションガイド 入門ガイド カスタマイズガイド
CD-ROM	インストール用CD 2枚 ソースCD 2枚 ワークステーション商用ソフトCD 2枚 ユーティリティCD 1枚 LoKi Game CD 1枚	インストール用CD 2枚 ソースCD 2枚 ワークステーション商用ソフトCD 2枚 ユーティリティCD 1枚 System Administrator CD 1枚 サーバ向け商用アプリケーションCD 1枚

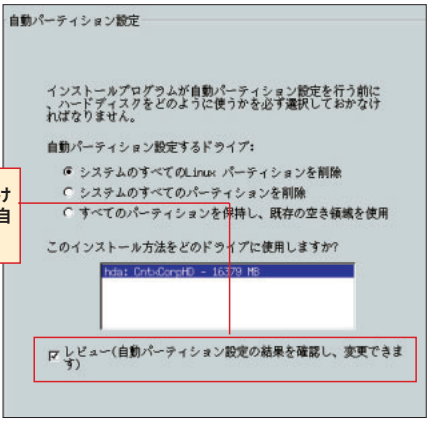
表1　Red Hat Linux 7.2のDeluxeとProfessionalの製品構成

コンポーネント	7.2	7.1
カーネル	2.4.7	2.4.2
glibc	2.2.4	2.2.2
XFree86	4.1.0	4.0.3
GNOME	1.4.0	1.2.4
KDE	2.2	2.1.1
Samba	2.2.1a	2.0.7
OpenSSH	2.9p2	2.5.2p2
PostgreSQL	7.1.3	7.0.3

表2　Red Hat Linux 7.2と7.1に収録されるコンポーネントの違い
GNOME、Samba、OpenSSHのバージョンが大きく上がっている。



画面1　パーティション作成画面
Red Hat Linux 7.2では、デフォルトファイルシステムが、旧来のext2にジャーナリング機能を加えたext3に変更された。



ここをチェックしておけば、パーティションの自動作成案を確認できる

画面2　パーティション設定画面
Red Hat Linux 7.2では、インストールするマシンのパーティション状況に合わせ、Linux用パーティションを自動作成できる。ユーザーは、インストーラのパーティション作成案を別の画面で編集できる。

味で、初心者にも嬉しい変更だ。

柔軟なパーティション作成

Red Hat 7.1以前では、インストール中、1024シリンダ以降にパーティションを作成する場合GUIのDiskDruidではうまくいかず、コマンドで操作するfdiskを使う必要があったが、Red Hat 7.2に付属のDiskDruidは、1024シリンダ以降にもパーティションを作成できるように改良された。

また、自動でLinux用パーティションを作成するオートパーティショニングモードでは、Windowsがすでにインストールされている／いないなど、マシンの状態に合ったモードを選べるようになった（**画面2**）。

Linux用パーティションを自動で作成する際も、インストーラが提示するのパーティション分割案をユーザーが変更することも可能だ。

すべてはデスクトップからスタート

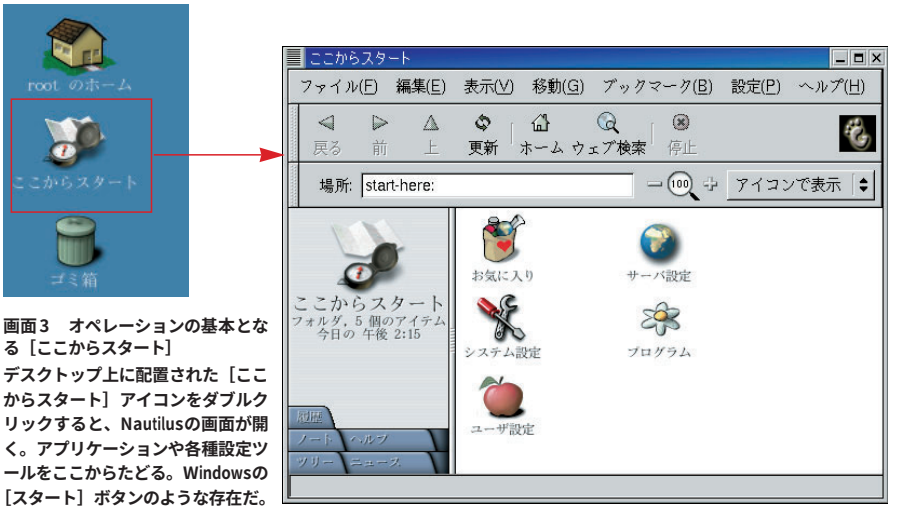
Red Hat 7.2では、GNOME 1.4を標準のデスクトップ環境としたことで、Nautilusを使った統合的なGUI環境を使えるようになった。

Red Hat 7.2のデスクトップに配置された［ここからスタート］というアイコンをダブルクリックすると、Nautilusが起動する（**画面3**）。この画面には、ジャンルごとに分類された設定ツールやアプリケーションが表示されているので、ユーザーは、この画面を起点にしてアプリケーションの起動やシステム管理などを、すべてNautilus内で操作できるのだ。

Red Hat Network

Red Hat Networkは、セキュリティアップデートの情報配信や、RPMパッケージのアップデートインストールなどを総合的に提供するサービスだ。

このサービスを利用するには、Red Hat 7.2の製品を購入するか、サービス用のライセンスを購入する必要がある。



画面3　オペレーションの基本となる［ここからスタート］
デスクトップ上に配置された［ここからスタート］アイコンをダブルクリックすると、Nautilusの画面が開く。アプリケーションや各種設定ツールをここからたどる。Windowsの［スタート］ボタンのような存在だ。

このサービスを利用することで、Linuxのシステム構成に詳しくないユーザーでも、システムのメンテナンスとして必要な作業を簡単に実行できるようになるだろう。

これまで日本国内向けのRed Hat Networkサービスは提供されていなかったが、ライセンスを持つユーザーは、今回から日本国内でも同サービスを受けられるようになった。ただ、セキュリティアップデートなどの情報は英語で配信されているので、国内向けのサービスとしてはまだ発展途上にあるようだ。

●

Red Hat Linuxは、マイナーバージョンアップを重ねて安定期に入るとともに、新しいサービスや機能を、おだやかに取り入れ始めた。

SuSE Linux 7.3

ドイツ産ディストリビューションSuSE Linuxの最新バージョン、SuSE Linux 7.3が10月13日にリリースされた。

SuSE Linuxは、最新に近いバージョンのカーネルやXFree86などを採用し、それら新しいコンポーネントを使いこなすための設定ツールをバンドルして、機能面と使いやすさのバランスがとられている。

今回リリースされたSuSE Linux 7.3では、基本コンポーネントにカーネル2.4.10、glibc2.2.4、XFree86 4.1.0が採用されている。また、デスクトップ環境には、KDE 2.2.1とGNOME 1.4.0が採用されている。

SuSE Linuxには、ドイツ国内向けのパッケージのほかに、各国語に対応するInternational-1と2が用意されている。今回は、日本語環境を利用できるInternational-2を紹介する。

先進機能をYaST2で使う

SuSE Linuxの設定・管理ツールであるYaST2は、SuSE Linuxの大きな特徴で、その完成度は非常に高い。

パッケージ管理、ネットワークやサーバの設定、ディスク管理などの機能はモジュール化してYaST2に統合されているため、ユーザーインターフェイスに統一感があり使いやすい。

たとえば、SuSE Linux 7.3で新たに追加されたLVMの設定機能を見てみよう。LVMは、複数のパーティションを1つの仮想ドライブにまとめる機能だ。まとめるパーティションは複数のドライブにまたがっていても構わないし、あとからパーティションを追加して、「ディスクの容量を動的に増やす」ような使い方もできる。

さて、LVMはカーネル2.4の標準機能なので、最近のディストリビューションであれば、技術的にはLVMを利用できる。しかし、LVMを使うためにはいくつかのコマンドが必要で、LVMの概念をイメージできないユーザーがLVMを使うのはかなり難しい。

その点、SuSE Linux 7.3で新たに追加されたYaST2のLVM管理機能を使えば、パーティションと仮想ドライブの関係を見ながら、直感的な操作で仮想ドライブを作成できる（**画面1**）。

このあたりは、先進機能とその機能を使うための設定ツールが用意されるという、SuSE Linuxのバランスの良さを示す一例となっている。

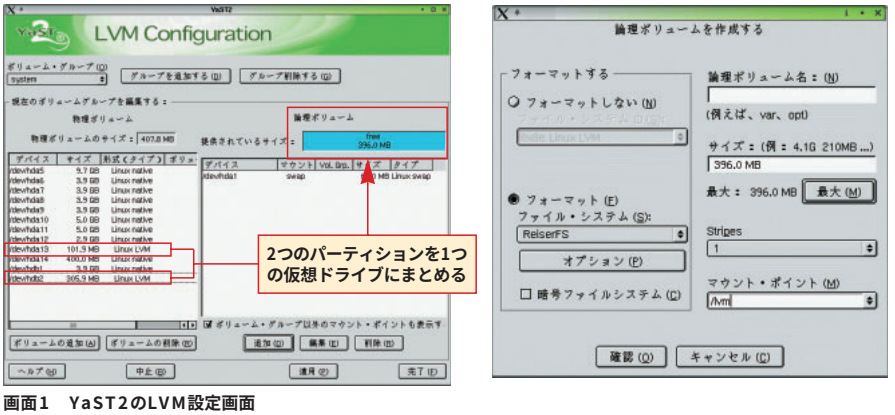
さて、SuSE Linux 7.3はext2のほかにもext3、ReiserFS、JFSといったファイルシステムをサポートしており、仮想ドライブ上に作成したパーティションに、それらのファイルシステム構築できるようになっている。

YaST2を使ってパッケージを管理

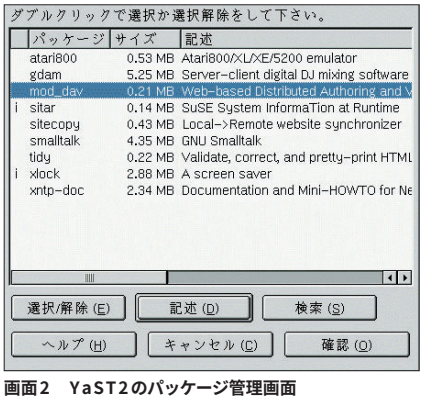
RPMパッケージのインストールも、YaST2を使えば簡単だ。

YaST2のパッケージ管理画面で、たとえば「dav」をキーワードにして検索すると、「dav」を含むパッケージが一覧表示される（**画面2**）。

このあと、希望のパッケージを選択して[確認]ボタンをクリックすれば、パッケージのインストールが始まる。このとき、そのパッケージに必要なものも一緒にインストールしてくれるので、ユーザーがパッケージのインストール時に、依存関係の問題に悩むことはない。



画面1　YaST2のLVM設定画面
/dev/hda13と/dev/hdb2を1つの仮想ドライブとして束ねる。このドライブ（論理ボリューム）に仮想的なパーティションを作成し、ReiserFSでフォーマットしたうえで/lvmにマウントする。



画面2　YaST2のパッケージ管理画面
「dav」をキーワードにして、Apache用のWebDAVモジュールを検索しているようす。

また、YaST2はCD／DVD-ROMのほかに、FTPサイトからのパッケージインストールにも対応しているので、ネットワーク経由でのパッケージアップデートも可能だ。

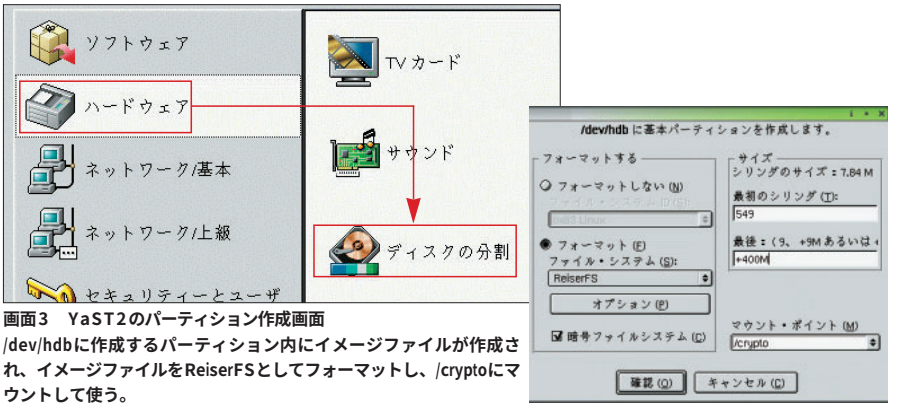
ファイルにもセキュリティを

インターネット常時接続時代の到来で、一般家庭のマシンにもセキュリティ対策が求められるご時世となった。

SuSE Linux 7.3には、YaST2を起動しワンクリックでネットワークセキュリティを設定できる「Personal Firewall」機能が備わっている。

これは、あらかじめ設定されたパラメータを元にして、iptablesで外部ホストからの接続を制限する機能だ。パラメータは、YaST2の画面で、「外部からの接続をすべて拒否する」という内容の項目に「yes」や「no」などと入力すれば自動的に生成される。

さらに、SuSE Linux 7.3には「暗号ファイルシステム」という機能が用意されている（**画面3**）。



画面3　YaST2のパーティション作成画面
/dev/hdbに作成するパーティション内にイメージファイルが作成され、イメージファイルをReiserFSとしてフォーマットし、/cryptoにマウントして使う。

YaSY2の「ディスクの分割」で、［暗号ファイルシステム］をチェックして新規にパーティションを作ると、そのパーティションにイメージファイルが作成される。そして、Linuxのループバックデバイス機能を通して、イメージファイルの中にファイルシステムが構築される。

イメージファイルはtwofishアルゴリズムで暗号化されており、パーティション作成時に設定したパスワードを知らない所有者以外の人間は、イメージファイル内のファイルにアクセスできない。この機能を使えば、ノートPCが紛失や盗難の憂き目にあっても、データを他人に見られる心配がない。

●

現在、SuSE Linux 7.3にバンドルされるSunのオフィススイートStarOfficeの商標問題で、日本でのSuSE Linuxの販売は見送られているが、この問題が解決され、完成度の高いSuSE Linuxが日本のLinuxユーザーにお目見えできる将来が望まれるところだ。

Column

LinuxはUNIXを超えるのか？

UNIXクローンと呼ばれてきたLinuxが、本家UNIXに接近しつつある。

●

コンサルティングを手がけるD.H.Brown Associatesが1999年にLinuxの性能を調査した際、当時のLinuxはWebサーバ、ファイルサーバ、プリントサーバとして優れたパフォーマンスを発揮するものの、UNIXが得意とするWebアプリケーションサーバやデータベースサーバの分野では大きく劣るとされていた。

しかし、最近、同社が再調査したところ、いくつかのLinuxディストリビューションは、すでにローエンドUNIXを凌ぎハイエンドUNIXのパフォーマンスに迫っているという。D.H.Brown Associatesは、今年9月にリリースした「2001 Linux Function Review」というレポートの中で、

・大規模サイトでの運用
・システムの信頼性と有効性
・システム管理のしやすさ

などの項目について、Red Hat Linux 7.1、SuSE Linux 7.2、Caldera OpenLinux 3.1、Turbolinux Sever 6.5、Debian GNU/Linux 2.2r3というLinuxディストリビューションと、ハイエンドUNIX、ローエンドUNIXについて性能比較を行っている。

このレポートによると、SuSE Linux 7.2は、オラクルデータベースにもっとも早く対応することや、LVMなどサーバに必要な機能とそれを使うためのツールが充実していることなどで、総合的にもっとも優れたLinuxディストリビューションだという。

また、Red Hat Linux 7.1はSPECweb99 BenchmarkというWebサーバベンチマークでの実績や、サードパーティ製ソフトウェアの充実度などで、大規模サイトでの運用について、もっとも高い評価を受けている。

そして、LinuxディストリビューションにSuSE Linux 7.2、 Red Hat Linux 7.1、Caldera OpenLinux 3.1、Turbolinux Sever 6.5という順番の総合評価をつけ、これらのディストリビューションのパフォーマンスは、ハイエンドUNIXとローエンドUNIXの間にあると結論している。なお、Debian GNU/Linux 2.2r3は、ローエンドUNIXよりも劣るとされている。

D.H.Brown Associatesは、Webサイトでこのレポートのまとめ部分を配布している。興味のある読者はダウンロードして読んでいただきたい。

■D.H.Brown Associates, Inc
http://www.dhbrown.com/

Mandrake Linux 8.1

フランス産ディストリビューションMandrake Linuxの最新版、Mandrake Linux 8.1（Mandrake 8.1）が、9月27日にリリースされた。Mandrake 8.1は、基本コンポーネントにカーネル2.4.8、glibc2.2.4、XFree86 4.1.0を、デスクトップ環境にKDE 2.2.1とGNOME 1.4.0を採用している（**表1**）。

ファイルシステムを網羅

Mandrake 8.1は、ジャーナリングファイルシステムとして、ext3、ReiserFS、XFS、JFSをサポートしている。

Mandrake Linuxは、早くからReiserFSをサポートしていたこともあってか、新しいファイルシステムにもいち早く対応している。

もちろん、サポートされるこれらのファイルシステムをインストーラのパーティション設定画面（**画面1**）で選択して、ルートパーティションとして使うことも可能だ。

また、後述するMandrake Control Centerという設定ツールでも、XFSなどを使ったパーティションを作成できる。

中途半端な日本語対応

前バージョンのMandrake 8.0では、インストーラ起動直後の言語選択場面で、日本語を表示するために［Japanese］を選択すると、以降の画面でメニューなどが文字化けしてインストールができない状態になっていたが、Mandrake 8.1では文字化けすることなく、きちんと日本語表示された状態でインストールを進められた。

ただ、インストール後にKDEを選択してログインすると、デスクトップのメニューはほとんど文字化けして、KDEの設定もままならない状態になってしまった。インストーラの不具合が修正されたと思った矢先に、今度はデスクトップの不具合だ。

コンポーネント	バージョン
カーネル	2.4.8
glibc	2.2.4
XFree86	4.1.0
Samba	2.2.1a
Apache	1.3.20
GNOME	1.4.0
KDE	2.2.1

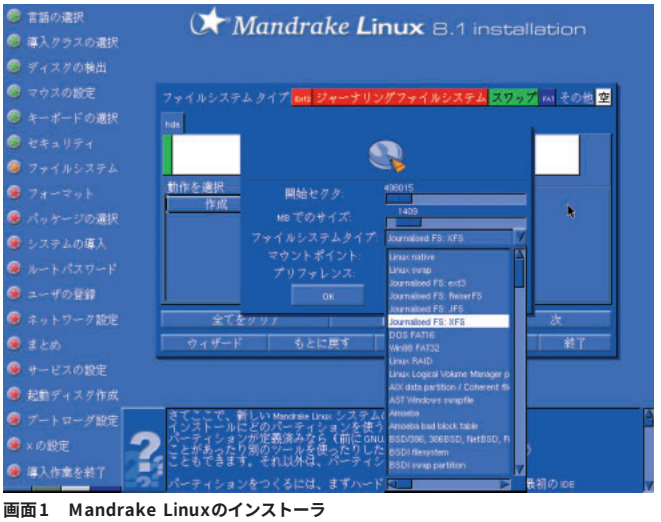
表1　Mandrake Linuxに収録される基本コンポーネント

統合設定ツールMandrake Control Center

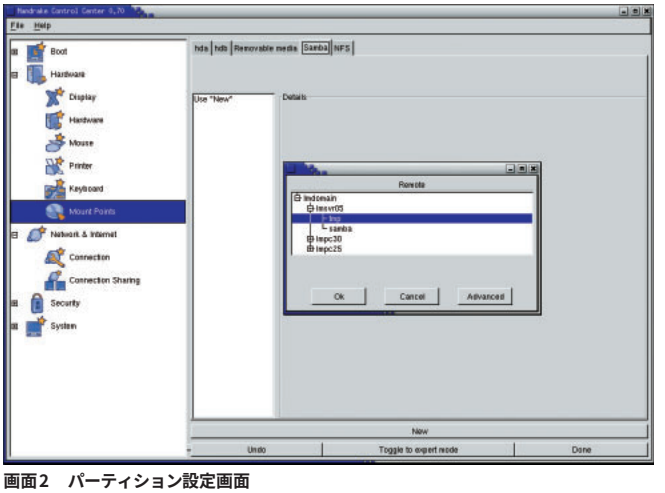
システム管理を一括して行えるMandrake Control Center（以下、コントロールセンター）は、Mandrake Linuxの大きな特徴だ。

■ディスク管理

コントロールセンターの［Mount Points］で起動するパーティション設定ツール（**画面2**）は、ローカルディスクだけでなく、NFSを利用したネットワーク上のディスクや、Windowsネットワーク上の共有ドライブ使用も設定できる。



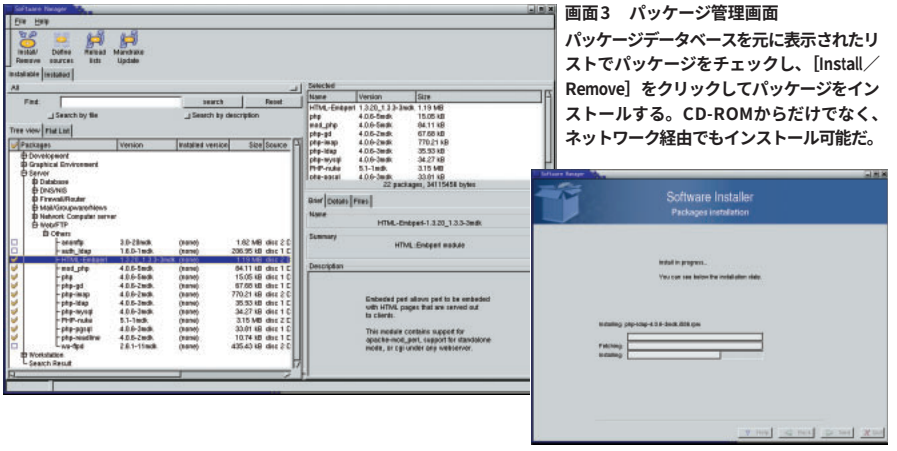
画面1　Mandrake Linuxのインストーラ
インストーラのパーティション設定画面では、Mandrake Linuxがサポートするファイルシステム、ext2、ext3、ReiserFS、XFS、JFSを使ってパーティションを作成できる。



画面2　パーティション設定画面
ディスク名の［Samba］タブを選択して、［New］をクリックすると、ネットワーク上のWindowsマシンを検索して、共有フォルダを表示する。表示された共有フォルダは、Mandrake Linuxのディレクトリにマウントして使う。

コントロールセンターから起動したパーティション設定画面で［Samba］タブを選択し、画面下側の［New］をクリックすると、ネットワーク上のWindowsマシンを検索し始める。

ネットワーク上に利用可能な共有ドライブが見つかると、Windowsのホスト名と共有ドライブ名が表示されるので、あとは、ローカルにあるパーティションと同じように、適当なディレクトリに共有ドライブをマウントして利用すればよい。



画面3　パッケージ管理画面
パッケージデータベースを元に表示されたリストでパッケージをチェックし、［Install／Remove］をクリックしてパッケージをインストールする。CD-ROMからだけでなく、ネットワーク経由でもインストール可能だ。

■パッケージ管理

コントロールセンターで［Software Manager］を選択すると、パッケージの管理画面が立ち上がる（**画面3**）。

このツールは、あらかじめ作成されたパッケージデータベースにしたがってカテゴリ分けされたパッケージの一覧表示や、指定したキーワードからのパッケージ検索が可能だ。

このツールは、FTPサイトなどからのネットワークインストールにも対応しているので、このツールを使ってアップデートパッケージをインストールして、システムを常に最新状態に保つことも可能だ。

■ネットワーク設定

コントロールセンターの［Connect］では、クライアントとしてのネットワーク設定を行う（**画面4**）。

この画面の［Configure］をクリックすると、ネットワーク接続のウィザードが表示される。ウィザードは、Windowsのコントロールパネルから起動するものと似た構成なので、Linux初心者でも簡単に接続設定できる。

■セキュリティ設定

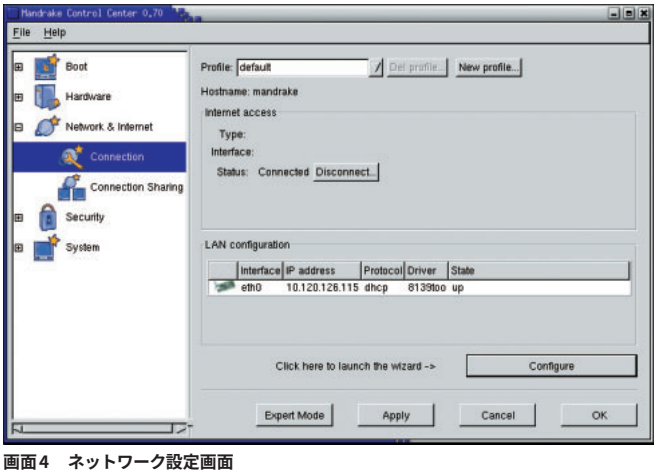
コントロールセンターで［Security Level］を選択し、3つあるセキュリティレベルの中から1つを選択すると、パラメータが設定され、そのパラメータをもとにパケットフィルタリング用のスクリプトが実行される。

Linuxに備わるパケットフィルタリング機能自体は優れたものだが、初心者がipchainsやiptablesの書式を理解して使いこなすのは難しい。

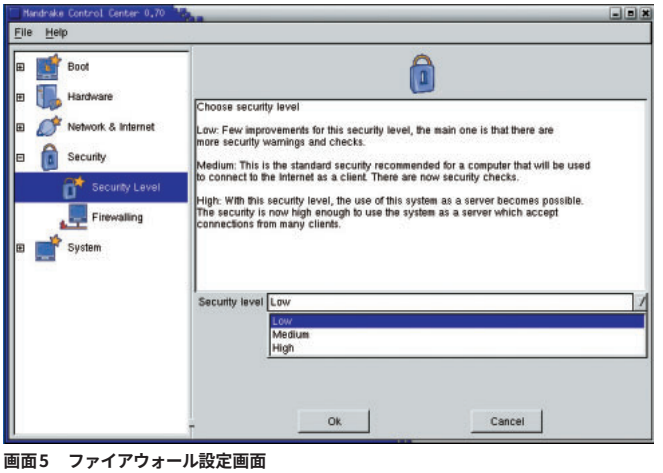
しかし、だからといって一般家庭でも常時接続できる現在では、セキュリティ対策に待ったはない。その点、初心者でも使えるMandrake Linuxのセキュリティ設定ツールは、高く評価できる。

●

Mandrake Linuxらしく、各パッケージはほぼ最新バージョンが収録されている。しかし、日本語環境の出来が良くないと、日本のユーザーには魅力が半減だ。今後、さらなる日本語対応を期待したいところだ。



画面4　ネットワーク設定画面
［Configure］をクリックすると、ネットワークを設定するためのウィザードが立ち上がる。Windowsなみの使いやすさだ。



画面5　ファイアウォール設定画面
3つあるレベルから1つを選択すると、それを元にiptables用のルールを生成して、iptablesでパケットフィルタリングを開始する。

MIRACLE LINUX Standard Edition Version2.0

オラクルデータベースに最適化したMIRACLE LINUXを開発するミラクル・リナックスが10月24日、メジャーバージョンアップとなるMIRACLE LINUX Standard Edition Version2.0（以下、MIRACLE LINUX 2.0）をリリースした。

MIRACLE LINUX 2.0にはインストールCDやサーバ管理ガイドなどが収録されるほか（**表1**）、30日間のインストールサポートが、メールかFAXで3インシデント提供される。

本誌付録のMIRACLE LINUX Standard Edition Version2.0 RC3（以下、MIRACLE LINUX）は正式版リリース前のRC3です。非商用ソフトだけが含まれており、製品版を販売しているミラクル・リナックス株式会社や編集部からのサポートは受けられません。

MIRACLE LINUXのバイナリは、Pentium Pro以上のCPUに最適化されています。このため、PentiumやAMDのK6-2といった古いCPUを搭載するマシンではインストーラが起動しません。あらかじめご了承ください。

また、MIRACLE LINUXはセキュリティ対策のために、デフォルトの状態では一般ユーザーがsuコマンドを使ってrootになれないように設定されています。たとえば、ログイン名が「linuxmag」という一般ユーザーを、suコマンドでrootになれるように変更する場合は、rootでログインしたあとで、

```
# usermod -G wheel linuxmag
```

と実行してください。

ベースをRed Hat Linuxに変更

前バージョンまでのMIRACLE LINUX 1.xは、Turbolinuxをベースに開発されていたが、MIRACLE LINUX 2.0はベースがRed Hat Linuxに変更された。

米国オラクル社はオラクルデータベースの開発プラットホームにSuSE Linuxを採用しているため、オラクルデータベースの動作確認はSuSE Linuxで最初にとられることになる。そして、Red Hat Linuxは、SuSE Linuxに次いで動作確認がとられている。

このため、オラクルデータベースのプラットホームとなることを第一とするMIRACLE LINUXのベースには、SuSE LinuxかRed Hat Linuxが適していると言える。しかし、SuSE Linuxは日本であまり知られていないので、日本市場をターゲットとするMIRACLE LINUXのベースには、Red Hat Linuxが採用されたというわけだ。

このほかにも、ミラクル・リナックスはレッドハットと技術提携を結ぶことで、カーネルやライブラリといった基本部分の拡張をレッドハットに任せ、ミラクル・リナックスはオラクルデータベースの最適化作業に専念できる体制を構築した。

サーバ向けにチューニング

MIRACLE LINUX 2.0は、オラクルデータベースサーバとしてはもちろん、ファイルサーバやインターネットサーバとしても優れたパフォーマンスを発揮するようチューニングされている。

■カーネル2.4の機能をフルサポート

カーネルには安定動作の確認がとれたバージョン2.4.7が採用されている。これにより、32個のCPUや大容量メモリを利用できる。カーネル2.4で64Gバイトのメモリをフルに使うには、Pentium Pro以上のプロセッサが必要だ。このため、MIRACLE LINUXのカーネルはi686に最適化してコンパイルされ、それに応じてRPMパッケージもi686向けに作成されている。

■kparamでカーネルパラメータを設定

MIRACLE LINUXには/proc以下にあるパラメータを設定するコマンド「kparam」が付属する。このコマンドを使えば、システム稼動中に共有メモリセグメントサイズやカーネルパニッ

収録物	概要
インストールCD 1枚	インストールの用CD-ROM
ソースCD 1枚	ソースパッケージを収録
コンパニオンCD 1枚	商用ソフトや追加パッケージなどを収録
Oracle9i Database for Linux トライアルCD 3枚	90日間限定のトライアル版
インストレーションガイド	通常のインストールからKickstartインストールまでを解説
サーバー構築・運用ガイド	ネットワークやサーバの設定を解説

表1　製品の収録物
このほかにインストールサポートが3インシデントまで30日間提供される。

コンポーネント	バージョン
カーネル	2.4.7
glibc	2.2.3
gcc	2.95
Samba	2.2.1a日本語版
PostgreSQL	7.1.2
Apache	1.3.20
sendmail	8.11.5
BIND	9.1.3
fml	4.0.2
OpenSSH	2.9p2
PHP	4.0.6
imap	2000
Postfix	0.0.20010228
qmail	1.03
djbdns	1.05
Tomcat	3.2.3

表2　収録されるおもなコンポーネント
標準のメールサーバとネームサーバには、それぞれsendmailとBINDが採用されているが、コンパニオンCDにはメールサーバPostfixや、Dan Bernstein作のqmail、djbdnsといったサーバ用のRPMパッケージも収録されている。

画面1　オラクルデータベースをインストールするORANAVI
デスクトップの配置されたアイコンからORANAVIを起動して、オラクルデータベースをスムーズにインストールできる。

ク時の挙動を設定するなどして、ユーザーは利用環境に合わせてシステムをチューニングできる。

■ReiserFSをサポート

インストール後での対応になるが、ジャーナリングファイルシステムとしてReiserFSを利用できる。LinuxはReiserFSのほかにもext3などをサポートしているが、デファクトスタンダードとなるジャーナリングファイルシステムが存在しないのが現状なので、ミラクル・リナックスは、ファイルシステムのテストを続けながら、MIRACLE LINUXに適したファイルシステムを順次採用していくとのことだ。

日本語対応が強化されたSamba 2.2.1a

MIRACLE LINUX 2.0には、独自に日本語対応が強化されたSamba 2.2.1aが収録されている。Samba 2.2.1aは、Windows 2000クライントがドメインログオンできるネットワークのPDCになれることが一番の特徴だ。また、国際語対応となっているので標準状態でも日本語に対応している。

さて、日本語環境で使われるSambaは、標準状態よりもさらに細かい日本語拡張が必要とされる。Samba 2.0系では、このあたりを日本Sambaユーザ会が拡張していたが、人手不足のためか、Samba 2.2.1aがリリースされたあとも、日本語拡張に手がつかない状態が続いていた。

そこで、ミラクル・リナックスは、クリティカルな用途にも耐えられるよう、Samba 2.2.1aを独自に日本語拡張した。この成果はGPL配布されるので、いずれMIRACLE LINUX以外のディストリビューションにも日本語対応が強化されたSambaが収録されるだろうが、MIRACLE LINUXは、オリジナリエーターとしてほかのディストリビューションをリードしていくだろう。

使いやすい管理ツール

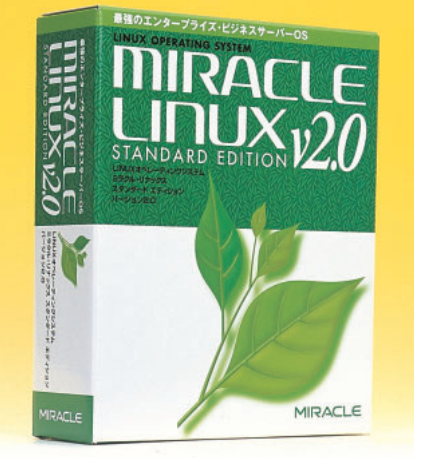
以前のMIRACLE LINUX 1.xでは、Turbolinux付属のTurboToolsをベースにしたMIRACLE Toolsを使ってシステムを管理していたが、ベースがRed Hat Linuxになった今バージョンでは、Webブラウザで操作するWebminがメインの管理ツールに据えられた。

また、これまでMIRACLE Toolsを使ってきたユーザーの便をはかるために、MRIACLE Toolsに似たインターフェイスをもつRed Hat Linuxの「printtool」なども、あわせて収録されている。

一方、オラクルデータベースをスムーズにインストールするためのツール「ORANAVI」も健在で、アイコンをデスクトップに配置して、素早く起動できるように工夫されている（**画面1**）。

●

Linuxディストリビューションの中には、サーバ用のパッケージを収録するだけで、サーバ対応を謳うものも存在するが、サーバとしてのカスタマイズをとことんまで進めたMIRACLE LINUXは、真のサーバシステムになり得るディストリビューションだといえる。



製品名　MIRACLE LINUX Standard Edition Version2.0
価格　6万円
問い合わせ先　ミラクル・リナックス株式会社
03-5562-8300
http://www.miraclelinux.com/

LASER5 Secure Server6.9

レーザーファイブは10月26日、ネットワークサーバに特化したディストリビューション、LASER5 Secure Server6.9（以下、LASER5 Server6.9）をリリースした。

LASER5 Server6.9は、インストールCDやマニュアルなどを収録する構成で（**表1**）、ユーザーにはインストールからコマンドプロンプトの表示までについて、回数無制限のサポートが提供される。

完全ネットワークサーバ仕様

LASER5 Server6.9に収録されているコンポーネントはちょっと古いようだ（**表2**）。これはなぜだろうか。

■カーネルに求めらるもの

LASER5 Server6.9では、カーネル2.2.19とglibc2.1.3が採用されている。現行リリースされているディストリビューションのほとんどが、カーネル2.4やglibc2.2を採用していることを考えると、LASER5 Server6.9の基本構成はいささか古い感がある。

さて、2.4カーネルに追加されたおもな機能が、大容量メモリやSMPのサポートなどであることを考えると、本当に2.4カーネルから恩恵を受けられるのは、Linuxを大規模サイトで使うユーザーとなるだろう。

しかし、セキュアなネットワークサーバとして開発されたLASER5 Server6.9にはカーネル2.4に追加された機能は必要ない。カーネル2.2は、2.4よりもアップデートされた回数が多いのでバグも少い2.2カーネルで必要な機能を得られるならば、安定動作を旨とするサーバに動作実績のあるカーネル2.2を採用するのは自然な発想だ。glibcのバージョンについての事情も、カーネルのそれと同じである。

■サーバにX Window Systemは不要

LASER5 Server6.9には標準でXがインストールされない。これは、Linuxをサーバマシンとして使う場合、Xがサーバのセキュリティを落とすことはあっても、セキュリティを上げることはないからだ。

どうしてもXを使いたいというユーザーのために、X関連のRPMパッケージが［Contrib CD］（**表1**）に収録されているので、必要に応じてインストールすればよいだろう。

コンパイラレベルでクラック対策

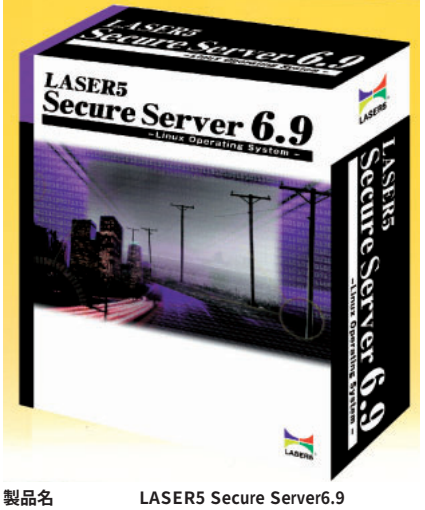
BINDなどでセキュリティホールが発見された際に、「バッファオーバーフロー」という言葉をよく耳にする。

「バッファ」とはBINDをはじめ、プログラムがデータを高速に処理するために、メモリなどに一時的に作るデータ受け渡し場所のような領域のことだ。

データ受け渡し場所として作成した領域のサイズを超えるデータがプログラムに押し付けられた場合（バッファオーバーフロー）、機能不全に陥ったプログラムがクラッカーに乗っとられてしまう可能性がある。

そこで、LASER5 Server6.9を開発しているレーザーファイブと鹿嶋コンピュータサービスは、gccとglibcにProPoliceというパッチを当て、そのgccを使ってRPMパッケージを作成している。

ProPoliceは、IBMの江藤氏が作成するバッファオーバーフロー対策用のパッチだ。このパッチを当てたgccでプログラムをコンパイルすると、そのプログラムには、バッファオーバーフローを自動的に回避するコードが埋め



製品名　LASER5 Secure Server6.9
価格　3万4800円
問い合わせ先　レーザーファイブ株式会社
03-5818-6626
http://www.laser5.co.jp/

収録物	概要
バイナリCD 1枚	インストール用のCD-ROM
ソースCD 1枚	ソースパッケージを収録
Contrib CD 1枚	sendmailやBIND9などの追加パッケージを収録
Commercial CD 1枚	製品版「駅すぱあとイントラネット」などの商用ソフトを収録
マニュアル	サーバ設定やセキュリティ対策などを詳細に解説
サポート	件数無制限のメールによるインストールサポート

表1　製品の収録物
インストールCDは1枚で、商用ソフトとして「駅すぱあとイントラネット」の製品版などが収録されている。また、マニュアルはサーバ管理者必須ともいえるほど充実度が高い。

コンポーネント	バージョン	概要
カーネル	2.2.19	Linuxカーネル
glibc	2.1.3	GNU Cライブラリ
Postfix	20010228	高機能でセキュリティにもすぐれたメールサーバ
Apache	1.3.20	世界でもっとも広く使われているWebサーバ
BIND	8.2.4	インターネット標準ともいえるDNSサーバ
OpenSSH	2.9.9p2	通信内容を暗号化して使うr系コマンド
Squid	2.4.STABLE1	広く使われているWebキャッシュサーバ
Samba	2.0.10日本語版1.1	Windowsネットワークのファイルサーバやプリントサーバとして使う
WU-FTPD	2.6.1	ポピュラーなFTPサーバ
ProFTPD	1.2.1	パフォーマンスとセキュリティにすぐれたFTPサーバ
PostgreSQL	7.1.2	オープンソース界で代表的なリレーショナルデータベース
inetd	0.16	広く使われているインターネットスーパーサーバ
fml	3.0.1	高性能なメーリングリストサーバ
Webmin	0.87	Webブラウザを通して使うシステム管理ツール
rp-pppoe	3.2	ADSL接続で使われるPPPoEクライアント

表2　収録されるおもなコンポーネント
カーネルとライブラリには、安定度を優先させるために、枯れたバージョンが採用されている。サーバパッケージは標準的な構成だ。

込まれる。

LASER5 Server6.9に収録されているプログラムには、ProPoliceを通してバッファオーバーフロー対策がなされているので、この手法を使ってクラックされる可能性は低いのだ。

サーバ管理の初心者でも大丈夫

LASER5 Server6.9には、サーバ管理の初心者でも、LASER5 Server6.9を使いこなせるよう、マニュアルと設定ツールが用意されている。

■内容充実のマニュアル

LASER5 Server6.9に付属するマニュアルは、ネットワーク管理者必携と言える内容だ。

まず、inittabやデーモン起動スクリプトを例にとって、システムの起動過程をわかりやすく解説している。

さらに、ネットワーク起動スクリプトについてはページを変えて、inetdやTCP Wrapperとともに、その仕組みと使い方を解説している。

これらをシステム編と呼ぶならば、このあとにはサーバ設定編とでも呼べる章が続く。サーバ設定編では、LASER5 Server6.9に収録されるサーバ、BIND、Apache、Samba、NFS、Postfix、DHCP、ProFTPD、FMLについて、インストールから設定までが解説されている。

サーバ設定編のあとには、セキュリティ対策編が続く。セキュリティ対策を説く章では、暗号化全般、SSH、IPSec、PGPといったトピックごとに章を立てて、トピック周辺の用語解説から、概念の説明、使用例、設定例などがていねいに書かれている。また、トピックの選択も網羅的で、セキュリティの全体を見渡せる内容だ。

■GUI設定ツールWebmin

LASER5 Server6.9にはセキュリティ対策のためにXが標準でインストールされなかった。ユーザーはコマンドラインでLASER5 Server6.9を操作することになる。そこで、コマンド操作に不慣れなユーザーが、ネットワークやサーバをGUIで設定できるように、Webminとうシステム管理ツールを収録している（**画面1**）。

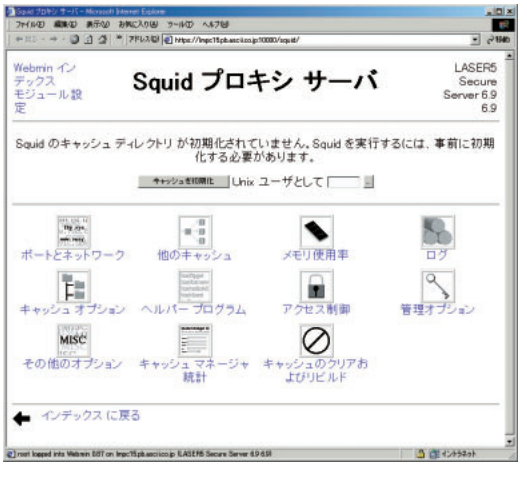
ユーザーは不慣れなコマンド操作に足を引っ張られることなく、Webブラウザを通してシステム管理に専念できるだろう。

なお、WebminはSSLに対応しているので、LASER5 Server6.9をネットワーク越しに操作する際も安全だ。

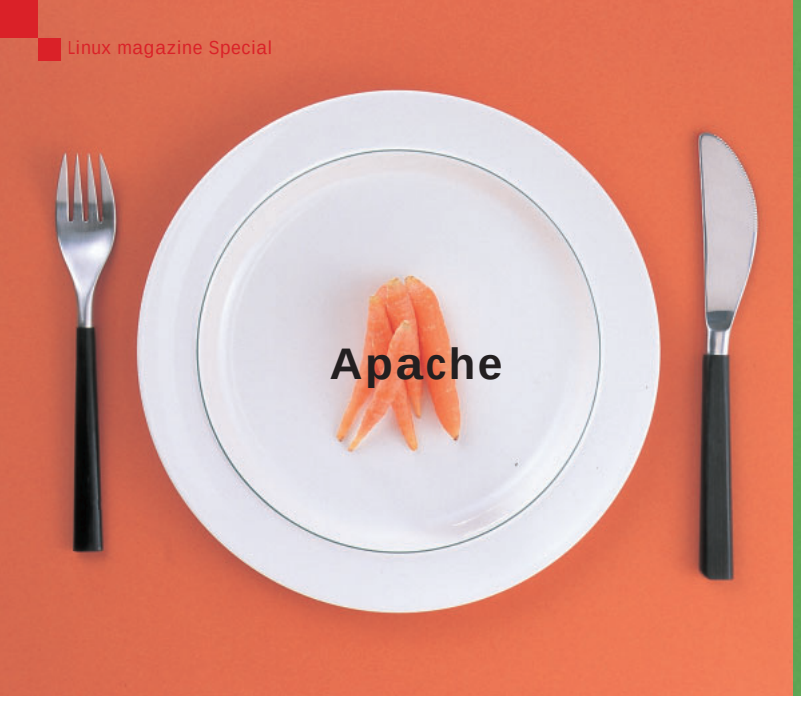
●

LASER5 Server6.9、インターネット常時接続時代サーバとして、非常に良い選択肢となるだろう。

画面1　システム設定ツールWebmin
Webminを使えば、Webブラウザを通してシステムやサーバを設定できる。



画面2　WebminのSquid設定画面
Webキャッシュサーバ Squidについて、アクセス制限を設定したり、メモリの使用状況を確認できたりする。

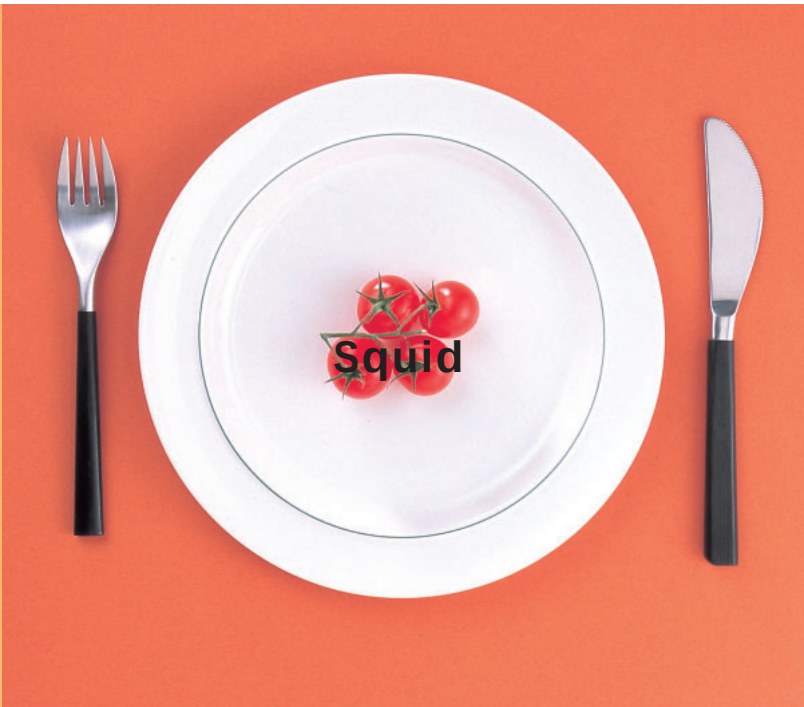


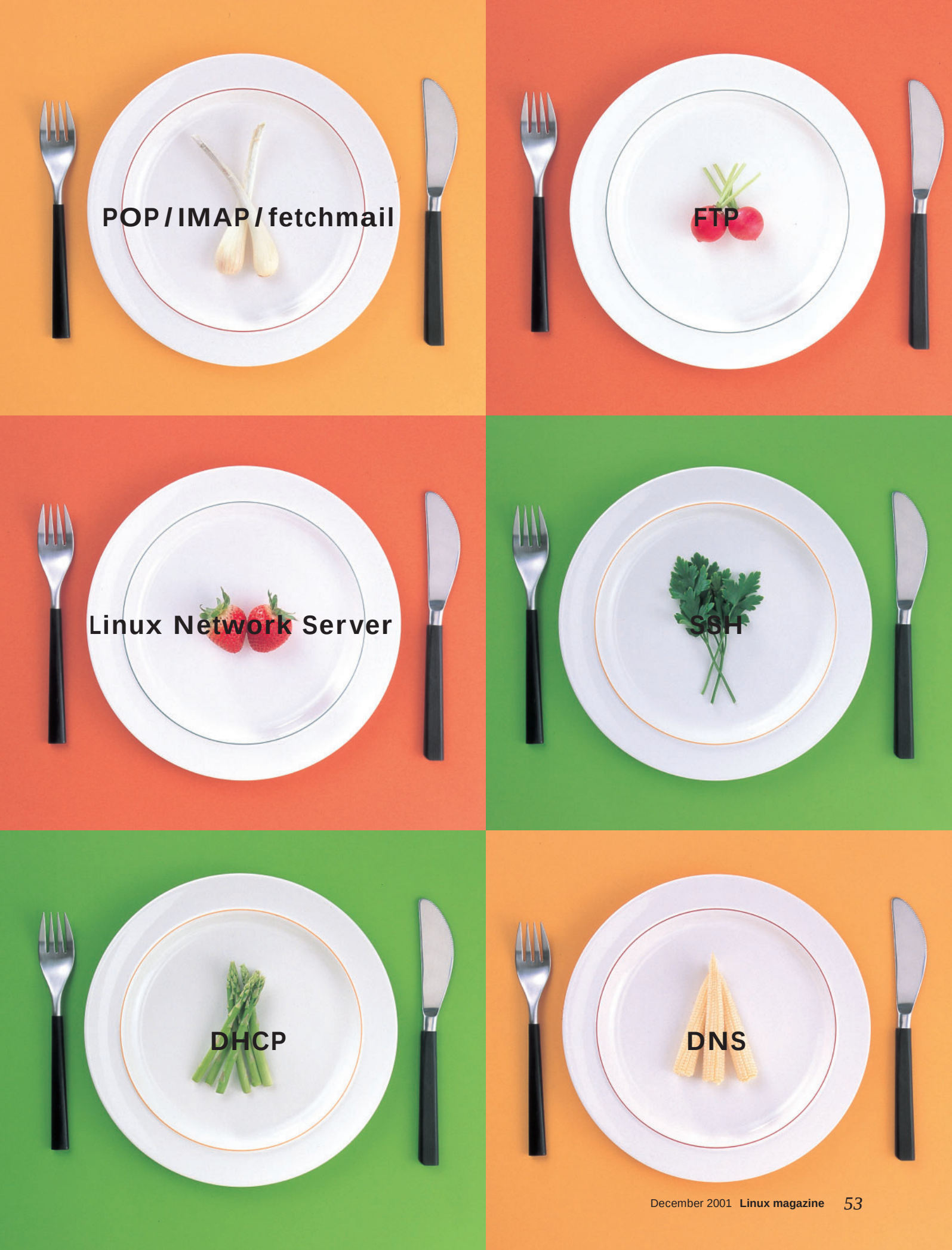
基礎から始めるネットワーキング

Linuxネットワークサーバ設定レシピ

Webサーバにメールサーバ、DNS、DHCPからファイアウォールまで、
各種サーバとサービスをおいしくいただくためのレシピを伝授！

Photo : Junko Kitade (Pacia)




POP / IMAP / fetchmail
FTP
Linux Network Server
SSH
DHCP
DNS

Linuxとネットワーク。それは、相性の良い組み合わせ。

みなさんの中には、職場や家庭でLANを利用している方も多いと思います。LANとは、Local Area Networkを略したもので、一般には同じ建物の中でコンピュータなどの機器をつないだネットワークのことを指します。LANを導入すると、PCからほかのPCへデータを転送したり、1台のプリンタを複数のPCで共有することができます。

Linuxサーバで LANをよりおいしくする

Windows 98やMeなどは、ネットワーク機能を標準サポートしているので、インターネットへの接続やファイルの共有が簡単に行えます。ネットワークの利用という観点から見ると、Webブラウズやメールの読み書きといったインターネット端末として、あるいは各PC間でファイルをやりとりするピアトゥピア環境を作るには便利なOSといえるでしょう。

ピアトゥピア環境は、特定のサーバマシンを必要とせず、手軽に利用できますが、個別のマシンで設定が必要なことと、データがあちこちのPCに分散して管理しにくいのが欠点となります。LANの中にサーバを置いて各種のサービスを行えば、データや設定を集中管理できますので、管理にかかる手間を大幅に減らすことができます。

そこでLinuxの登場です。動作が安定していてマルチタスク性能も高いLinuxは、ネットワークサーバに適したOSのひとつです。サーバとして使うための数々のフリーソフトや、自由度の高いGPLというライセンスも大きな魅力となります。

今回は、イーサネットによる小規模なLAN環境（**図1**）をモデルに、Red Hat Linux 7.1をインストールしたサーバでいろいろなサービスを動かしてみましょう。Linuxサーバの「server01」は、インターネットとLANの両方につながっていますが、インターネット向けにサービスを提供するためには万全のセキュリティ対策を行わなければなりません。ここではLAN内にだけサービスを行うものとして話を進めます。

Webサーバ

では最初にLinuxサーバでどのようなサービスを行えるのかを簡単に紹介します。

Webブラウザからのリスエストに対して、HTMLファイルや画像ファイルなどのデータを送信するのがWebサーバです。このやりとりには、HTTP（Hyper Text Transfer Protocol）というプロトコルを使うので、HTTPサーバとも呼ばれます。

UNIXやLinuxで動作する代表的なWebサーバプログラムはApacheというもので、Red Hat Linuxをはじめ、ほとんどのディストリビューションに採用されています。

世界中にあるWebサーバのおよそ60％がApacheを使用しているという調査結果もあります（http://www.netcraft.com/survey/）。まさに、世界標準のWebサーバプログラムといえるでしょう。

LinuxサーバでApacheを動かすことで、LAN内にWebページを公開できます。ハードディスクの容量が許す限り、大量のデータを置くことができますし、CGIやSSIを自由に使えるのでぜひ活用してみてください。

メールサービス

メールサーバやPOPサーバ、あるいはIMAPサーバというプログラムを動かすと、LANの中で電子メールを利用できます。

■メールサーバ

メールサーバプログラムは、ほかのサーバにメールを配送したり、メールを受け取って、それをユーザーのメールボックスに格納します。メールを配送したり、受け取るときには、SMTP（Simple Mail Transfer Protocol）というプロトコルを使用するので、SMTPサーバとも呼ばれます。また、

MTA（Mail Transfer Agent）ともいいます。

メールサーバプログラムの代表格がsendmailです。sendmailは、古くから広く使われてきた非常に高機能なプログラムで、Red Hat Linuxなど数多くのディストリビューションで採用されています。長い歴史を経て開発が続けられていることから、今ではあまり使われない機能があったり、設定ファイルが難解だということもあって、最近ではsendmailよりも設定が簡単なPostfixというプログラムも人気を集めています。Vine Linux 2.1.5やKondara MNU/Linux 2.0などでは、sendmailに代わりPostfixを採用して話題になりました。

このほか、qmailやeximというプログラムもよく使われるメールサーバです。

■POP／IMAPサーバ

メールサーバマシンでPOPサーバというプログラムを動かしておくと、クライアントPCからメールをダウンロードできるようになります。POPサーバは、POP3（Post Office Protocol 3）というプロトコルを使ってクライアントPCのメールクライアントソフト（メーラ）と会話をし、メールを渡すプログラムです。

多くのディストリビューションには、ipop3dというPOPサーバプログラムが含まれています。また、qpopperもよく使われるPOPサーバプログラムです。

メールをクライアントPCにダウンロードして、サーバから削除してしまうと、デスクトップPCとノートPCの両方からメールを読みたい場合などに不便です。そこで、メールサーバ上にメールを残したままでメールを振り分けたりできるようにIMAP4（Internet Message Access Protocol 4）というプロトコルが作られました。imapdというプログラムを使えば、IMAP4に対応したメーラでサーバにあるメールを整理したり、メールを検索することができます。

■メールダウンローダ

fetchmailというプログラムは、POP3やIMAP4などでメールサーバからメールをダウンロードします。これを使えば、プロバイダのメールサーバから定期的にメールをダウンロードして、Linuxサーバにためることが可能になります。Linuxサーバでimapdを動かしておいて、IMAP4でメール管理をしたり、複数のプロバイダを巡回することも可能なので、メールのヘビーユーザーの強い味方となることでしょう。

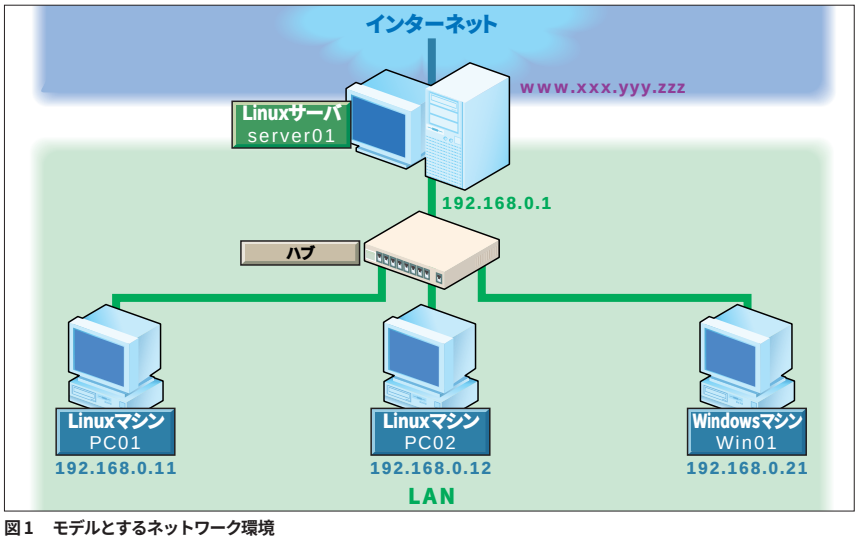


図1　モデルとするネットワーク環境
Linuxサーバ「server01」は、2つのネットワークインターフェイスを持ち、そのひとつ（IPアドレスWWW.XXX.YYY.ZZZ）でインターネットに接続し、もうひとつ（IPアドレス192.168.0.1）がLANにつながっています。

ファイル転送／共有

Linuxマシンをファイルサーバにして、クライアントPCでファイルを共有することができます。

■FTP

FTP（File Transfer Protocol）は、その名の通りファイル転送のためのプロトコルです。Linuxでは、wu-ftpdとProFtpdというFTPサーバプログラムがよく使われています。

LANでは、このあとに紹介するファイル共有サーバを利用するほうが便利なのですが、ほとんどのOSが標準でFTPクライアントを持っているのでイザというときに役に立ちます。また、Webページ作成ソフトウェアには、作成したデータをFTPで転送できるものが多いので、これらを使う際には便利でしょう。

■Samba

WindowsからLinuxサーバのファイルやプリンタを共有できるようにするサーバソフトウェアです。今月号**92ページ**からの「Linuxで作るWindowsファイルサーバ」では、次号で掲載予定の続編との2号にわたり詳細な使い方を解説します。

Sambaには、LinuxからWindowsネットワークを利用するsmbclientというプログラムも含まれています。

■NFS

LinuxやUNIXの間でファイルを共有するにはNFS（Network File System）を使います。NFSでは、サーバの共有ディレクトリをクライアントでマウントして利用します。**図2**のようにすれば、サーバの/home/dataディレクトリをほかのLinuxマシンやWindowsマシンで共有することも可能です。

■Netatalk

MacintoshからLinuxサーバのファイルやプリンタを共有するためのサーバソフトウェアです。Netatalkを含んでいるディストリビューションはあまりありませんが、LANにMacintoshをつなげているならぜひチャレンジしてみてください。

リモート接続

LinuxやUNIXはネットワーク越しにログインして利用できます。

■telnet

Linuxサーバでtelnetサーバプログラムを有効にしておくと、コマンドラインでのログインができます。

非常に便利ではありますが、ユーザー名やパスワードも含めて、通信内容が暗号化されることなくネットワークを流れます。パスワードやデータを盗まれる可能性がありますので、次に紹介するSSHを使うようにしましょう。特にインターネットでtelnetを使ってはいけません。

■SSH

SSH（Secure SHell）は、telnetと同様にコマンドラインでのリモートログインを実現します。パスワードや通信内容を暗号化してやりとりするのでセキュリティが確保できます。

LinuxではOpenSSHというフリーのパッケージがよく使われており、この中にはSSHサーバ、クライアントのプログラムのほかに、安全にファイル転送を行うscpコマンドも用意されています。

■X11

LinuxやUNIXで標準的に使われているX Window Systemは、クライアント／サーバシステムで動作しています。そのため、手元のマシンで動いているXサーバに、リモートマシンで動いているXクライアントのウィンドウを表示することが可能です。

複数のLinuxマシンでXプログラムを実行し、そのウィンドウをすべて1

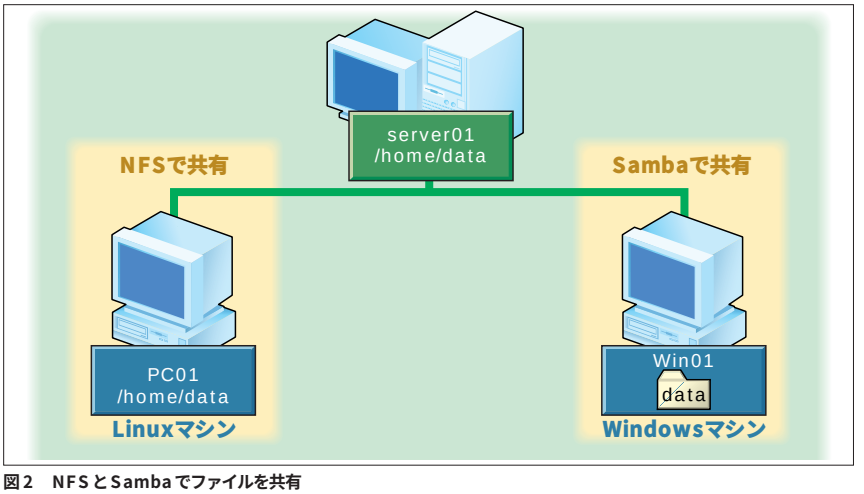


図2　NFSとSambaでファイルを共有
Linuxクライアントは、サーバの/home/dataディレクトリを任意のディレクトリ（この例では/home/data）にマウントできます。また、Windowsマシンは、サーバの/home/dataディレクトリをdataフォルダとしてネットワーク共有できます。

Column

世界標準のTCP/IP

Webやメールはじめとするインターネットのサービスはすっかり一般的なものになりました。インターネットでは、TCP/IPというプロトコルを利用しています。プロトコルというのは、通信手順のことで、データのやりとりを行うための決まりごとです。通信をする機器やソフトウェアは、この決まりに従って動作することが期待されています。そうしないと正しく通信することができません。

プロトコルには階層があり、TCP/IPの上でHTTP（Web）やSMTP（メール配送）、POP3（メールのダウンロード）といった上位のプロトコルを利用して各種のサービスが動くようになっています。

TCP/IPの下位にもプロトコルがあります。イーサネットプロトコルや、ダイヤルアップ接続で利用するPPP（Point to Point Protocol）、ADSLで使われるPPPoE（PPP over Ethernet）などがこの層になります。

PPPでも、PPPoEでもインターネットを利用できることからもおわかりのように、この層がどのようなものであれ、その上でTCP/IPを使うことができれば、さらに上位にあるHTTPなども利用できます。

もちろん、イーサネットによるLANの中でもインターネットと同じサービスを使うことができるのです。

台のクライアントPCに表示して操作することももちろんできます。

NAT／ファイアウォール

Linuxは、NAT（Network Address Translation）という機能を持っています。この機能を使うと、**図3**のようにLinuxサーバにつながっているインターネット回線を、LAN内のマシンで共有できます。

図3のpc01やwin01は、インターネットの先にあるサーバとデータをやりとりしています。pc01から送られるデータには発信元として192.168.0.11というIPアドレスが書かれていますが、Linuxサーバがこれを WWW.XXX.YYY.ZZZに書き換えて送り出します。インターネットのサーバから送られてくるデータは、WWW.XXX.YYY.ZZZ宛てに送られてきますので、Linuxサーバはこれを192.168.0.11に書き換えてpc01に送ります。これがNATの大まかな動作です。

また、LinuxはIPアドレスなどをもとにして特定の通信を遮断したり許可する「パケットフィルタ」という機能を持っています。この機能を使うことで、不正アクセスからLANを守るファイアウォールを構築できます。

プロキシサーバ

プロキシサーバは、クライアントの代わりに接続先のサーバに接続します。もともとは、内部のネットワークと外部のネットワークを切り離して安全性を確保するのが主な目的とされていましたが、Squidというwebプロキシサーバは、Webサーバからのデータを一時的に保管し、同じページへのアクセスがあったときにパフォーマンスを向上させることに主眼を置いています（**図4**）。

ホスト名／IPアドレス管理

クライアントPCの数が増えてくると、ホスト名やIPアドレスが重複しないようにするのが大変になります。そんなときは、LinuxマシンでDHCP（Dynamic Host Configuration Protocol）サーバを動かしましょう。DHCPを使えば、クライアントPCに割り当てるホスト名やIPアドレスなどのネットワーク設定情報をサーバで一元管理できます。

また、ホスト名とIPアドレスの相互変換を行うのがDNS（Domain Name System）という仕組みです。これによって、ホスト名をIPアドレスに変換したり、その逆変換ができます。

人間にとって、数字の羅列に過ぎないIPアドレスでアクセスするマシンを指定するのは不便でしょう。DNSサーバプログラムのbindを使って、LANでつながっているマシンのホスト名とIPアドレスが相互に解決できるようにしておくと、ホスト名でマシンを指定できて便利です。

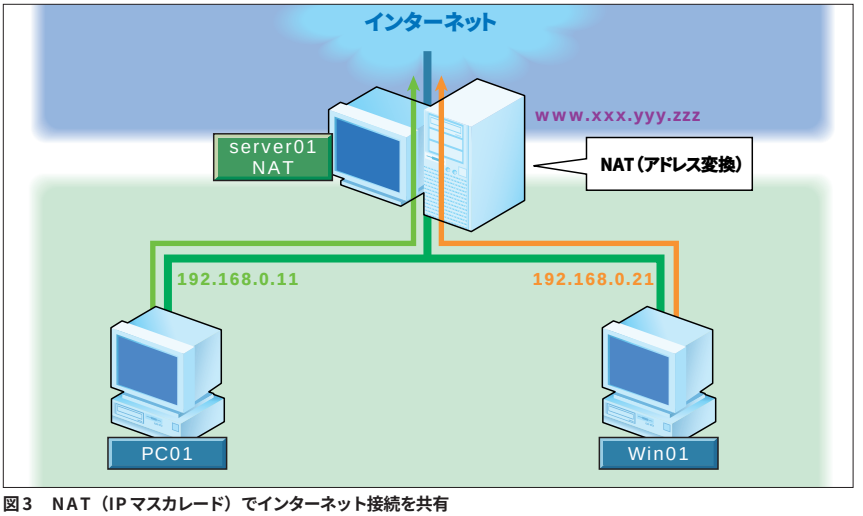


図3　NAT（IPマスカレード）でインターネット接続を共有
Linuxサーバの NAT機能を使うと、LANにつながったすべてのマシンで同時にインターネットを利用することができます。

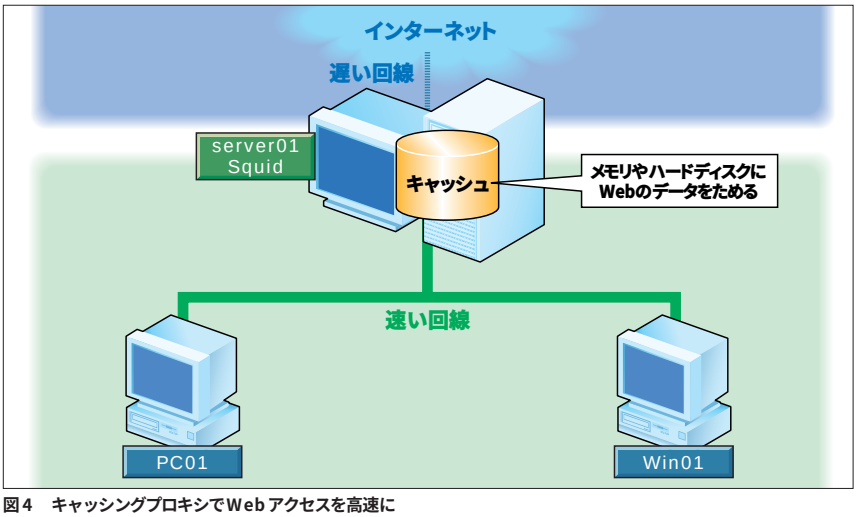


図4　キャッシングプロキシでWebアクセスを高速に
プロキシサーバのSquidは、Webアクセスのデータを保存して、同じデータへのリクエストを高速にします。

サーバクッキングのその前に。下ごしらえが大事です。

LANとLinuxサーバを組み合わせるとどんなことができるのかがだいたいわかったと思います。Linuxサーバでいろいろなサービスを行うためには、まずLinuxサーバ自身がLANにつながっていなくてはなりません。最近のディストリビューションは、インストールのときにハードウェア構成を調べ、できるだけ自動的に設定を行うようになっています。

ここでは、LinuxサーバをLANにつなぐための方法を解説します。サーバプログラムの設定については、次のセクションでサーバごとに詳しく説明しますので、LinuxサーバをLANにつないでからゆっくりと試してみてください。

ネットワーク機器の選び方

LANを組むにあたっては、イーサネットによって接続するのがもっともポピュラーです。イーサネットには、各機器を同軸ケーブルで接続する10BASE-5や10BASE-2、電話のモジュラケーブルに似たツイストペアケーブルとハブを用いて接続する10BASE-Tや100BASE-TXなどの種類があります。また、ケーブルの代わりに電波でデータを送受信する無線LANも普及してきました。

新しくLANを構築するなら、伝送速度が100Mbpsと高速な100BASE-TXに対応するネットワーク機器を購入しましょう。また、無線LANを導入する場合は、伝送速度が11MbpsのIEEE802.11b規格のものがよいでしょう。現在販売されている100BASE-TX対応イーサネットカードは、そのほとんどがLinuxで利用可能ですが、ノートPCで利用するPC Card（PCMCIA）のイーサネットカードや、無線LANカードの中には動作実績のないものもあります。こうした機材を購入する場合には、Linuxディストリビューターやメーカーのwebページなどで動作実績を確認してから購入することをお勧めします（**画面1**、**2**）。ただし、Linuxでの動作を確認していても、動作を保証するメーカーはほとんどないのが現状です。

イーサネットケーブルやハブはOSの種類に関わらず利用できます。100BASE-TXでLANを作る場合は、カテゴリ5以上のストレートケーブルを接続するPCやルータなどの数と同じだけ購入してください。ハブは、100BASE-TXに対応するスイッチングハブを選ぶとよいでしょう。

イーサネットカードの設定

Linuxをインストールするときにイーサネットカードを自動認識していれば、そのカードに対応するデバイスドライバがモジュールとして組み込まれ、カードが動作するようになっているはずです。デバイスドライバというのは、Linuxカーネルが周辺機器を制御するためのソフトウェアで、個々の機器に対応したものが必要です。

Linuxでは、これらのデバイスドライバをモジュールというファイルにしておき、必要に応じて組み込むことができます。モジュールとしてドライバをカーネルから分離することで、カーネルが不必要に巨大にならないようにしているのです。

■カードの状態を見てみよう

さて、イーサネットカードの設定が正しくできているかどうかを確かめてみましょう。次のコマンドを実行してください。

```
$ /sbin/ifconfig
```

画面3のように、eth0とloという名前が付いた2つのネットワークインターフェイスについて情報が表示されているでしょうか。eth0が1つめのイーサネットカードで、loはループバックインターフェイスと呼ばれるものです。複数のイーサネットカードを使う場合は、2つめがeth1、3つめがeth2として表示されます。eth0が表示されない場合は、次の節へ進んでください。では、eth0の状態を見てみます。

画面1　Red Hatのハードウェア動作確認リストページ（http://hardware.redhat.com/）

画面2　プラネックスコミュニケーションズの動作確認リストページ（http://www.planex.co.jp/support/freeos/f_unix.htm）

HWaddrに続いて表示されるのは、ネットワークカード1枚1枚に固有のハードウェアアドレスです。これはMAC（Media Access Control）アドレスともいい、イーサネットなどで通信する場合に相手を特定するために使われます。

inet addr：に続くのが、インターフェイスに割り当てられたIPアドレスです。Bcast：はブロードキャストアドレスで、同じネットワーク内にあるすべてのマシンにデータを送信する際に指定するIPアドレスです。Mask：はサブネットマスクです。これらのアドレスが自分で設定した内容どおりであることを確認しましょう。ネットワーク設定をDHCPで行うようにしているならば、DHCPサーバから割り当てられたアドレスが表示されます。

RX packets：とTX packets：の行は、それぞれ受信パケットと送信パケットに関する統計情報です。イーサネットでは、データを細切れにしてやりとりしており、細切れにされたデータをパケット（小包）と呼びます。errors：に続く数字が多いときは何らかの不具合が発生しています。ハードウェアの設定ミスか故障の可能性があります。

ループバックインターフェイスは、ハードウェアとしての実体はない、仮想的なネットワークインターフェイスで、そのマシン自身にアクセスするときに指定するものです。多くのネットワークプログラムが内部的にこのインターフェイスを利用しています。IPアドレスは127.0.0.1を割り当てることが多いのですが、実は127.0.0.1～127.255.255.254までのすべてのIPアドレスがループバックインターフェイスに割り当てられています。

```
$ /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:40:95:30:15:8A
          inet addr:192.168.0.1  Bcast:192.168.0.255
          Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:121830 errors:0 dropped:0 overruns:0 frame:0
          TX packets:181362 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:10 Base address:0xa000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:5927 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5927 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

画面3　ネットワークインターフェイスの状態を確認する

■イーサネットカードを認識させる

イーサネットカードが認識されていないとifconfigコマンドを実行したときにeth0の項目が表示されません。ハードウェアの自動検出がうまくいかなかったり、カードを交換した場合などは、カードに対応したモジュールが組み込まれずにうまく認識されません。

イーサネットカード用のモジュールは、カードに搭載されたコントローラチップごとに作られています。そのため、使用したいカードがどのようなコントローラを搭載しているかを調べ、それに合ったモジュールを組み込まなければなりません。

コントローラチップの種類を知るためには、カード上にあるコントローラチップ表面に印刷されている型番をメモするか、lspciコマンドを使います。lspciコマンドは、PCIバスにつながれているデバイスの情報を表示するコマンドです。画面にさまざまなデバイス

```
●3Com 3c905の例　（対応モジュール：3c59x）
00:0a.0 Ethernet controller: 3Com Corporation 3c905 100BaseTX [Boomerang]

●DEC 21140チップ搭載カードの例　（対応モジュール：tulip）
00:0b.0 Ethernet controller: Digital Equipment Corporation DECchip 21140 [FasterNet] (rev 22)

●Intel EtherExpress Pro 100＋の例　（対応モジュール：e100またはeepro100）
00:0c.0 Ethernet controller: Intel Corporation 82557 [Ethernet Pro 100] (rev 08)

●Realtek RTL-8139チップ搭載カードの例　（対応モジュール：8139tooまたはrtl8139）
02:02.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-8139 (rev 10)

●VIA VT86C100Aチップ搭載カードの例　（対応モジュール：via-rhine）
00:0b.0 Ethernet controller: VIA Technologies, Inc. VT86C100A [Rhine 10/100] (rev 06)
```

画面4　lspciコマンドでイーサネットコントローラを調べる

```
$ find /lib/modules/2.4.2-2/kernel/drivers/net/ -name '*.o' -exec strings --print-file-name {} \; |
grep -i vt86c100a
/lib/modules/2.4.2-2/kernel/drivers/net/via-rhine.o: VIA VT86C100A Rhine
```

画面5　力わざで対応モジュールを探す

の情報が表示されますので、その中からイーサネットカードの行を見つけましょう。**画面4**は、主なイーサネットコントローラを搭載したカードの出力例です。

コントローラチップの種類がわかったら、次は対応するドライバを調べましょう。ディストリビューターやイーサネットカードメーカーのWebページを見るとだいたいはわかります。わからないときは、少々強引ですがモジュールに書かれた文字列からチップの型番を検索することもできます。lspciコマンドの結果をもとにチップの型番やチップメーカーの名前を検索してみましょう。**画面5**では、ネットワークインターフェイスのモジュールファイルの内部に書かれた文字列から大文字小文字の区別をせずに“vt86c100a”を検索しています。その結果、via-rhine.oというモジュールファイルがこの文字列を含んでいることがわかりました。

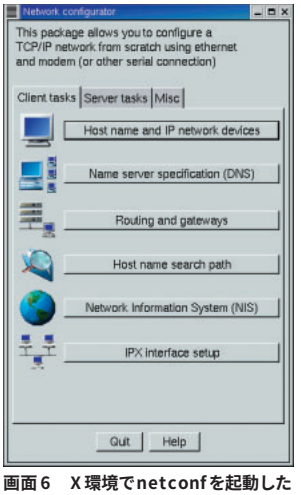
モジュールを組み込むには、rootユーザーになって、「modprobe モジュール名」を実行します。

```
# modprobe via-rhine
```

次にlsmodコマンドを実行し、組み込んだモジュールが表示されれば成功です。Linuxを起動する際にモジュールを組み込むようにするには、/etc/modules.confというファイルに、

```
alias eth0 via-rhine
```

というように、モジュール名を登録します。

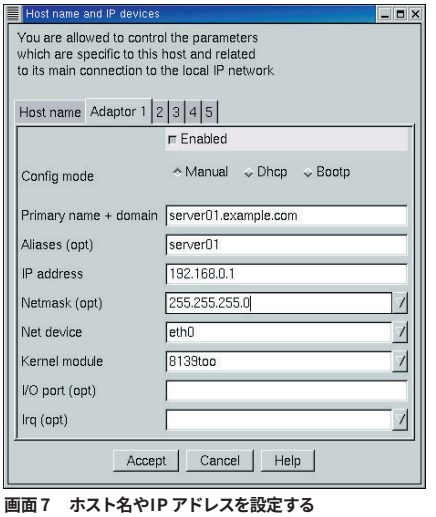


画面6　X環境でnetconfを起動したところ

ネットワークの設定

ネットワークインターフェイスに割り当てるIPアドレスやホスト名などはnetconfコマンドで設定します。netconfは、X環境とコンソールの環境のどちらでも動作しますが、ユーザーインターフェイスは別々になります（**画面6～9**）。

ネットワークの設定をDHCPで行う場合は、Config modeを「DHCP」にして、Kernel moduleを使用する



画面7　ホスト名やIPアドレスを設定する

画面8　コンソール環境でnetconfを起動したところ
項目の移動はTabキー、決定はEnterキーで行う。

画面9　ホスト情報を設定する画面
ドロップダウンリストはCtrl＋Xキーで開く。

カードに合ったモジュール名にするだけ作業は終了です。

手動で行う場合は、Config modeを「Manual」にして、ホスト名、IPアドレス、ネットマスク、モジュール名を指定します。さらに、「Name server specification（DNS）」の画面を開いてDNSサーバのIPアドレスを記入します。

サーバの起動と終了

次のセクションでは、いよいよ各種サーバソフトウェアの設定レシピを紹介しますが、その前にサーバの起動と終了の方法を知っておきましょう。Red Hat Linux 7.1では、各サーバの起動と終了を行うシェルスクリプトが用意されていますので、これを使うと簡単です。スクリプトはサーバごとに用意されており、すべて/etc/init.dディレクトリに収められています。rootユーザーになり、start（開始）、stop（終了）、restart（再起動）といったオプション付きでスクリプトを実行するとサーバを起動したり終了させることができます。たとえば、WebサーバのApacheを起動するなら次のようにします。

```
# /etc/init.d/httpd start
```

サーバの設定ファイルを書き換えたときは、新しい設定を反映させるためにrestartオプションでサーバを再起動してください。

各サーバをLinuxの起動とともに動かすことも可能です。どのサーバを起動時に有効にするかを設定するには、ntsysvコマンドを使うと便利です。rootユーザーになって、次のようにntsysvコマンドを実行しましょう。

```
# ntsysv --level 345
```

--level 345というオプションは、ランレベルの指定です。この例では、テキストログインのランレベル3とランレベル4、そしてグラフィカルログインのランレベル5のいずれかでLinuxが起動する場合の状態を設定します。

ntsysvを起動して表示される画面で、設定するサーバプログラムを選び、スペースキーで起動時の有効／無効を切り替えましょう。名前の前に＊（アスタリスク）が付いているサーバが有効になります。

Column

レシピの読み方

次のページから、主なサーバプログラムやサービスの設定方法を紹介します。個々のサーバプログラムやサービスのレシピには、材料として、

- 必要なRPMパッケージの名前
- 設定ファイルの場所と名前
- サービスの名前

をまとめています。

必要なRPMパッケージがインストールされていなければCD-ROMからインストールしましょう。

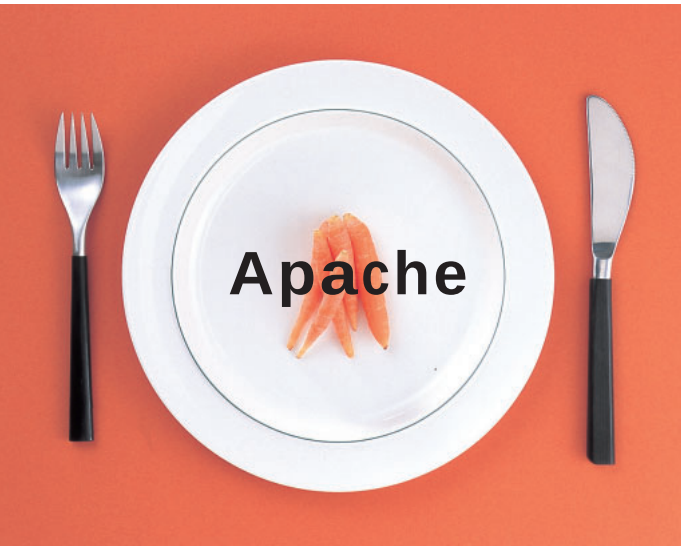
●例　bindのRPMパッケージがインストールされているかどうか調べる

```
# rpm -q bind
パッケージ bind はインストールされていません
```

●例　bindとbind-utilsのRPMパッケージをインストールする

```
# mount /mnt/cdrom
# cd /mnt/cdrom/RedHat/RPMS/
# rpm -ivh bind-9.1.0-10.i386.rpm
# rpm -ivh bind-utils-9.1.0-10.
  i386.rpm
```

また、サービスの名前は、サーバの起動／停止スクリプト、ntsysvで選ぶサービスの名前を示します。



情報の共有／発信をするならWebサーバ

■RPM
apache

■設定ファイル
/etc/httpd/conf/httpd.conf

■サービス名
httpd

「インターネット」といえば、あちこちのWebサイトを見て回ること。そのように思っているユーザーもいるくらい、Webブラウズは一般的になりました。Webブラウザ上に表示されるかっこいいWebページの元となるデータは、ネットの向こう側にあるWebサーバから送られてきます。現在、世界中で最も多く利用されているWebサーバが、Apacheです。Linuxをはじめとするunix系OSだけでなく、Windows上でも動作します。

Apacheの特徴

Apacheは、オープンソースで開発されているため、セキュリティホールが見つかっても、素早く対応が施されます。このためApacheは、商用製品と比較しても、クラックされにくいWebサーバと言っていいでしょう。もちろん、セキュリティ問題は常に進行形であり、「ここまでやれば終わり」というものではありません。古過ぎるApacheをそのまま使って外部に公開するWebサーバを構築するのは、危険です。

Apacheのもうひとつの特徴は、プログラムがモジュール化されていることです。Apache本体は、ごく基本的な機能だけを持ち、それ以外の機能はモジュールによって提供されます。

最新安定版であるApache 1.3.xでは、DSO（Dynamic Shared Object）と呼ばれる仕組みによって、モジュールの動的なロードができるようになりました。ある拡張機能が必要になったところで、そのモジュールがメモリ上にロード・実行されます。これによって、Apache全体で使用するメモリ量がかなり減少しました。

Apacheの設定

Apacheを利用するからには、「外部に向けたWebサーバを構築」したいところです。しかし、そのために必要な設定項目（Apacheではディレクティブと呼びます）をすべてこの場で説明することはできません。そこで、まず手始めにLAN内の各マシンで共有したいデータを、Webブラウザからアクセスできるようにしてみましょう。最近では、HTML形式で書かれたドキュメントが増えてきました。Apache自身のマニュアル（RPMパッケージ名はapache-manual）もHTML形式で提供されています。

以前のバージョンでは、Apacheの設定ファイルは複数に分かれていましたが、1.3.xでは、

```
/etc/httpd/conf/httpd.conf
```

に一本化されています。

httpd.conf内部は、以下の3セクションに大別されています。

（1）全般的な設定
（2）サーバの機能に関する設定
（3）バーチャルホスト関係の設定

1番めは、Apacheの動作に必要となるディレクトリやファイルの指定、モジュールの設定、そしてApacheのパフォーマンスに影響するディレクティブが含まれています。今回のように、小規模なLAN内で使うのが目的の場合は、特に変更する項目はありません。

3番めは、1台のサーバマシンで複数のドメイン（example.comとhoge.com）を利用する際に必要になります。こちらも今回は、特に手を加えずにそのままにしておきましょう。

サーバの機能に関する設定項目

今回のように、初歩的な使用方法で必要になるのは、2番めのセクションです。最低限、以下に示すディレクティブの意味を理解しておくといいでしょう。

■Port

Apacheが用いるポート番号を指定します。RPMパッケージからインストールしていれば、80になっているはずです。一方、tarボールからインストールすると、このディレクティブには8080が指定されます。

Webサーバが用いるデフォルトのポート番号は80です。

```
http://www.example.com/
```

のようにポート番号なしでURLを指定した場合には、80が用いられます。あえて変更しても特に利点もありませんので、80のままにしておきます。

■User、Group

Webサーバに対するリクエストを実際に処理するプログラム（httpd）の所有者（ユーザー）、グループを指定します。Red Hat Linux 7.1に含まれるApacheの初期値は、“ User ”／“ Group ”ともに“ apache ”となっています。

外部に向けてWebサーバを公開する場合には、これらのディレクティブは、セキュリティ問題に関わるので、特に慎重に設定する必要があります。というのは、Apacheのセキュリティホールを利用して不正侵入したクラッカーは、httpdのユーザー／グループの権限を得るからです。仮に“ User ”が“ root ”になっていたら、クラッカーにルートの権限を与えてしまうことになります。

ユーザー“ apache ”は、ログインシェルに“ /bin/false ”が指定されています。このコマンドは何もせず、「失敗」の終了コードを返すだけなので、万が一Apacheがクラックされても、クラッカーにシェルを用いた操作を許さないようになっています。

■DocumentRoot

Webサーバで公開するHTMLファイルを格納するディレクトリを指定します。Red Hat Linux 7.1での初期値は、“ /var/www/html ”となっています。一方、tarボールからインストールした場合の初期値は、“ /usr/local/apache/htdocs ”です。

■DirectoryIndex

URLをディレクトリで指定した場合に、補完されるファイル名を指定します。Red Hat Linux 7.1での初期値は、“ index.html ”など複数のファイルが指定されています。

```
http://example.com/
```

のようにファイル名なしでURLを指定すると、“ DocumentRoot ”ディレクトリにあるindex.htmlなどが読み出されます。

■UserDir

サーバマシン上にアカウントを持つユーザーが、Webに公開するデータを配置するディレクトリを指定します。ユーザーのホームディレクトリからの相対表記で表します。Red Hat Linux 7.1での初期値は、“ public_html ”です。

```
http://example.com/~hoge/
```

のように指定すると、“ /home/hoge/public_html/ ”がアクセスされます。

上記の項目が設定できたら、起動スクリプトを利用して、Apacheを起動します。

```
# /etc/init.d/httpd start
```

サーバマシン起動時にApacheを自動的に実行するためには、以下のコマンドを実行します（サーバのランレベルが3の場合）。

```
# chkconfig --level 3 httpd on
```

動作の確認のため、クライアントマシンでWebブラウザを起動し、

```
http://192.168.0.1/
```

というURLを入力してみましょう。**画面1**が見えていれば、成功です。

ここがポイント！

- **Webサーバのポート番号は80**
- **User／Groupの設定は特に慎重に！**

画面1　Apacheの起動確認画面

ネットワークメッセージングサービスの要
世界中にメールを届けるSMTPサーバ

【sendmail】
■RPMパッケージ
sendmail
sendmail-cf
procmail
■設定ファイル
/etc/sendmail.cf
■サービス名
sendmail

【Postfix】
■RPMパッケージ
postfix
procmail
■設定ファイル
/etc/postfix/main.cf
■サービス名
postfix

メールサービスは、Webと並んでインターネットで最も多く利用されているネットワークサービスです。利用する側からはメッセージを宛先アドレスに送るだけの単純なサービスのように見えますが、実際にメールサーバを公開し運用していくためには、高度なネットワークに関する知識と経験が欠かせません。ここでは、メールサーバを理解する第一歩として基本的な設定方法について解説します。

メール配信の仕組み

メールサーバの設定に話を進める前に、メールサーバがどのようにしてメールを処理するのかについて簡単に触れておきましょう。

図1はインターネットでのメール配信のようすをごく単純に示したものです。この図からメールサーバには、大きく分けて次の4つの役割があることがわかります。

①送信側ユーザーからのメールの受け付け
②受信側サーバへのメールの転送
③送信側サーバからのメールの受け付けと保管
④受信側ユーザーへのメールの配信

これらの役割は、実際には別々のサーバプログラムで実行され、その際に使用されるプロトコルも異なります。**図2**はメールサービスにおけるソフトウェアコンポーネントの構成と働きを図示したものです。図のように、SMTPサーバが前記の①～③の処理を、POPサーバまたはIMAPサーバが④の処理を実行しています。これらのサーバの呼称は、その処理で使われるプロトコルを表しています。

メールサーバには、受信メールを保管するための「メールスプール」と呼ばれるディスク領域が用意されています。メールスプールはユーザーアカウントごとに専用の領域が割り当てられます。SMTPサーバは宛先メールアドレスをもとに、ほかのサーバから受け取ったメールを各ユーザーのメールスプールに振り分け、POP／IMAPサーバは、ユーザーからのリクエストに応じて、そのユーザーのスプールからメールを取り出して配信します。

メールサービスを構成するコンポーネントには、プロトコルをもとにした呼称のほかに、そのコンポーネントの機能をもとにした呼称があります。たとえば、メール配信機能を司るSMTPサーバ（プログラム）はMTA（Mail Transfer Agent）とも呼ばれます。MTAに対して、ユーザー側で動作するメールクライアントソフトウェアはMUA（Mail User Agent）と

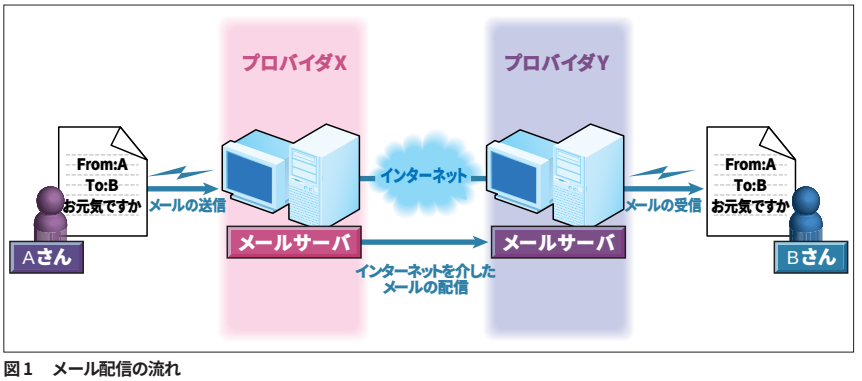


図1　メール配信の流れ

呼ばれます。代表的なMUAには、WindowsのOutlook ExpressやBecky!、LinuxのWanderlustやSylpheedなどがあります。

また、MTAが受信したメールをサーバのユーザーメールスプールに振り分けるプログラムをMDA（Mail Delivery Agent）と呼ぶ場合があります。「場合があります」というのは、MTAによってはMDAの機能を備えていることがあるためです（同じ理由で**図2**ではMDAを特に区別して示していません）。Linuxでは通常、procmailというプログラムがMDAとして使われます。

MTA、MUA、MDAといった用語は、メールサービス関連のドキュメントで頻繁に使われるものです。それぞれの役割を整理して、しっかりと理解しておいてください。

続いて実際の設定について見ていきましょう。ここではLinuxで利用できる代表的なMTAであるsendmailとPostfixの設定を取り上げます。

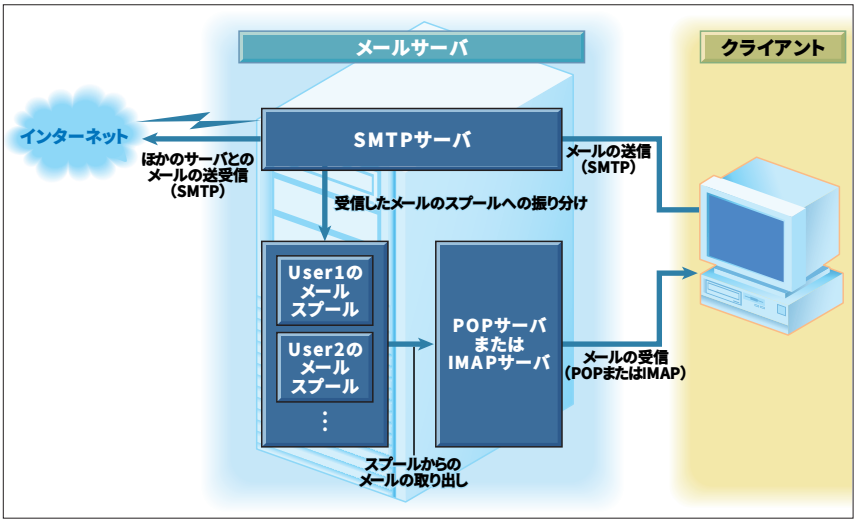


図2　メールサーバのコンポーネントとその働き

sendmailの起動と動作確認

Red Hat 7.1に収録されているsendmailのバージョンは8.11.2です。sendmail本体のほかにも、メールサービスに必要となるパッケージがいくつかあります。前述したように、LinuxではMDAとしてprocmailを使います。また、sendmailの設定には、sendmail-cfという専用ツールとm4というマクロプロセッサが必要です。

サービスを起動する前に、まず現在の動作状況を確認しておきましょう。以下のコマンドを実行します。

```
# /etc/init.d/sendmail status
```

“sendmail（pid XXXX）is running...”という応答があればOK、動作していないようであれば以下のコマンドで起動します。

```
# /etc/init.d/sendmail start
```

今のところインストールしただけのデフォルトの設定でサービスが開始されたわけですが、それでも基本的なメールサーバの機能は利用できる状態にあります。確認してみましょう。Red Hat 7.1のデフォルト設定では、sendmailが動作しているマシンのローカル環境でのみ、その機能を利用できるようになっています。コンソールから次のコマンドを実行すると、ローカルで動作しているsendmailに接続できます。

```
$ telnet localhost 25
```

“25”はMTAとの通信に使われるプロトコルであるSMTP（Simple Mail Transfer Protocol）のTCPポート番号です。SMTPはテキストベースプロトコルなので、telnetを使ってTCPポート25に接続し、コマンドを発行してサーバと直接やり取りすることが可能です。

画面1は、この方法で一般ユーザーからrootにテストメールを送った例です。罫線で囲った部分が実際に入力したコマンドで、それ以外はサーバからの応答です。5行目にある“ESMTP Sendmail 8.11.2”というメッセージから、MTAがsendmail 8.11.2であることがわかります。

sendmailの設定

sendmailの設定ファイルは、/etc/sendmail.cfです。中身を確認してもらえばわかるとおり、非常に多くの設定項目があり、なおかつ各項目の記述方法も複雑です。直接このファイルを編集して設定変更することは難しいので、通常は先ほど紹介したsendmail-cfというツールを利用します。

sendmail-cfは、sendmailの設定ファイル（.cfファイル）を作成するためのm4マクロプロセッサ形式のマクロ（.mcファイル）のサンプル集と、作成手順を自動化するMakefileなどをまとめたものです。Red Hat 7.1では、/usr/share/sendmail-cf/ディ

レクトリ以下にインストールされます（ディストリビューションによっては/user/lib/sendmail-cf/にインストールされているかもしれません）。

サンプルマクロは、sendmail-cf/cf/ディレクトリに置かれています。Red Hat 7.1のデフォルト設定と同じsendmail設定ファイルを作成するマクロのサンプルはredhat.mcです。設定を変更するには、このマクロを雛型にすればよいわけです。では実際に、ちょっと設定を変更してみましょう。ただ、この特集では外部にサーバを公開することは想定していませんから、ローカルマシンでテストできる設定項目としてSMTPのユーザー認証機能を有効にする手順を紹介します。

もともとSMTPにはユーザー認証の仕組みはありませんでしたが、その後プロトコルの仕様が拡張され、現在は「SMTP AUTH」というユーザー認証機構が定義されています。sendmailでは、バージョン8.11.0からSMTP AUTHをサポートしています。SMTP AUTHを使ってメールを送信するには、メールクライアント（MUA）でのサポートも必要です。Linuxで利用できるメールクライアントでは、Mozillaのメールコンポーネントが SMTP AUTHをサポートしています。ここでも設定後の動作確認にMozillaメールを使います。

まず、/usr/share/sendmail-cf/cf/redhat.mcをコピーして実際に修正を加えるファイルを作成します。

```
# cd /usr/share/sendmail-cf/cf
# cp redhat.mc test.mc
```

次に、test.mcにある**リスト1**に示す行の先頭から“dnl”を削除します。変更できたら、マクロを実行し、作成された設定ファイル（この場合はtest.cf）を/etcディレクトリにsendmail.cfという名前でコピーします。このとき、オリジナルのsendmail.cfはバックアップしておきましょう。

```
# make test.cf
# cp /etc/sendmail.cf sendmail.ORG
# cp test.cf /etc/sendmail.cf
```

sendmail側での設定は以上で完了です。変更を反映にするには、sendmailを再起動する必要があります。コマンドは次のとおりです。

```
# /etc/init.d/sendmail restart
```

以上でSMTP AUTHが有効になったはずです。テストしてみましょう。

Mozillaメールの［Edit］メニューから［Mail/News Account Settings...］を選択します。表示されるダイアログの左側のペインにある［Outgoing Server（SMTP）］をクリックします。［Server Name］に“localhost”と入力し、［Use name and password］チェックボックスをオンして［User Name］にはローカルのユーザーアカウント名を指定します（例ではtest-user）。［OK］ボタンを押してダイアログを閉じます。

実際にユーザー宛にメールを送ってみてください。送信ボタンを押すとパスワードの入力を求めるダイアログが表示されるはです。ログインパスワードを入力するとユーザー認証が行われ、パスワードが正しければメールが送信されます。このことからわかるように、Red Hat 7.1のデフォルト設定では、SMTP AUTHのユーザー認証においても通常のログインと同じ認証機構が使用されています。

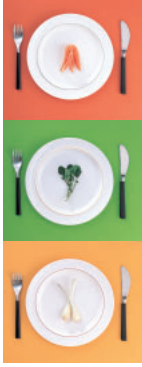
SMTP AUTH以外の設定を変更する場合も、マクロ（.mcファイル）を編

```
$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
220 XXXXX.ascii.co.jp ESMTP Sendmail 8.11.2/8.11.2; Fri, 26 Oct
2001 06:23:27 +0900
Mail From:<test-user@sample.net>
250 2.1.0 <test-user@sample.net>... Sender ok
RCPT To:<root@localhost>
250 2.1.5 <root@localhost>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
TEST MAIL
.
250 2.0.0 f9PLO3303946 Message accepted for delivery
QUIT
221 2.0.0 XXXXX.ascii.co.jp closing connection
Connection closed by foreign host.
$
```

画面1　SMTPによるMTAとのやり取り

▼リスト1　main.cfでSMTP AUTHを有効にする

```
                :
TRUST_AUTH_MECH(`DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
define(`confAUTH_MECHANISMS', `DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
                :
```



集して設定ファイル（.cfファイル）を作成し、/etc/sendmail.cfを置き換えるという基本的な手順は同じです。

Postfixのインストールと起動

Postfixはsendmailに替わるMTAとして開発されたメールサービスシステムで、sendmailとの互換性を保ちながら機能性と安全性を高めることを目標としています。特集の冒頭でも触れたように日本語ディストリビューションでは、Vine Linux 2.1.XとKondara Linux 2.0で採用されています。以降の説明はVine Linux 2.1.5をベースに進めていきます。Postfixのバージョンは、Releases 20010228です。Postfixでもユーザーのメールスプールへのローカルな配信にprocmailが使われます。あわせてインストールを確認しておきましょう。

sendmailと同様に、Postfixもデフォルトの設定でMTAとしての基本的な機能を提供します。起動して確認してみてください。

```
# /etc/rc.d/init.d/postfix start
```

動作確認はsendmailの解説中に紹介した、telnetを使った方法で行えます。今度は、MTAの種類を示す“ESMTP ～”の部分が“Postfix”となっているはずです。

Postfixの設定

Postfixの設定ファイルは/etc/postfix/に置かれています。最も重要なのがmain.fcです。main.fcでは、各設定項目が「パラメータ＝値」という形式で記述されているため、sendmail.cfに比べてかなり読みやすくなっています。また、各項目には詳細なコメントが付けられていて、設定を行う際にたいへん参考になります。

では、Postfixサーバにアクセスするクライアントを制限する設定を例に作業手順を見ていきましょう。

デフォルトの設定では、サーバと同じサブネット上のクライアントからのアクセスが許可されています。これは、main.cfにクライアントアクセスの制限に関するパラメータが記述されていない場合のPostfixのデフォルトでもあります。これをローカルマシンからのアクセスのみを許可する設定に変更してみましょう。デフォルトのmain.cfに次の1行があるはずです。

```
# mynetworks_style = host
```

“mynetworks_style”はPostfixがアクセスを許可するクライアントの「タイプ」を指定するパラメータで、“host”が指定されている場合はローカルホストのみを信頼します。このパラメータに“subnet”が指定されている場合は同じサブネット上のクライアント、“class”が指定されている場合は同じIPアドレスクラスのクライアントからのアクセスを受け付けます。

行頭の“#”は、この行がコメントであることを示しています。つまり、デフォルト設定ではこのパラメータは有効になっていません。このため、デフォルト値である“subnet”が適用されているのです。パラメータ自体を有効にするには“#”を削除すればいいだけですが、実際にはこのパラメータだけではクライアントからのアクセスは制限されません。そのためには、“smtpd_client_restrictions”パラメータを明示的に指定する必要があります。次の1行を追加してください。

```
smtpd_client_restrictions
permit_mynetworks, reject
```

この指定で「mynetworksで指定されたタイプのクライアントからのアクセスは許可するが、それ以外のすべてのクライアントに対してはサーバへのアクセスを禁止する」という設定になります。

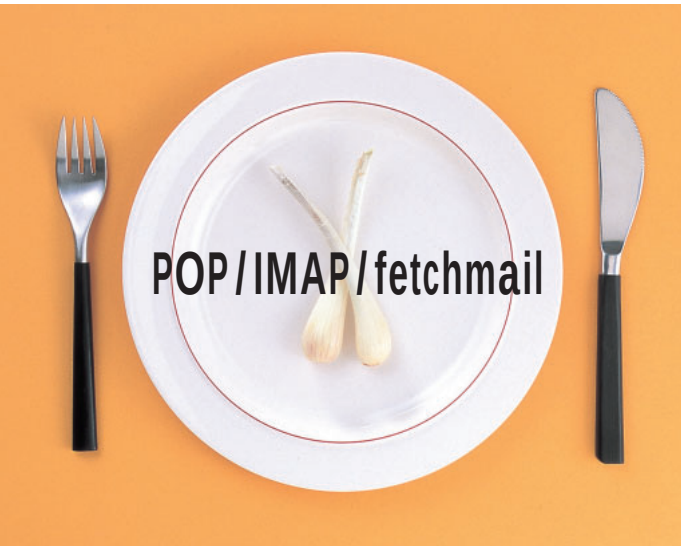
以上の変更で、アクセスをローカルホストのみに制限するための設定は完了です。あとは、この設定をPostfixにロードさせ、変更を反映すればよいだけです。起動の際に使ったpostfixコマンドには、このためのオプションが用意されています。

```
# /etc/rc.d/init.d/postfix reload
```

“smtpd_client_restrictions”パラメータを追加したことからわかるように、すべての利用可能なパラメータがデフォルトのmain.cfに記載されているわけではありません。実際には、ごく一部の基本設定だけです。それ以外のパラメータについては、RPMパッケージに収録されているサンプルファイルが参考になります（インストール先は/usr/doc/postfix-0.0.20010228/sample/）。たとえば、SMTP AUTH関連の設定はsample-auth.cfで詳しく解説されています。また、Postfixのホームページにも、各パラメータについての解説が掲載されています（http://www.postfix.org/basic.html）。

ここがポイント！

- **複雑なsendmailの設定を理解するためにはサンプルマクロが参考になります。いろいろな設定に目を通して知識を深めましょう。**



メールサービスの最終経路。宛先ユーザーとサーバを結ぶ2つのプロトコル

【IMAP／POP】
■RPMパッケージ名
imap
■設定ファイル
/etc/xinet.d/ipop3
/etc/xinet.d/imap
■サービス名
ipop3
imap

【fetchmail】
■RPMパッケージ名
fetchmail
■設定ファイル
~/.fetchmailrc

POP(Post Office Protocol)とIMAP(Internet Message Access Protocol)は、メールサーバのスプールに配信されたメールをクライアント側から操作する際に使われるプロトコルです。ユーザーはこれらのプロトコルでメールサーバにアクセスし、メールアカウント用に割り当てられたスプールに到着した自分宛のメールを読むことができます。このサービスを提供するサーバは、プロトコル名からPOPサーバ、IMAPサーバと呼ばれています。

POPサーバの設定

POPサーバは、ユーザー(MUA)からのリクエストに応じてそのユーザーのメールスプールをチェックし、到着したメールがあれば、それをすべて転送します。このとき、特に指定がない限り転送済みのメールはメールスプールから削除されます。

POPはとてもシンプルなプロトコルですが、そのぶんユーザーにとっても使いやすく、サーバ側でのメンテナンスも容易なことから、リモートクライアントからメールを取り込むための手段として広く普及しています。

■POPサーバの有効化

Red Hat 7.1では、米国のワシントン大学(University of Washington)で開発されたimap-2000のパッケージに含まれるPOPサーバプログラムが利用できます。imap-2000にはPOP2とPOP3に対応した2つのPOPサーバが収録されていますが、ここではPOP3用のサーバであるipop3dを設定してみましょう。

とはいっても、設定手順はごく簡単で、デフォルトで無効になっているサービスを利用できるようにすればいいだけです。この設定にはネットワークサービスツールのntsysv(**画面1**)を使います。

```
# ntsysv --level 345
```

サービスのリストにある[ipop3]にカーソルを移動してスペースキーを押します。アスタリスク(*)でマークされたことを確認して、Tabキーで[OK]ボタンにカーソルを移しEnterキーを押せば設定完了です。

画面1　ntsysvでサービスを有効化する
POPやIMAPだけでなく、あらゆるネットワークサービスのオン／オフを制御することができる。

■POPサーバのテスト

実際にメールを取り出せるか確認してみましょう。POPサーバのコンソールで、以下のコマンドを実行します(このコマンドを実行するにはsendmailがインストールされている必要があります)。

```
$ echo HELLO | mail -s TEST user1
```

“HELLO”がメールの本文、“TEST”がSubjectになります。“user1”の部分にはサーバマシンでのローカルアカウントを指定してください。これでuser1のメールスプールにメールが送られます。

メールの受信テストは任意のメールクライアントソフトで行えます(POP3はほとんどのメールソフトでサポートされています)。ここでは、Mozillaのメールコンポーネントでの手順を紹介しておきます。

Mozillaのメールコンポーネントを起動して、[Edit]メニューから[Mail/News Account Settings...]を選択します。表示されるダイアログの左下側にある[New Acount]をクリックします。ウィザードが起動したら、[Next]ボタンで画面を進めていき[Server Information]ダイアログ(**画面2**)を表示します。[POP]をチェックし、[Server Name]にサーバマシンのホスト名またはIPアドレスを入力します。[Next]ボタンで次の画面に進み、[Account Name]にメールの宛先に指定したユーザーアカウント名を入力します(**画面3**)。最後にもう一度[Next]ボタンを押してウィザードを完了します。

ツールバーにある[Get Msg]ボタンをクリックするとパスワードの入力を求めるプロンプトが表示されるので、アカウントのパスワードを入力して[OK]ボタンを押してください。POPサーバとの通信が開始され、ユーザー認証に問題がなければ、先ほど送信したメールが受信されます。

POPとIMAPの比較

POPサーバを利用してメールを受信する場合、いったんサーバのスプールにあるメールをすべてダウンロードしてから、クライアントマシンのローカル環境でメールを読んだり、管理したりすることになります。

そのため、異なるクライアントマシンからメールを受信した際には、以前に受信したメールを読むことができません。サーバに残すように指定していたとしても、今度はすでに目を通したメールまでダウンロードされてしまいます。いずれにしろ、複数のマシン上にメールが分散してしまうため管理がたいへんです。

このような問題を解決してくれるのがIMAPです。IMAPでは、サーバ上にメールボックスを作成してメールを一元的に管理できます。このほかにも、複数のメールボックスにメールを振り分けたり、メールボックスを他のユーザーと共有したりといった多くの機能を備えています。もちろん、クライアントマシンにメールをダウンロードすることもできます(添付ファイルのみをダウンロードすることも可能です)。

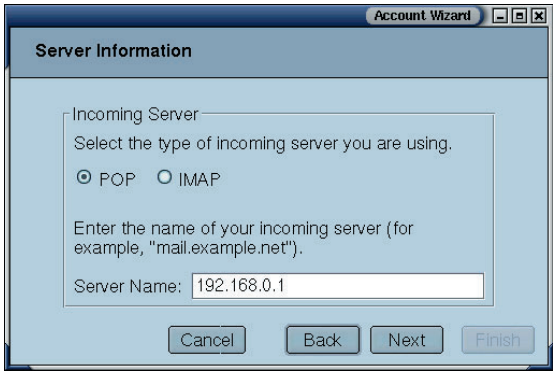
ただ、サーバ上での一元管理というIMAPの最大の利点には、サーバのディスクスペースを消費するという副作用があります。このため、多くのユーザーにサービスを提供しなければならないプロバイダではIMAPをサポートしていないことが多いようです。

また、IMAPの豊富な機能も有効に活用できなければ意味がありません。シンプルなPOPで十分なのか、多機能なIMAPが必要なのか、メールサービスをどのように利用しているかによって選択が分かれるところでしょう。

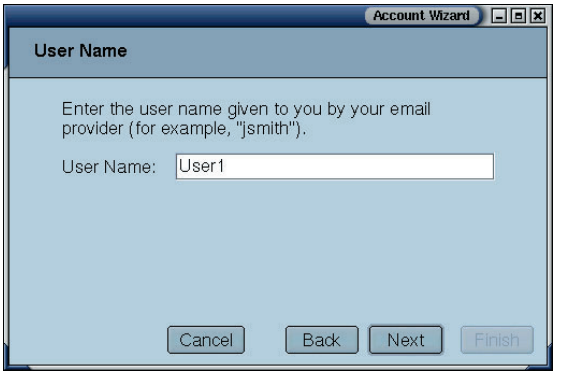
IMAPサーバの設定とテスト

IMAPにもいくつかのバージョンがありますが、現在最も広く普及しているのはIMAP4です。Red Hat 7.1に収録されているIMAPパッケージ(imap-2000)のIMAPサーバ(imapd)もIMAP4をサポートしています。

非常に多機能なIMAPですが、サーバの設定はそれほど難しくありません。POPサーバのときと同じ手順でntsysvを起動して、リストの[imap]



画面2　Mozillaでのpopサーバの指定



画面3　POPサーバのアカウント名を指定する

をマークすれば完了です。テストもPOPサーバとほぼ同じ手順で行うことができます。違いはアカウント作成ウィザードの［Server Information］ダイアログで［IMAP］をチェックすることだけです。

■セキュリティを強化する

メールサーバは、ユーザーの私信（メール）をサーバ上で保管するという性質上、セキュリティに十分すぎるほど配慮する必要があります。特にPOPサーバとIMAPサーバはメールを取得する機能をリモートクライアントに提供するわけですから、さらなる注意が必要になってきます。

標準のPOP3とIMAP4による通信では、ユーザー認証の際にパスワードが平文でネットワークに流されます。つまりパスワード漏洩の危険性があるということです。imap-2000のPOPサーバとIMAPサーバは、CRAM-MD5という認証方式をサポートしています。この認証方式ではパスワードが暗号化されるため、セキュリティを高めることができます。今回の特集では外部にサーバを公開することを想定していませんが、そのときに備えて設定方法を確認しておきましょう。

設定手順はごく簡単で、/etc/cram-md5.pwdというテキストファイルを作成し、ユーザー名とパスワードを記述すればよいだけです。**リスト1**のように1行ごとにユーザーとパスワードを記述します。ユーザーとパスワードの間はスペースではなくタブで区切らなければならないことに注意してください。ここで指定するパスワードは、通常のログイン時のユーザー認証で使われるものとは異なるものにできます。というよりも、異なるパスワードに「すべき」です。そうしておけば、万が一パスワードを知られてしまったとしても、サーバーにログインされることは防ぐことができます。

当然のことながら、パスワードを知られないように措置を施しておくことも重要です。ユーザー認証時には暗号化されますが、このファイル自体はテキストファイルですから、必ず以下のようにパーミッションを変更してrootユーザー以外のアクセスを制限するようにしてください。

```
# chomod 0400 /etc/cram-md5.pwd
```

CRAM-MD5認証を使用するには、メールクライアントでのサポートも必要です。ユーザーの使用するメーラがAPOPあるいはIMAP4／CRAM-MD5をサポートしているかどうか、事前に確認する必要があります。**表1**に主なメールクライアントソフトのサポート状況をまとめておきますので参考にしてください。

▼リスト1　cram-md5.pwdの例

```
# CRAM-MD5 authentication
# "#"ed Lines are comments
user1   vcZ20Lhtc
user2   aatss0LB7
user3   WsiR4xq9s
```

fetchmailを利用したメール管理

fetchmailは、メールサーバにアクセスしてメールを取得してMTAまたはMDAに受け渡すユーティリティソフトです。多くのオプションを備えていて、工夫しだいでさまざまなメールの処理が可能です。さらに、ほかのサーバプログラムと連携させることにより、柔軟なメール管理を実現することができます。Red Hat 7.1には、fetchmail 5.7.4が収録されています。

もともとは、SLIPやPPPなど低速な回線でインターネットに接続するLAN環境において、外部のメールサーバから受信したメールをローカル配信することを目的としていたようです。そのためか、デフォルトでは受信したメールをTCPの25番ポート（つまりSMTPサーバ）に受け渡します。

■サンプルケース

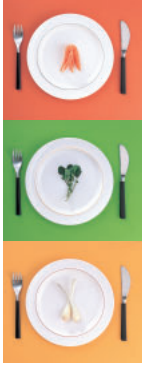
ここでは、複数のプロバイダに加入していて、複数のメールアカウントを持っているユーザーが効率良くメールを管理するための設定をサンプルとして紹介します。ネットワーク構成としては**図1**を想定しています。LANからインターネットへの接続方法は、特に限定していません（この点については後述します）。

おおまかな処理の流れは以下のようになります。

①fetchmailでプロバイダからメールを取得します。2つのプロバイダからの取得を一括して行えるように設定すると便利でしょう
②ローカル配信プログラム（MDA）

ソフト名	バージョン	APOP	IMAP＋MD5
Linux			
Mozilla メール	0.9.3	○	○
Sylpheed	0.6.4	○	○
Windows			
Outlook Express	5.5	○	○
Becky!	2.00.07	○	○
AL-Mail	2.12	○	×

表1　主なメーラのCRAM-MD5認証サポート



であるprocmailに受け渡すことで、LAN環境のユーザーアカウントのメールスプールにメールを配信します

③LAN上のIMAPサーバ経由でメールスプールのメールを操作します。こうすれば、複数のクライアントにメールが分散することもありません

■fetchmailの設定

fetchmailの設定ファイルは、ユーザーのホームディレクトリの.fetchmailrcです。**リスト2**を参照しながら設定内容を確認していきましょう。

・set postmaster "user1"

エラーメールの通知先など使われるユーザーの指定です。ここでは、実際に操作を行うユーザーである“user1”を指定しています。

・set no bouncemail

エラーメールをメールの送信者に転送することを禁止する指定です。

・poll pop.hogehoge.co.jp

プロバイダのメールサーバの指定です。環境に合わせてプロバイダや勤務先など任意のサイトのメールサーバを指定してください。

・proto APOP

メールサーバでサポートしているプロトコルの指定です。fetchmailは、POP3、APOP、IMAPなど多くのプロトコルに対応しています。

・userとpassword

メールアカウントのユーザー名とパスワードの指定です。

・mda "usr/bin/procmail -d user1"

取得したメールをprocmailに渡してユーザーのメールスプールに配信するための指定です。

リスト2からわかるように、メールの取得とローカル配信を2つのプロバイダについて設定しています。この設定により、一度のfetchmailコマンドの実行で両方のプロバイダからメールがダウンロードされ、user1のメールスプールにまとめられます。

fetchmailの実行はインターネットに接続した状態で行う必要があります。また、常時接続環境であれば~/.fetchmailrcに次のような設定を追加することで定期的にメールチェックを行うことができます（間隔は秒単位で指定します）。

```
set daemon 1800
```

あとは、メールクライアントの受信設定をLAN環境のIMAPサーバに変更すれば、複数のアドレス宛に送られたメールを一元管理できるようになります。

ここがポイント！

・手軽さのPOPか一元管理のIMAPか、メールの活用度に合わせて選択しましょう。また、セキュリティも忘れずに。

▼リスト2　.fetchmailの設定例

```
set postmaster "user1"
set no bouncemail

poll pop.hogehoge.co.jp
proto APOP
user "hoge"
password "g2CVaneh8"
mda "/usr/bin/procmail -d user1"

poll mail.example.net
proto POP3
user "nyaa"
password "Ln0Bb6not"
mda "/usr/bin/procmail -d user1"
```

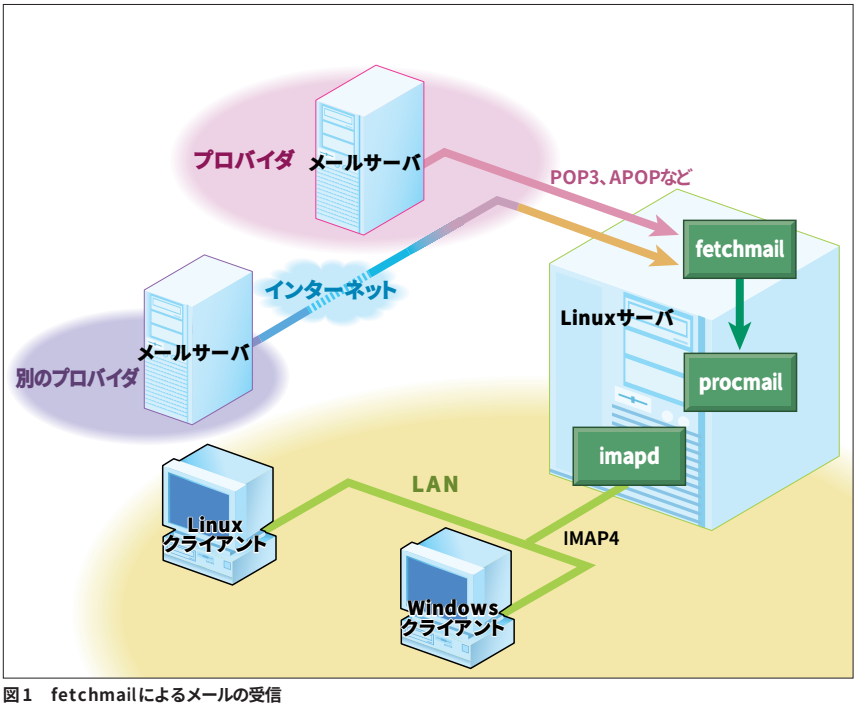


図1　fetchmailによるメールの受信

太く長く生き続けるネットワークファイル転送サービス

【wu_ftpd】
■RPMパッケージ
wu-ftpd
anonftpd

■設定ファイル
/etc/ftpaccess

■サービス名
wu-ftpd

【ProFTPD】
■RPMパッケージ

proftpd
■設定ファイル
/etc/proftpd.conf

■サービス名
proftpd（ftp）

FTP（File Transfer Protocol）は、ネットワーク上のコンピュータ間でファイルをやり取りする手段として長い間使われ続けてきたプロトコルです。現在でもインターネット上には、無数のFTPサーバが存在しており、フリーソフトウェアの配布やWebサーバデータの更新などの目的で多くのユーザーに活用されています。

wu-ftpdの起動と設定

wu-ftpdは、Linuxだけでなく Solaris、HP-UX、AIXなど、多くのUNIX系OSで使われているFTPサーバプログラムです。最新バージョンは2.6.1で、Red Hat 7.1にもこのバージョンが収録されています。

デフォルトでは、wu-ftpdのサービスは有効化されていません。これを有効にするには、ntsysvユーティリティを使う方法が簡単です。ntsysvコマンドを実行して、サービスのリストにある［wu-ftpd］にカーソルを移動します。スペースキーを押してアスタリスク（＊）でマークし、Tabキーで［OK］ボタンにカーソルを移してEnterキーを押します。これでクライアントからの要求に応じてwu-ftpdが起動するようになります。

wu-ftpdの設定は、/etcに置かれている“ftp”で始まる名前を持つファイルに定義されています。基本的な設定は/etc/ftpaccessに記述されています。この設定ファイルでは、ユーザーに提示されるFTPサーバの名前やログインメッセージ、アクセス可能なユーザーのクラス、ユーザーが実行できる操作の制限といった項目を設定できます。ここでは、Red Hat 7.1のデフォルト設定を参照しながら、主な設定項目について解説します。

■ユーザーIDによるアクセス制御

冒頭のコメント行に続く以下の4行では、ユーザーIDとグループIDをもとにFTPサーバへのアクセスを制御しています。

```
deny-uid %-99 %65534-
deny-gid %-99 %65534-
allow-uid ftp
allow-gid ftp
```

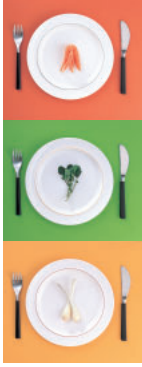
初めの2行でシステムアカウントによるログインを禁止し、あとの2行で例外としてユーザーftpとグループftpにログインを許可しています。ftpは、Anonymous FTPで使われるユーザーアカウントです。

■クラスによるアクセス制御

“class”で始まる行は、ユーザーのタイプとドメイン名、IPアドレスによるユーザークラスの定義です。といっても、わかりづらいと思いますので少し詳しく解説しておきましょう。

デフォルトでは“all”という名前のクラスを定義し、ユーザータイプとしてreal（FTPサーバにアカウントを持っているユーザー）、guest、anonymousが指定されています。行末のアスタリスク（＊）は、ドメイン／IPアドレスのワイルドカード指定です。この設定は「この設定ファイル内では、世界中のホストからアクセスするすべてのタイプのユーザーをallというクラスとして扱う」という意味です。

どのクラスにも当てはまらないユーザーはアクセスを拒否されるので、デフォルトの設定では、アカウントを持っていないユーザーは、guestまたはanonymousとしてログインしない限りFTPサーバにアクセスできないとい



ことになります。したがって“anonymous”を削除することで、anonymous FTPを無効にできるわけです。

複数のクラスが定義されている場合は、先にマッチするほうが適用されます。**リスト1**に示す例では、サイト（example.net）外からのアクセスをanonymousに限定しています。

■パスワード入力の試行回数

“loginfails 5”という指定でログイン時のパスワード試行の回数を制限しています。オーソドックスなセキュリティ対策のひとつです。

■ログの記録

“log”で始まる行では、どのユーザー（real、guest、anonymous）のどんなファイル操作（transfersとinbound、outbound）についてログをとるのかを指定しています。デフォルトでは、すべてのファイル転送についてログをとっています。ログファイルは、/var/log/xferlogです。

●

このほかにも、同時セッション数を制限するlimit、特定のホストからの接続を拒否するdenyなど、多くの設定項目があります。詳しくはftpaccessのmanページが参考になります。

ProFTPDの起動と設定

ProFTPDは、wu-ftpdと同様に細かなアクセス制限が可能で、よりパフォーマンスの高いセキュアなFTPサーバを目指して現在も積極的に開発が進められています。最新バージョンは10月19日にリリースされた1.2.4です。Vine Linux 2.1.5（ProFTPDのバージョン1.2.1を収録）をベースに起動方法と設定を解説していきましょう。

▼リスト1　ftpaccessでのクラスの設定例

```
class  local  real,guest,anonymous  *.example.net
class  remote  anonymous *
```

ProFTPDもやはり、デフォルトでは自動的に起動するように設定されていません。起動方法は少し違っていて、コマンドを実行してデーモンを起動する方式になっています。コマンドは次のとおりです。

```
# /etc/rc.d/init.d/proftpd start
```

マシン起動時に自動的にサービスを開始するには、ntsysvでの指定が必要です。サービスのリストで[proftpd]をマークして有効にしておきましょう。

設定ファイルは、/etc/proftpd.confです。では、こちらもデフォルトの設定を追いながら、主なオプションを確認していきましょう。

■サーバ名とサーバタイプ

“ServerName”は文字どおりサーバの名前の指定です。ここで指定した名前は、ユーザーがFTPサーバにログインした際に表示されます。“ServerType”はproftpdデーモンの動作モードの設定です。デフォルトの“standalone”モードでは、セッションが確立されるごとに子プロセスが生成され、次の接続要求に応えるために待機します。

■TCPポートの指定

“Port”パラメータを使って接続に使用されるTCPポート番号を指定できます。この設定は“ServerType”が“standalone”の場合にのみ有効になります。スーパーサーバ経由でproftpdが起動される場合、つまり“ServerType”が“inetd”の場合は無視されます。

■umask値の指定

“Umask”で始まる行は、FTPユーザーがファイルやディレクトリを作成する際に使われるumaskの設定です。デフォルトでは“022”が指定されているので、putされたファイルのパーミッションは“644”になります。

■プロセス数の制限

“MaxInstances”は接続時に生成される子プロセスの上限値の設定です。これも“standalone”モードでのみ有効です。デフォルトでは“30”が指定されています。

■anonymous FTPの設定

“<Anonymous ~ftp>”と</Anonymous>で挟まれた行では、anonymous FTPの設定をしています。デフォルトでは、接続数の上限が“10”、anonymousユーザーの書き込みはいっさい認めないといった指定になっています。

●

デフォルトの設定はかなりシンプルなものですが、ProFTPDは非常に細かなアクセス制御の機能を備えています。/usr/doc/proftpd-1.2.1に設定項目に関する詳しいマニュアル（Configuration.html）がありますので、設定時の参考にしてください。

ここがポイント！

・FTPサーバにうまく繋がらない場合は、サービスが有効になっているかどうかをまずチェックし、それでもダメなときは設定ファイルでアクセスが制限されていないか確認してみましょう。

SSHを使って通信データを暗号化

■RPM
openssh
openssh-server

■設定ファイル
/etc/ssh/sshd_config

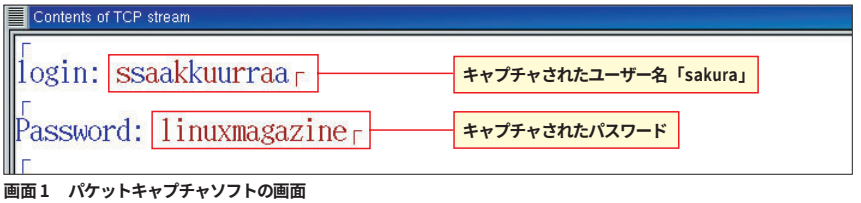
■サービス名
sshd

常時接続環境が普及するにつれて、一般家庭のマシンもクラックの対象になりやすい物騒な時代になりました。ふだんなにげなく使っているアプリケーションは、クラッカーの食指をそそるような状態で、大切な個人情報をネットワークに流しています。

たとえば、ログイン名が「sakura」、パスワードが「linuxmagazine」というアカウントが、server01に用意されているとします。

telnetを使ってwin01からserver01へログインする際にパケットキャプチャソフトを走らせると、簡単にパスワードを取得できます（**画面1**）。

また、ホームページへコンテンツをアップロードするときに使うFTPや、メールの受信に使うPOPなども、通信データを平文で流すという意味ではtelnetと同じ危険性をもっています。



画面1　パケットキャプチャソフトの画面
telnetはパスワードやログイン名などを平文でネットワークに流す

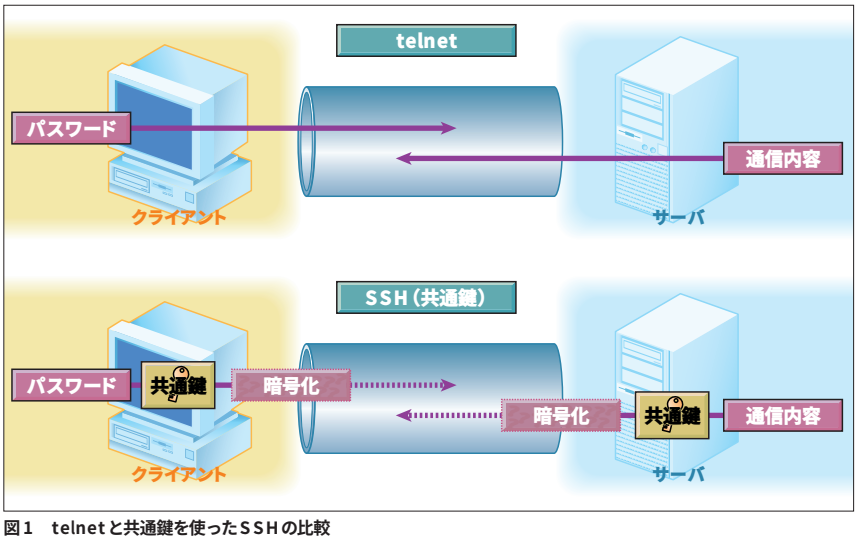


図1　telnetと共通鍵を使ったSSHの比較

SSHを使って安全を確保！

そこで登場するのがSSH（Secure SHell）です。SSHは、通信データを暗号化するので、telnetのようにパスワードなどをクラッカーに取得されることがありません（**図1**）。

また、SSHに組み込まれたSFTPやSCPを使えばファイルを暗号化して転送できますし、SSHのポートフォワード機能を使えばデータを暗号化してメールを受信できたりもします。

さて、SSHには、サポートする暗号化アルゴリズムや、ファイル転送の仕組みなどが異なる2つのプロトコルバージョンがあります。

SSHのプロトコルバージョン1は、公開鍵認証にRSAという暗号化アルゴリズムを使いますが、SSHのプロトコルバージョン2は、RSAに加えてDSAというアルゴリズムも使えます。

また、バージョン1ではファイル転送にSCPを使いますが、バージョン2

ではSFTPを使います。

このレシピでは、SSHのプロトコルバージョン1をSSH1、プロトコルバージョン2をSSH2と呼んで両者を区別します。

公開鍵を使って認証を強力に！

SSHは、共通鍵方式と公開鍵方式という、暗号を使った2つの認証方式をサポートしています。

共通鍵方式を使ってリモートログインする場合（**図1**）、ログイン名やパスワードは、サーバとクライントで共通のアルゴリズム（共通鍵）で暗号化され、ネットワークを流れます。

一方の公開鍵方式では、ユーザー認証に秘密鍵と公開鍵という鍵のペアを使います。公開鍵は南京錠の錠前、秘密鍵は錠前を開ける鍵だとイメージすれば、公開鍵と秘密鍵の対応を理解しやすいでしょう。

SSHで公開鍵方式を使う場合、ユーザーはまず公開鍵と秘密鍵のペアを作成します。作成した公開鍵で封じたデータは、その公開鍵に対応する秘密鍵を使わないと開封できません。

それでは、公開鍵方式を使ってリモートログインする際のおおまかな流れを見ていきましょう（**図2**）。

まず、SSHクライアントからログイン要求を受けたSSHサーバが、サポートするプロトコルバージョンや暗号化方式をクライアントに伝え、クライアントはサポートするプロトコルバージョンなどをサーバに返します。

このとき、クライアントは複数の接続を区別するために、サーバへ接続のID情報を送信しています。

サーバとクライアントがお互いの情報交換を終えると、通信に使うプロトコルバージョンや暗号化アルゴリズムを決めて、通信データの暗号化を始めます。

このあとサーバは、クライアントがサーバに置かれている公開鍵に対応する秘密鍵を持っているかどうかをチェックします。サーバが鍵のチェックのために送った問題に、クライアントが正解を返せば認証は成功です。

このあとは、通信開始時にサーバとクライアントが取り決めたアルゴリズムを使って、データを暗号化します。

SSHサーバ設定のポイントはココ！

それではSSHサーバのセットアップにとりかかりましょう。SSHには商用、非商用の複数の実装がありますが、ここでは多くのLinuxディストリビューションに含まれているOpenSSHをサーバとして使います。

まず、server01上で「$ rpm -qa | grep ssh」を実行し、「openssh」と「openssh-server」がインストールされているかどうかを確認します。インストールされていない場合は、ディストリビューションのCD-ROMからRPMパッケージを探してインストールします。

パッケージをインストールしたら、サーバの設定ファイル「/etc/ssh/sshd_config」を、**表1**を参考にして編集します。

このあと、root権限で、

```
# /etc/init.d/sshd start
```

と実行すれば、SSHサーバが起動します。また、

```
# chkconfig --level 35 sshd on
```

としておけば、マシンの起動時にSSHサーバが自動的に立ち上がります。

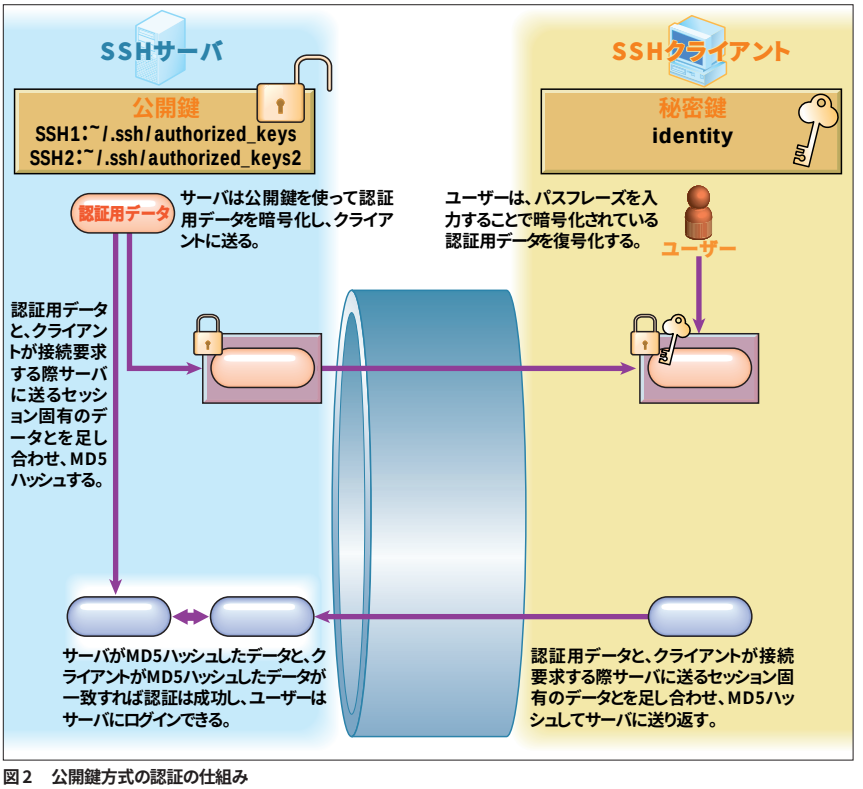


図2　公開鍵方式の認証の仕組み

おもな項目	設定例	概要
Port	22	SSHサーバがクライアントの接続を受け付けるポート番号
Protocol	2,1	SSHサーバがサポートするSSHのプロトコルバージョン。SSH1のみを使う場合は「1」を、SSH2のみを使う場合は「2」を、両方を使う場合は「2,1」または「1,2」と書く
PermitRootLogin	no	スーパーユーザー「root」のログインを制限する。「yes」になっている場合は、「no」に変更して、「root」がSSHを使ってログインできないようにする
StrictModes	yes	ユーザーのホームディレクトリ、公開鍵などを置く「~/.ssh」や公開鍵のパーミッションなどが適切に設定されているかどうかをチェックする
RhostsAuthentication RhostsRSAAuthentication	no	rhostsやhosts.equivを使うr系コマンドと同等のホスト認証、ユーザー認証の設定
RSAAuthentication	yes	「yes」にして、RSAを使った公開鍵認証を有効にする
PasswordAuthentication	no	「no」にして、クリアテキストを使った認証を無効する

表1　sshd_configのおもな設定項目

MindTermは使い方がとっても簡単！

SSHクライアントは数多くありますが、今回はJavaベースのMindTermを紹介します。MindTermは、Windows、Linux、Mac OSなど多くのプラットホームで動作するクライアントで、ターミナル、ファイル転送、ポートフォワードなど多くの機能を備えます。

■JavaとMindTermのインストール

MindTermを使うには、まず**表2**のサイトから、バージョン番号が1.1.x以降のJDKかJREをダウンロードしてインストールします。筆者はwin01にJRE 1.3.1をインストールしました。

次に、(http://www.appgate.com/mindterm）から「mindterm_2.1-bin.zip」をダウンロードして展開すると、「mindterm.jar」というファイルが生成されるので、このファイルをダブルクリックするなどしてMindTermを起動します（**画面2**）。

■公開鍵の作成

[File]—[Create Keypair]を選択して開かれる**画面3**にパスワードなどを入力します。

[Generate]をクリックして開く画面上でマウスを動かし続けると、鍵のペアが生成されます。生成された鍵のペアは、「identity」が秘密鍵、「identity.pub」が公開鍵で、ユーザーのホームディレクトリ内の、「mindterm」ディレクトリに配置されます。

Windows 2000では、「C:\Documents and Settings\ユーザー名\mindterm」などになります。

次に、作成した公開鍵「identity.pub」をserver01の「~/.ssh」ディレクトリに置きます。公開鍵は、フロッピーディスクに保存してコピーするか、FTPなどを使って転送すればよいでしょう。この時点では、まだwin01とserver01の間に安全な通信が確立されていないので、リモートではなく直接server01にログインして操作します。

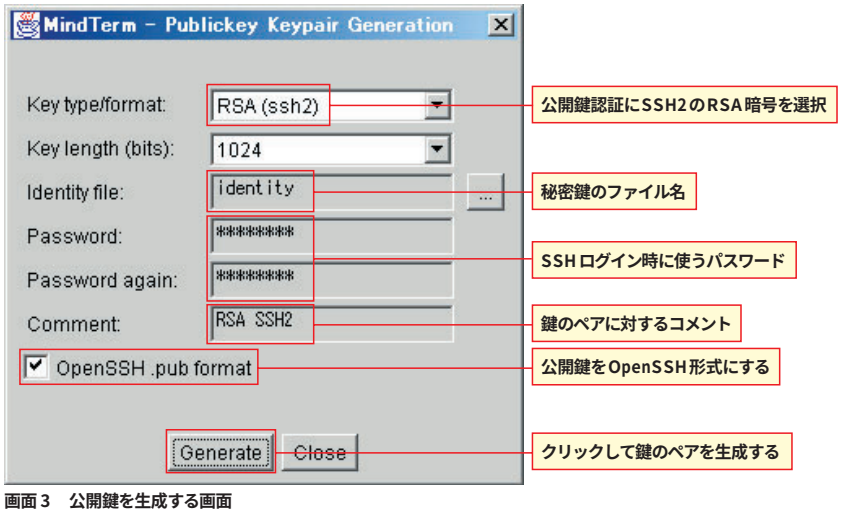
プラットフォーム	URL
Linux	http://www.blackdown.org/java-linux.html http://www.ibm.com/developer/java
Linux、Windows、Solaris	http://java.sun.com/products/archive/
Macintosh	http://www.apple.com/java/

表2　Java実行環境のダウンロードサイト
LinuxでMindTermを使う場合は、ダウンロードする前にディストリビューションのCD-ROMにRPMパッケージが収録されていないかどうかを確認しよう。

画面2　MindTermのメイン画面

SSHクライアント	URL	概要
MindTerm	http://www.appgate.com/mindterm	このレシピで紹介するJavaベースのSSHクライアント
TTSSH	http://www.zip.com.au/~roca/ttssh.html	TeraTermにSSH機能を追加するためのプラグイン
Tera Term	http://hp.vector.co.jp/authors/VA002416/	Windows用のtelnetクライアント
WinSCP	http://winscp.vse.cz/eng/	Windows用のSCPクライアント
PortForwarder	http://portforwarder.nttsoft.net/	Windows用のポートフォワードクライアント
MacSSH	http://www.macssh.com/	MacOS用のSSHクライアント

表3　おもなSSHクライアントの一覧
日本語環境を重視するWindowsユーザーは、Tera TermとTTSSHを組み合わせて使うのがお勧めだ。



画面3　公開鍵を生成する画面

「identity.pub」をserver01の「~/.ssh」に置いたら、「~/.ssh」と公開鍵のパーミッション、鍵のファイル名を変更します。

```
$ chmod 700 ~/.ssh
$ cd ~/.ssh
$ mv identity.pub authorized_keys2
$ chmod 600 authorized_keys2
```

公開鍵のファイル名は、SSH1の場合は「authorized_keys」、SSH2の場合は「authorized_keys2」です。ファイルとディレクトリのパーミッションやファイル名を間違うとあとでserver01は接続を拒否するので、確実に操作してください。

●

■SSHを使ったリモートログイン

[Settings]—[New Server]を選択すると**画面4**が開きます。ログインに必要な情報を入力し、[Connect]をクリックしてserver01に接続します。

server01に初めて接続する際は、server01のID情報を記録するかどうかを尋ねてくるので[Yes]をクリックします。この画面は、初回接続時にしか表示されませんから、2度目以降の接続時にこの画面が表示された場合、server01以外のマシンがserver01を名乗っている可能性があるので注意してください。

■SSHを使って安全にファイル転送！

SSHを使ってのファイル転送は、[File]—[SFTP File Transfer]を選択し、ファイル転送ウィンドウを開いて行います。SFTPはSSH2でのみサポートされているので、SSH1を使う場合は[SCP File Transfer]を選択します。このほか、[FTP To SFTP Bridge]を選択すると、使い慣れたFTPクライアントを使ってファイルを転送できます。この機能はパッシブモードをサポートするクライアントでのみ有効で、接続先のFTPサーバ名を「localhost」として使います。たとえば、WindowsのDOS窓では「ftp localhost」とします。

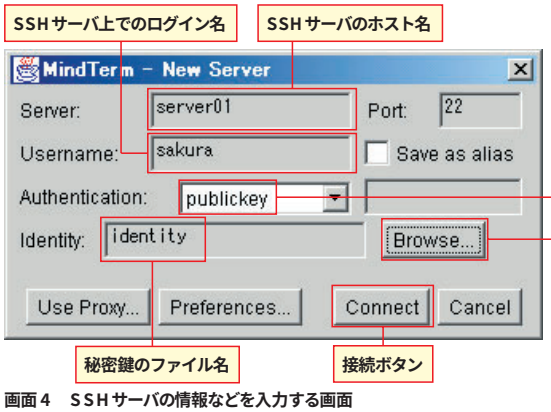
■あらゆるプロトコルを暗号化！

[Tunnels]—[Basic]を選択して**画面5**を開きます。**画面5**のように入力すれば、たとえばDOS窓で「telnet localhost」として、telnetを暗号化した状態で使えます。

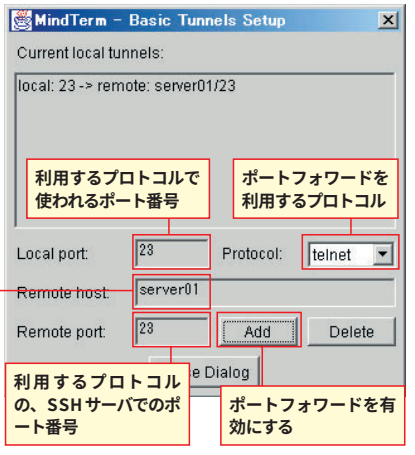
この機能は、telnetだけでなくPOPなどでも使えます。

ここがポイント！

- **接続に失敗したら、「~/.ssh」のパーミッションや、公開鍵のファイル名、パーミッションを確認してください。**
- **公開鍵を他人に渡しても危険はありませんが、秘密鍵は盗まれないようにしっかりと管理しましょう。**



画面4　SSHサーバの情報などを入力する画面



画面5　ポートフォワードを有効にする画面

IPアドレスを有効活用できるNAT

■RPM
ipchains

■設定ファイル
/etc/sysconfig/ipchains
/etc/sysctl.conf
/etc/sysconfig/network

■サービス名
ipchains

ipchainsは、Linuxカーネルに備わっているパケットフィルタリングの機能をコントロールするコマンドです。インターネット上を行き来するデータは、すべてパケットと呼ばれる固まりとして扱われます。パケットフィルタリングは、自ホストに出入りするパケットを、一定の規則に従って受け入れ／拒否／転送を行います。

さらにipchainsは、パケットの転送時にIPアドレスの変換（IPマスカレード）をすることができます。この機能によって、LAN全体をインターネットから隠蔽しつつ、同時にLAN内のクライアントマシンからインターネットにアクセスすることが可能になります。

ipchainsが持つIPアドレス変換の機能は、一般的にはNAT（Network Address Translation）と呼ばれますが、LinuxではIPマスカレード（Masquerade）と表現することが多いようです。

Linuxのパケットフィルタリングに関するコマンドは、カーネル2.0のipfwadm、カーネル2.2のipchains、そしてカーネル2.4にはiptablesがあります。しかし、カーネル2.4を採用したディストリビューションでもその多くは、パケットフィルタリングをipchainsを用いて実現しています。将来的にはiptablesへの移行が進むと思われますが、今回はRPM系ディストリビューションで最も多く使われている、ipchainsを中心に話を進めます。

ipchainsの2つの機能

ipchainsには、以下の2つの機能がまとめられています。

- **IPマスカレード**
- **パケットフィルタリング**

これら2つの機能の目的や用途は、別のものです。すなわち、パケットフィルタリングは、LAN／インターネット間で通すべきではないパケットを遮断することで、簡単なファイアウォールの機能を実現します。

一方、IPマスカレードによって、LAN内部のマシンがインターネット上のマシンと直接（見かけのうえでは）通信できるようになります（**図1**）。

同じipchainsコマンドを用いてはいますが、これらは起動時に行うべきタイミングも異なります。つまり、手を加える設定ファイルも別になります。

そこで本セクションでは、最初にIPアドレスやipchainsコマンドに関する全般的な説明を行い、続いてIPマスカレード、パケットフィルタリングの設定方法を解説します。

IPアドレス

インターネットには非常に多くのマシンが接続されており、それらのマシン同士がデータをやりとりしています。私たちの手元にあるマシンは、その中からどうやって目的のマシンを見つけているのでしょうか。

インターネット上の各マシン（正確にはネットワークインターフェイス）には、IPアドレスという32ビットの数字が割り振られています。IPアドレスは、マシンごとに異なる値が付けられており、インターネット上に同じIPアドレスを持ったマシンが複数台存在することはありません。

もっとも、Webブラウズの際にわざわざIPアドレスを指定することはほ

とんどありません。たとえば、

```
http://www.ascii.co.jp/
```

というようなドメイン名で指定します。ユーザーにとってはこのほうが、

```
210.140.231.23
```

のような数字よりも理解しやすいからです。このIPアドレスとドメイン名を相互に変換してくれるのが、別項で説明しているDNS（Domain Name Service）です。

グローバルとプライベート

IPアドレスには、グローバルアドレスとプライベートアドレスがあります。インターネット上で利用できるのは、グローバルアドレスのほうです。

それに対して、LAN内部に限定して用いるのがプライベートアドレスです。**表1**に示す範囲のアドレスが、プライベートアドレス用に予約されています。今回の特集のモデル環境でも、LAN内部のマシンにプライベートアドレスが割り振られていることがわかるでしょう。

インターネット上を行き交うパケットには、送信元（ソース）と受信先（デスティネーション）のIPアドレスが含まれていますが、このうちのどちらかがプライベートアドレスのパケットは、インターネット上を通ることができません。ですから、このままではLAN内部のマシンが外部のマシンと直接パケットをやりとりすることはできません。そこで、あとで述べる方法で、IPマスカレードを有効にする必要があります。

10.0.0.0～10.255.255.255
172.16.0.0～172.31.255.255
192.168.0.0～192.168.255.255

表1　プライベートアドレスに割り当てられているIPアドレスの範囲

ipchainsコマンド

ipchainsは、Linuxカーネルに備わっているパケットフィルタリングの機能を利用して、簡単なファイアウォールとしての機能を実現するコマンドです。

ipchainsが用いるルールは、チェイン（chain）と呼ばれます。チェインには、

input（入力）
output（出力）
forward（転送）
ユーザー定義

の4種類があります。

各チェインには、パケットフィルタリングの規則と、ターゲットを指定します。ターゲットの値と意味は、表2の6種類です。ACCEPTとREJECT（DENY）は、ファイアウォール機能のために、MASQはIPマスカレード機能で用いられます。フィルタリングされたパケットを、ターゲットに結び付ける（チェイン）というイメージでとらえるといいでしょう。

IPマスカレード

IPマスカレードを行うためには、Linuxカーネルが持つ「パケットフォワード」の機能を有効にする必要があります。パケットフォワードとは、複数のネットワークインターフェイスの

ターゲット	動作
ACCEPT	パケットを通過させる
DENY	パケットを捨てる
REJECT	パケットを捨て、送信元にICMPメッセージを送る
MASQ	IPマスカレード（IPアドレスの変換）を行う（forward、ユーザー定義チェインのみ）
REDIRECT	外部ホスト宛のパケットをローカルのホストに振り向ける（input、ユーザー定義チェインのみ）
RETURN	ルールの最後を示す

表2　ipchainsのターゲット

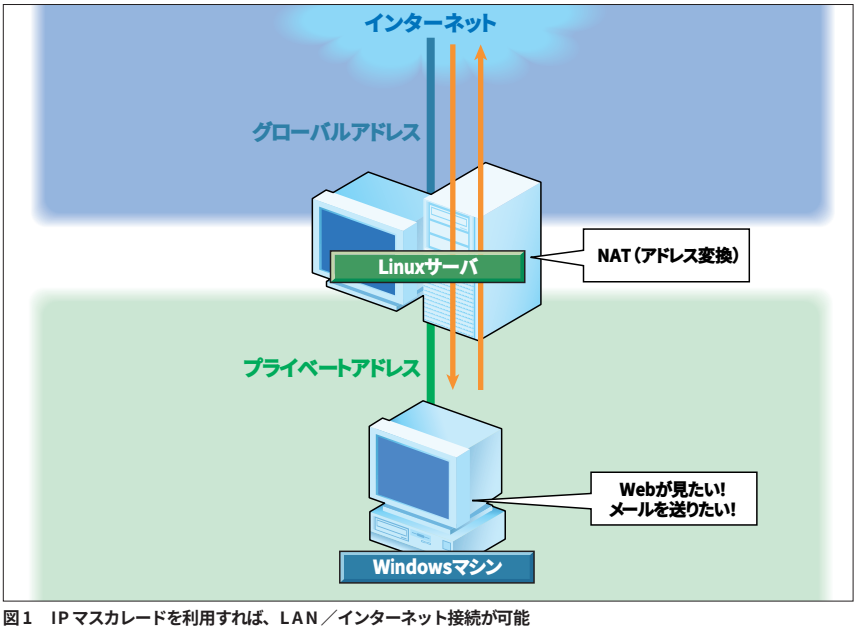


図1　IPマスカレードを利用すれば、LAN／インターネット接続が可能

間で、パケットを転送する機能のことです。

今回想定しているLinuxサーバは、LAN側にイーサネットカードがあり、インターネット側にもう1つ別のインターフェイスを持っていて、この間での転送が行われます。インターネット側のインターフェイスには特に制限はなく、アナログモデムやISDNならシリアルポートが、ケーブルモデムやADSLモデムなら2つめのイーサネットカードが用いられます。

■パケットフォワードを有効にする

フォワードを有効にするには、いくつかの方法がありますが、サーバ上でX Window Systemが動作しているなら、Network Configuratorを使うといいでしょう。rootユーザーになって、

```
# netcfg
```

と入力すると、**画面1**のウィンドウが現れます。“Routing”のボタンを押し、画面上部にある“Network Packet Forwarding (IPv4)”項目にチェックを入れます。マシンの再起動後には、パケットフォワードが有効になります。

サーバ上でX Window Systemを動かしたくない場合には、設定ファイルをエディタで直接変更します。パケットフォワードを指定するのは、

```
/etc/sysctl.conf
/etc/sysconfig/network
```

のいずれかです。/etc/sysctl.confの場合は、

```
net.ipv4.ip_forward = 0
```

という行を見つけて、行末の「0」を「1」に変更します。/etc/sysctl.confは、Red Hat Linuxの7.x以降に導入された設定ファイルであり、これより古いディストリビューションを利用している場合には、/etc/sysconfig/networkを利用します。

```
FORWARD_IPV4=false
```

という行の“false”を“true”に変更します。いずれの場合も、変更後にマシンを再起動すると、パケットフォワードが有効になります。

■IPマスカレードを有効にする

Red Hat Linux 7.1では、IPマスカレードを行うためのコードは、ローダブルモジュール形式になっているので、以下のように入力してモジュールをロードしておきます。

```
# modprobe ipchains
```

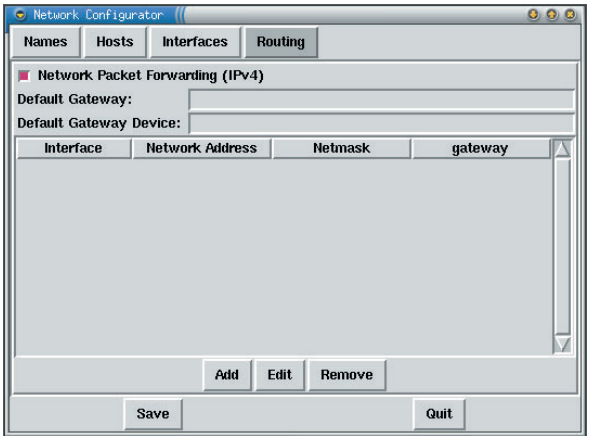
その後、**リスト1**のコマンドを入力することで、IPマスカレードが有効になります。**リスト1**の各オプションは、“-A”はチェインの追加、“-s”と“-d”はパケットの発信元と受信先の指定、“-j”がターゲットを示しています。つまり全体の意味としては、「発信元が192.168.0.0～192.168.0.255で、受信先が任意のIPアドレスになっているパケットは、転送（forward）時にIPマスカレード（MASQ）される」というチェインを追加していることになります。

マシン起動時に、IPマスカレードが有効になるように、上述のmodprobeコマンドと**リスト1**は、/etc/rc.d/rc.localの末尾に足しておきましょう。

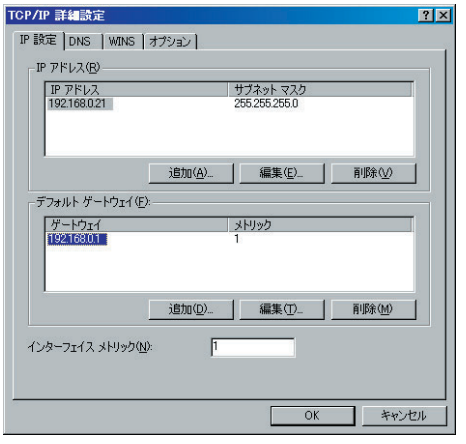
最後に、クライアントマシンのデフォルトゲートウェイをサーバマシンに設定しておきましょう（**画面2**）。

▼リスト1　IPマスカレードを有効にする

```
ipchains -A forward -s 192.168.0.0/24 -d 0.0.0.0/0 -j MASQ
```



画面1
Network Configuratorを用いて、パケットフォワードを有効にする。



画面2
クライアントマシンのデフォルトゲートウェイは、サーバマシン（192.168.0.1）にしておく。

パケットフィルタリング

パケットフィルタリングを正しく設定することは、セキュリティ上の観点から、とても重要なことです。しかし、何もない状態から適切なフィルタリング規則を作り上げるのは、なかなか大変です。そこで今回は、Red Hat Linux 7.1に備わっているファイアウォール設定ツール“lokkit”を使ってみましょう。Red Hat Linux 7.1のインストール中にファイルウォールの設定を尋ねられたはずです。それがlokkitです。

コマンドラインからlokkitを起動すると、**画面3**左のような画面が表示されます。ここでは“High”、“Medium”、“No firewall”という3通りのセキュリティレベルが選択できます。

“High”を選択すると、外部へのTCPの接続要求とDNSのやりとりのみが可能になります。非常に安全な状態ではありますが、サービスの提供がいっさい行えないので、サーバとは呼べません。

“Medium”を選ぶと、1023番以下のポート（Well-knownポート）、NFS、Xサーバ、Xフォントサーバへの接続が遮断されます。

また、“No firewall”では、/etc/sysconfig/ipchainsファイルそのものが作られないので、フィルタリングがまったく行われません。いまどき、インターネットと接続するマシンがノーガードというのは、許されません。

結局“Medium”を選択して、さらにカスタマイズするのがもっとも効率が良いでしょう。“カスタマイズ”ボタンを押すと、代表的ないくつかのサービスを選択できる画面に切り替わります（**画面3右**）。ここでいくつかのサービスへの接続を許可した結果、作成された/etc/sysconfig/ipchainsをリスト2に示します。

1～3行めは、入ってくるパケット、出ていくパケット、転送されるパケットすべてについて、デフォルトで許可するようにされています。

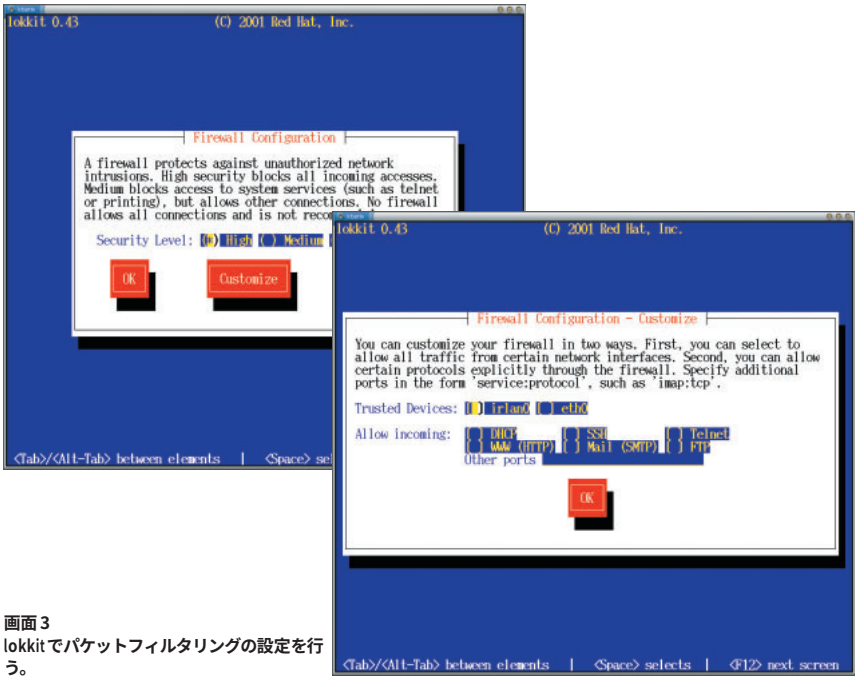
4行め以降は、1行がそのままipchainsコマンドの引数になります。たとえば4行目は、22番のポートを利用するSSHの接続要求の受け入れを許可しています。

/etc/sysconfig/ipchainsの内容を、サーバマシン起動時に設定するためには、以下のコマンドを実行します（サーバのランレベルが3の場合）。

```
# chkconfig --level 3 ipchains on
```

ここがポイント！

- **IPマスカレードとパケットフィルタリングはわけて考えましょう**
- **lokkitの設定はMediumをカスタマイズ！**



画面3
lokkitでパケットフィルタリングの設定を行う。

▼リスト2　カスタマイズされたフィルタリングルール

```
:input ACCEPT
:forward ACCEPT
:output ACCEPT
-A input -s 0/0 -d 0/0 22 -p tcp -y -j ACCEPT
-A input -s 0/0 -d 0/0 25 -p tcp -y -j ACCEPT
-A input -s 0/0 -d 0/0 80 -p tcp -y -j ACCEPT
-A input -s 0/0 -d 0/0 21 -p tcp -y -j ACCEPT
-A input -s 0/0 67:68 -d 0/0 67:68 -p udp -i eth0 -j ACCEPT
-A input -s 0/0 67:68 -d 0/0 67:68 -p udp -i eth1 -j ACCEPT
-A input -s 0/0 -d 0/0 -i lo -j ACCEPT
-A input -p tcp -s 0/0 -d 0/0 0:1023 -y -j REJECT
-A input -p tcp -s 0/0 -d 0/0 2049 -y -j REJECT
-A input -p udp -s 0/0 -d 0/0 0:1023 -j REJECT
-A input -p udp -s 0/0 -d 0/0 2049 -j REJECT
-A input -p tcp -s 0/0 -d 0/0 6000:6009 -y -j REJECT
-A input -p tcp -s 0/0 -d 0/0 7100 -y -j REJECT
```

細い回線でも快適にインターネットを利用できるプロキシサーバ

■RPM
squid

■設定ファイル
/etc/squid/squid.conf

■サービス名
squid

最近、ADSLやCATV、光ファイバなど、いわゆるブロードバンドなインターネット環境が急速に普及しつつあります。しかし、日本全国でそのようなサービスが提供されるには、まだ時間がかかるため、アナログモデムやISDNを用いた環境でインターネットに接続する場合もあるでしょう。プロキシサーバを用いれば、このような環境でのWebアクセス速度を改善できます。

“proxy”とは、「代理」という意味で、クライアントマシンの代わりにWebページにアクセスし、その結果をキャッシュしてメモリやディスクに保存しておきます。その後、クライアントマシンが同じWebページにアクセスした際には、まずそのページが更新されているかどうかを確認し、更新されていない時は、キャッシュしてあるデータをクライアントに渡します。LAN内のデータ転送は、インターネットの先からデータを取得してくるより高速なので、結果としてWebアクセスのレスポンスが向上します（**図1**）。

上記のキャッシュは、複数のクライアント間でも有効なので、数人で同じ回線を利用しているSOHO環境や、マシンやOSを複数利用しているユーザーにも有用です。Webブラウザが持っているキャッシュを、複数のOSやマシンで共有できると思えば良いでしょう。

Squid

Squidは、UNIX系のOSで用いられるプロキシサーバの代表です。現在のところ、2.4系列が最新の安定版（STABLE）、2.5系列が開発版（DEVEL）となっています。今回利用するのは、Red Hat Linux 7.1に含まれているsquid 2.3 STABLE4です。系列が変わると、設定ファイルの項目や書式も変わるため、まだ2.3系列を用いているディストリビューターが多いようです。2.4／2.5系列を利用するには、本家のWebサイト（http://www.squid-cache.org/）からtarボールを取得し、自分でビルドする必要があります。

Squidの設定

Squidの設定ファイル群は、/etc/squidディレクトリ以下に置かれています。その中でユーザーが変更する必要があるのは、squid.confです。2000行以上の巨大なファイルですが、大部分は行頭に“#”を付けたコメントとなっており、設定項目の説明とデフォルトの設定値が記されています。

ほとんどの項目は、デフォルトのままで利用しても問題はありませんが、アクセス制限に関する項目だけは、修正が必要です。アクセスの制限は、以下の形式で記述します。

```
http_access allow/deny リスト名
```

「リスト名」の定義は、以下の形式で前もって作成しておきます。

```
acl リスト名 タイプ データ…
```

デフォルトでは、Linuxサーバ自身からのみ利用できるように設定されていますので、これをクライアントからも使えるように変更しましょう。まず/etc/squid/squid.confをエディタで開き、アクセスリストが定義されている場所を表示します。そして、今回の特集で想定しているローカルネットワークを“mynet”という名前で定義

します。“ acl CONNECT method CONNECT ”と書かれた行の次に、以下の行を追加します。

```
acl mynet src 192.168.0.0/255.255.255.0（実際は1行）
```

次に“ http_access deny all ”と書かれた行の前に、以下の1行を追加します。

```
http_access allow mynet
```

ファイルを保存後、起動スクリプトを利用して、Squidを起動します。

```
# /etc/init.d/squid start
```

サーバマシン起動時にSquidを自動的に実行するためには、以下のコマンドを実行します（サーバのランレベルが3の場合）。

```
# chkconfig --level 3 squid on
```

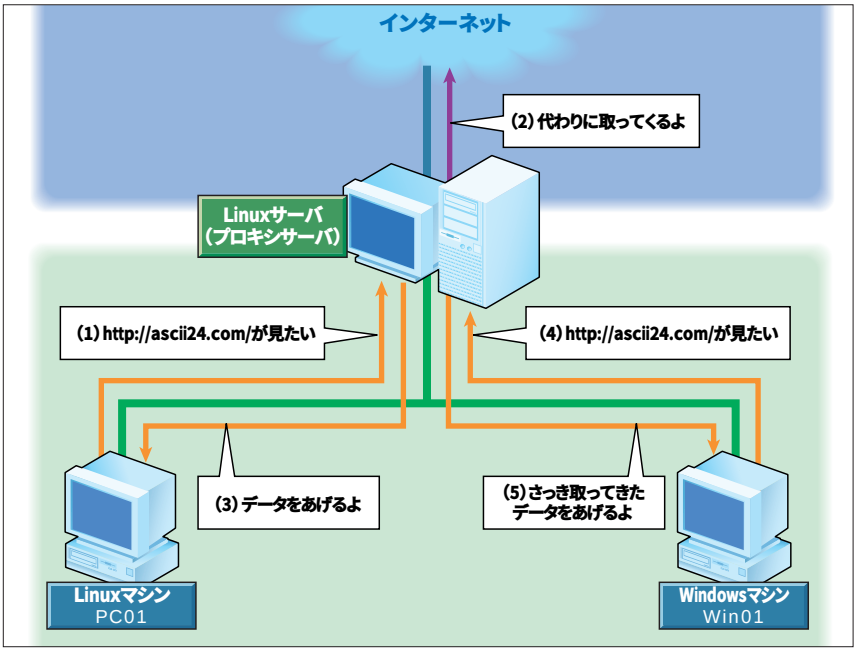


図1　プロキシサーバの仕組み

Webブラウザの設定

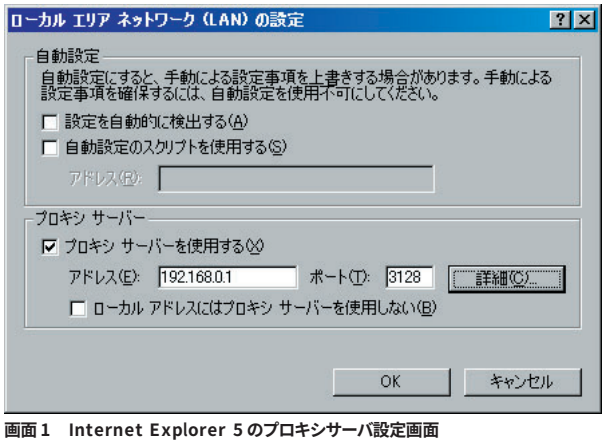
プロキシサーバを利用したWebアクセスを行うためには、Webブラウザ側でも設定が必要です。

Windows の Internet Explorer では、[ツール]メニューから[インターネット オプション]を選択し、[接続]タブを開いて[LANの設定]ボタンを押します（**画面1**）。「プロキシサーバーを使用する」にチェックを入れ、サーバマシンのIPアドレスとポート番号（ここではデフォルトの3128）を指定し、[OK]ボタンを押します。Linux版のMozillaでも、設定項目はほぼ同じです（**画面2**）。

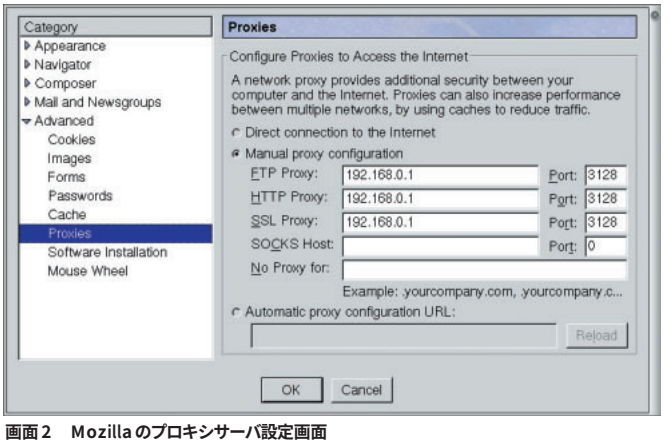
以上の設定で、Squidが利用可能になります。性能に関するチューニングは、まったく行っていませんが、家庭内LANやSOHOクラスのネットワークなら、デフォルトの設定でこと足りると思われます。最近はメモリが安くなってきたので、大量に搭載してメモリ上のキャッシュのサイズ（cache_mem）などを調整することで、より快適になるかもしれません。

ここがポイント！

- **/etc/squid/squid.confでアクセス制限を設定しましょう**
- **Webブラウザの設定も必要です**



画面1　Internet Explorer 5のプロキシサーバ設定画面



画面2　Mozillaのプロキシサーバ設定画面

ホスト情報を自動的に取得するDHCPサーバ

■RPM
dhcp

■設定ファイル
/etc/dhcpd.conf
/var/lib/dhcp/dhcpd.leases

■ネットワーク名
dhcpd

ネットワークにTCP/IPで接続する機器（クライアント）には、IPアドレス、DNSサーバアドレス、デフォルトゲートウェイ、ネットマスクなど、さまざまなネットワーク情報（ホスト情報）を設定しなければなりません。これらのホスト情報は機器ごとに設定する必要があり、ネットワークに接続する機器が多くなると設定作業が大変になるばかりか、管理自体ができなくなってしまいます。

そこである程度の規模のネットワークでは、接続する際に必要となるホスト情報をサーバに問い合わせ、クライアント側の設定を自動的に行うDHCP（Dynamic Host Configuration Protocol）という仕組みを利用します。DHCPを利用すれば、ネットワークの変更があった場合でもサーバ側のみ変更すればよく、クライアント側を変更する必要がありません。さらに、IPアドレスを自動的に割り当てることができますので、IPアドレスが重複したりする問題が回避できます。こうしたことから、ネットワーク管理者とユーザーの負担を大きく軽減できるというメリットがあります。

DHCPサーバの設定

DHCPサーバを構築する前に、ホスト情報を整理してみましょう。

ネットワークに接続する機器に最低限設定しなければならないホスト情報は**表1**のようなものです。

IPアドレスの割り当ての範囲が、192.168.0.2～192.168.0.254となっているのは、192.168.0.1がサーバ（server01）のIPアドレスとしてすでに使用されているためです。また、192.168.0.0はネットワークアドレス、192.168.0.255はブロードキャストアドレスとして使用されるため、それぞれIPアドレスの割り当て範囲から除外します。

必要なホスト情報が整理できましたので、DHCPサーバを設定してみましょう。DHCPサーバの設定ファイルは、/etc/dhcpd.confファイルです。

表1の情報を元に作成したdhcpd.conf は**リスト1**のようになります。

リスト1の1行めでは、ネットワークアドレスとネットマスクを指定しています。

続く2行めでは、IPアドレスを割り当てる範囲を指定しています。前述したように192.168.0.2から割り当てているのは、192.168.0.1がすでにDHCPサーバに割り当てられているためです。また、192.168.0.0と192.168.0.255も除外しています。

最後の3行目ではデフォルトゲートウェイを指定しています。

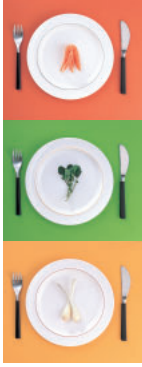
設定ファイルを作成したら、DHCPサーバが割り当てたIPアドレス情報などを記録しておくデータベースファイルを作成します。このファイルはDHCPサーバ自体が使うもので、設定の必要はありません。単に、空のファイルを作成するだけです。

```
# touch /var/lib/dhcp/dhcpd.leases
```

以上で設定は終了です。

項目	ネットワーク情報
ネットワークアドレス（サブネット）	192.168.0.0
ネットマスク	255.255.255.0
IPアドレス割り当ての範囲	192.168.0.2～192.168.0.254
デフォルトゲートウェイ	192.168.0.1

表1　サンプルネットワークに必要なネットワーク情報



設定が完了したら、DHCPサーバを起動してみましょう。

```
# /etc/init.d/dhcpd start
```

もし、DHCPサーバが正常に起動できない場合は、設定ファイルに記述ミスがないか確認してみましょう。設定ファイルにミスがないかどうかを確認するには、次のように入力します。

```
# /usr/sbin/dhcpd -d -f
```

もし、設定ミスがあった場合はエラー箇所が表示されます。

設定が問題なければ、システム起動時にDHCPサーバが起動するようにchkconfigコマンドを使って設定しておきます。

```
# chkconfig dhcpd on
```

クライアント側のIPアドレスは、192.168.0.2～192.168.0.254の中から使用されていないものが自動的に割り当てられます。

より高度な設定

リスト1では最低限の設定しか行っていませんが、DHCPサーバではさらに多くのホスト情報を提供することができます。たとえば、DNSサーバのアドレスや、WINSサーバのアドレスを提供することも可能です。また、特定の機器に固定のIPアドレスを割り当てることもできます。

リスト2は、より多くのホスト情報を提供するdhcpd.confです。

最初の3行はDHCPサーバの名前、ドメイン名、DNSサーバのアドレスを指定しています。このように、subnetで始まる{ }のブロック外にある設定はデフォルトの設定となります。一方、subnetで始まる{ }で囲まれたブロック内にある設定は、そこで指定されたネットワーク（サブネット）内でのみ有効な設定となります。つま

▼リスト1　最低限のdhcpd.confファイル

```
subnet 192.168.0.0 netmask 255.255.255.0 {          ネットワークアドレスとネットマスク
        range 192.168.0.2 192.168.0.254;            IPアドレス割り当ての範囲
        option routers 192.168.0.1;                 デフォルトゲートウェイ
}
```

▼リスト2　一般的なdhcpd.confファイル

```
server-identifier server01.example.com;             DHCPサーバの名前
option domain-name "example.com";                   ドメイン名の指定
option domain-name-servers 192.168.0.1;             DNSサーバの指定

subnet 192.168.0.0 netmask 255.255.255.0 {
        range 192.168.0.2 192.168.0.254;
        option domain-name "pb.example.com";        サブドメインの指定
        option routers 192.168.0.1;                 デフォルトゲートウェイの指定
        option subnet-mask 255.255.255.0;           ネットマスクの指定
        option broadcast-address 192.168.0.255;     ブロードキャストアドレスの指定
        option netbios-name-servers 192.168.0.1;    WINSサーバの指定
        default-lease-time 86500;                   リースタイムの設定
        max-lease-time 604800;
        host pc01 {                                 固定IPアドレス割り当ての設定
                hardware ethernet 00:03:ff:85:f8:f0;
                fixed-address 192.168.0.11;
        }
}
```

り、複数のネットワークがある場合は、ネットワークの数だけsubnetで始まる{ }で囲まれたブロックがあり、それぞれのネットワークごとの設定を記述します。たとえば、ネットワークAではDNSをserver01、ネットワークBのDNSはserver02などと設定することができるわけです。ここでは1つのネットワークしか存在しませんので、どちらに記述しても意味は一緒です。

14～17行めにあるhostで始まるブロックは、IPアドレスを固定するための設定です。ここでは、00:03:ff:85:f8:f0というMACアドレス（ハードウェアアドレス）を持つ機器に、IPアドレス192.168.0.11を「固定」で割り当てるように設定しています。このMACアドレスは、ネットワークカードに割り振られた固有の番号で、ネットワークカードごとに異なります。

設定を変更した場合は、DHCPサーバを再起動する必要があります。

```
# /etc/init.d/dhcpd restart
```

なお、プロバイダとのルータとして設置したサーバでDHCPを稼働させる場合は、プロバイダ側にDHCPを公開しないようにネットワークインターフェイスを指定する必要があります。ネットワークインターフェイスを指定する場合は、/etc/init.d/dhcpdスクリプト内の、dhcpdデーモンを起動している部分を**リスト3**のように変更します。

クライアントの設定——Linux

LinuxマシンをDHCPクライアントとして設定する場合は、pumpかdhcpcdのどちらかがインストールされている必要があります。

DHCPクライアントの設定は、/etc/sysconfig/network-script/ifcfg-eth0ファイルを**リスト4**のように記述します。**リスト4**は、eth0を起動時に有効にし、DHCPサーバからホスト情報を取得するという設定です。

設定が完了したら、次のようにしてネットワークを再起動します。

```
# /etc/init.d/network restart
```

▼リスト3　/etc/init.d/dhcpd

```
daemon /usr/sbin/dhcpd eth0
```

▼リスト4　LinuxをDHCPクライアントにする設定（/etc/sysconfig/network-script/ifcfg-eth0）

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

DHCPサーバからホスト情報を取得する設定

書式	内容
default-lease-time ＜seconds＞	クライアントが指定しない場合のIPアドレスの貸し出し時間（秒）
host ＜host name＞	固定IPを割り当てるホスト名
hardware ethernet ＜FF:FF:FF:FF:FF:FF＞	固定IPアドレスを割り当てる対象となる機器のMACアドレス
fixed-address ＜XXX.XXX.XX.XXX＞	割り当てる固定IPアドレス
max-lease-time ＜seconds＞	IPアドレスの最大貸し出し時間（秒）
option broadcast-address ＜XXX.XXX.XXX.XXX＞	ブロードキャストアドレス
option domain-name ＜domain name＞	ドメイン名。サブドメイン名を記述することも可能
option domain-name-server ＜domain name server＞	DNSサーバ名またはDNSサーバのIPアドレス
option netbios-name-servers ＜wins server＞	WINSサーバのアドレス
option routers ＜default router＞	デフォルトゲートウェイの指定
option subnet-mask ＜XXX.XXX.XXX.XXX＞	サブネットマスク
range ＜SSS.SSS.SSS.SSS＞ ＜EEE.EEE.EEE.EEE＞	IPアドレスの範囲（SSS.SSS.SSS.SSS～EEE.EEE.EEE.EEE）
server-identifier ＜server-name＞	DHCPサーバの名称
subnet ＜SSS.SSS.SSS.SSS＞ netmask ＜NNN.NNN.NNN.NNN＞	ネットワークアドレス（SSS.SSS.SSS.SSS）とネットマスク（NNN.NNN.NNN.NNN）の指定

表2　dhcpd.confで指定できる内容

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:03:FF:85:F8:F0
          inet addr:192.168.0.11  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:11 Base address:0x1080
```

画面1　ホスト情報の確認（ifconfigコマンドの実行）

正常にホスト情報が取得できているかどうかを確認するにはifconfigコマンドを実行します（**画面1**）。

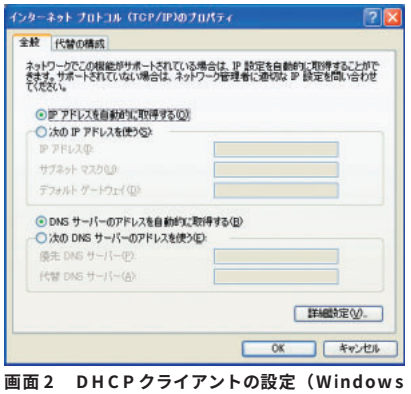
クライアントの設定——Windows

Windowsマシンの場合は、Windowsのバージョンによって設定方法が少しずつ違います。ここでは、Windows 2000、XP、Meでの設定方法を解説します。

■Windows 2000

Windows 2000の場合は、「インターネット プロトコル（TCP/IP）のプロパティ」ダイアログボックスで設定します。設定方法は次の手順です。

①コントロールパネルを開く
②［ネットワークとダイヤルアップアダプタ接続］アイコンをダブルクリック
③［ローカル エリア接続］アイコンをダブルクリック
④［ローカル エリア接続 状態］ダイアログボックスの［プロパティ］をクリック
⑤［インターネット プロトコル（TCP/IP）］を選択して、［プロパティ］をクリック
⑥「インターネット プロトコル（TCP/IP）のプロパティ」ダイアログボックスで、［IP アドレスを自動的に取得する］と［DNSサーバーのアドレスを自動的に取得する］をチェックし、［OK］をクリックする

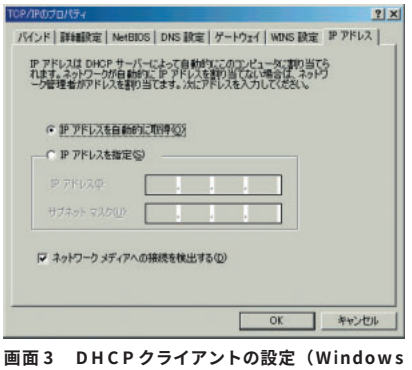


画面2　DHCPクライアントの設定（Windows XP）

■Windows XP

Windows XPの場合も、Windows 2000の設定方法とほぼ同じです。しかし、「インターネット プロトコル（TCP/IP）のプロパティ」ダイアログボックスの表示方法が若干違いますので注意してください。

①コントロールパネルを開く
②［ネットワークとインターネット接続］アイコンをクリック
③［ネットワーク］アイコンをクリック
④［ローカル エリア接続］アイコンをクリック
⑤［ローカル エリア接続 状態］ダイアログボックスの［プロパティ］をクリック
⑥［インターネット プロトコル（TCP/IP）］を選択して、［プロパティ］をクリック
⑦「インターネット プロトコル（TCP/IP）のプロパティ」ダイアログボックスで、［IP アドレスを自動的に取得する］と［DNSサーバーのアドレスを自動的に取得する］をチェックし、［OK］をクリックする（画面2）



画面3　DHCPクライアントの設定（Windows Me）

■Windows Me

Windows Meの場合は、「TCP/IPのプロパティ」ダイアログボックスで設定します。設定方法は次の通りです。

①デスクトップにある、「マイ ネットワーク」アイコンを右クリック
②「ネットワーク」ダイアログボックスで、［TCP/IP-><ネットワークカード名>］を選択し、［プロパティ］をクリック
③「TCP/IPのプロパティ」ダイアログボックスの［IP アドレス］タブにある、［IP アドレスを自動的に取得］をチェックし、［OK］をクリックします（画面3）。

■ホスト情報の確認

正常にホスト情報が取得できているかどうかを調べるには、Windows 2000／XPでは、コマンドプロンプトでipconfigコマンドを実行します。

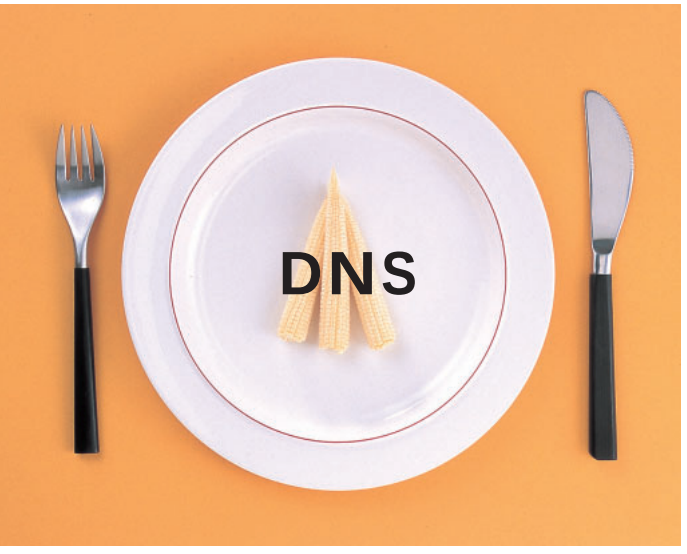
```
C:¥>ipconfig /all
```

Windows Meでは、MS-DOSプロンプトで次のように実行します。

```
C:¥WINDOWS>winipcfg
```

ここがポイント！

- **DHCPサーバ自身はホスト情報を受け取ることができませんので、固定のIPアドレスを指定する必要があります。**
- **dhcpd.confファイルの設定では、各行の最後に「；（セミコロン）」が必要です。セミコロンの付け忘れには注意しましょう。**



名前の解決を行う分散データベースDNSサーバ

■RPM
bind
bind-utils
■設定ファイル
/etc/named.conf
/etc/resolv.conf
/etc/host.conf
/var/named/
■ネットワーク名
named

DNS（Domain Name System）はネットワーク上の名前を解決するための仕組みです。TCP/IPで接続される機器は、すべてIPアドレスを持っています。このIPアドレスは111.122.133.144といった数字の羅列で表現されます。しかし、数字の羅列では人間にとって覚えにくく、どんな機器なのか想像もできません。そこでこのIPアドレスに、pc01.example.comといった人間が覚えやすく、イメージしやすいアドレスを割り当てて相互に変換するようにしています。この変換のことを名前の解決といいます。

名前の解決は携帯電話のアドレス帳をイメージするとわかりやすいでしょう。アドレス帳に名前と電話番号を登録しておけば、電話番号を覚えていなくても名前で電話をかけることができるのと同じです。

このように、DNSサーバは非常に便利なもので、インターネットの根幹をなす重要なシステムです。

DNSの仕組み

DNSサーバはインターネット上の巨大な分散データベースです。DNSはツリー構造になっており、ルートである「.」の下に階層的にDNSサーバが存在します。インターネットに接続しているドメインは、外部から要求のあった自ドメイン以下の名前解決を担当します。それ以外の名前の解決は、ほかのDNSサーバへ問い合わせします。

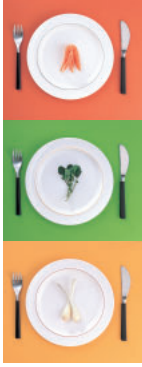
DNSによる名前の解決は2種類あります。1つは、アドレス（ホスト名）から数字のIPアドレスへ変換する正引き（順引き）で、もう1つは数字のIPアドレスからホスト名へ変換する逆引きです。人間が使う場合はほとんど正引きですが、コンピュータにとっては逆引きも必要です。

しかし、今回のようにLAN内に設置されるDNSサーバは、外部から参照される必要がありません。そこで、LAN内のホストの名前の解決と、外部への名前の問い合わせだけを行うように設定します。

hostsファイルとDNS

名前の解決を行う同様の仕組みとして、hostsファイルがあります。こちらもDNSと同じように名前の解決を行うためのデータベースですが、DNSのように分散システムにはなっていません。hostsファイルを各ホストに用意し、それぞれのホストで名前の解決を行います。

しかし、機器が変更されるたびに、すべてのホストのhostsファイルを更新しなければならないなどメンテナンスが大変なため、大規模なシステムでの運用に限界があります。そこで、分散データベースであるDNSが開発されました。しかし、小規模なLAN内での運用であればhostsファイルでも十分実用的です。DNSサーバを構築するかhostsファイルで運用するかはそのシステムの運用形態次第ですが、DNSサーバはDHCP同様、ホスト情報をサーバ側で管理できるため、結果的にはシステム管理者とユーザーの負担を軽減させることができます。また、DNSは問い合わせされた名前を一定期間キャッシュする機能がありますので、外部の名前の解決の高速化と、ネットワークトラフィックが軽減できるというメリットもあります。さらに、DNSはhostsファイルと相互運用できるようになっていますので、LAN内の名前の解決はhostsファイルで、外部の名前解決にはDNSといった使い



方も可能です

BINDの設定

DNSサーバには、BINDというプログラムが多く使われています。BINDでは、BIND自体の設定ファイルと、ホストのアドレス情報を記載したゾーンファイル（DNSデータベース）を作成します。なお、DNSサーバの設定を誤ると、プロバイダや外部のネットワークに迷惑をかける可能性がありますので、十分注意して設定しましょう。

■BINDの設定ファイル

BINDの設定ファイルは/etc/named.confファイルです。

named.confファイルでは、DNSが管理するネットワーク（ゾーン。サブネットと同等）の指定、ゾーンファイルの指定、DNS自体の動作の指定を行います。

ゾーンファイルは、名前の解決を行うためのデータベースファイルで、ホスト名とIPアドレスをどのように割り当てるかを定義したファイルです。

リスト1の最初の「options」セクションでは、ゾーンファイルが置かれているディレクトリと、forwardersを指定しています。

forwardersは、自ドメイン内のDNSサーバで名前の解決ができなかった場合に、最初に問い合わせる外部のDNSサーバの指定です。ここにはプロバイダのDNSサーバのIPアドレスを記述します。続く、forward onlyはforwardersで名前の解決ができなかった場合（つまり、プロバイダのDNSサーバで名前の解決ができなかった場合）、ほかのDNSサーバに問い合わせないための指定です。プロバイダのDNSサーバで解決できない名前は、ほかのDNSに問い合わせてもほとんど意味がありませんし、LAN内のDNSサーバがルートDNSに問い合わせするのを避けるためです。

次のzone "."で始まるセクションはルートDNSサーバの指定です。このDNSサーバはルートDNSへの問い合わせは行いませんが、形式上記述してあります。

次のzone "example.com"で始まるセクションが正引き用の設定です。ここでは、正引き用のゾーン名とゾーンファイルの名前の指定、問い合わせに関する設定を行っています。

notify noは、ゾーンに変更があった場合でもセカンダリDNSサーバに通知しないための設定です。ここではセカンダリDNSサーバを構築していませんので通知は不要です。

▼リスト3　LinuxをDHCPクライアントにする設定（/etc/sysconfig/network-script/ifcfg-eth0）

```
options {
        directory "/var/named";
        forwarders {
                111.122.133.144;
        };
        forward only;
};

zone "." {
        type hint;
        file "named.ca";
};

zone "example.com" {
        type master;
        file "example.com";
        notify no;
        allow-transfer {
                192.168.0/24;
                127.0.0.1;
        };
        allow-query {
                192.168.0/24;
                127.0.0.1;
        };
};

zone "0.168.192.IN-ADDR.ARPA" {
        type master;
        file "192.168.0";
        notify no;
        allow-transfer {
                192.168.0/24;
                127.0.0.1;
        };
        allow-query {
                192.168.0/24;
                127.0.0.1;
        };
};

zone "0.0.127.IN-ADDR.ARPA" {
        type master;
        file "named.local";
};
```

ゾーンファイルの保存ディレクトリ
プロバイダのDNSサーバのIPアドレス
ほかのDNSサーバへの問い合わせ禁止
ルートDNSサーバの指定
正引き用の設定
逆引き用の設定
ローカルループバックの設定

allow-transferと、allow-queryは、それぞれ転送許可と問い合わせ許可の設定です。ゾーン内とローカルホストにのみ許可しています。この設定をきちんとしないと、LAN内の情報が外部へ漏れてしまいますので気をつけてください。

その次のzone"0.168.192.IN-ADDR.ARPA"で始まるセクションは逆引き用の設定です。この0.168.192は、ゾーンのIPアドレスを逆に指定したもので、IN-ADDR.ARPAは逆引きするための指定です。このセクションのそのほかの部分は正引き用の設定と同じです。

最後の、zone "0.0.127.IN-ADDR.ARPA" で始まるセクションは、ローカルループバック用の設定です。これはlocalhostの名前を解決します。

■正引き用ゾーンファイル

/var/named/example.comファイルは、正引き用のゾーンファイルです（**リスト2**）。通常は外部に公開する自ドメインのゾーン情報を設定しますが、今回のDNSサーバはLAN内のみにサービスを提供するためのものですので、内部用の設定のみを行っています。

最初の行はSOA（Start Of Authority）レコードと呼ばれ、ゾーンファイルを定義している部分です。「@」で始まっていますが、これは決まりのようなものでゾーン自体のアドレスに置換される置換記号です。

「SOA」のうしろの2つのアドレスは、1つめがDNSサーバの名前、2つめが管理者のメールアドレスです。2つとも、アドレスの最後に「.」が付いていますが、これはアドレスがFQDN（ Fully Qualified Domain Name）であるということを表しています。この表記方法は、絶対ドメイン指定とも呼ばれるもので、ルートドメインからの絶対位置を表すものです。なお、ここで指定するメールアドレスは、「@」部分を「.」で指定します。

その次の行はゾーンファイルのシリアル番号です。シリアル番号は、ゾーンファイルをアップデートした際に大きな数字に変更し、ほかのDNSサーバやネットワークにゾーンファイルが変更されたことを伝えるために利用されるものです。シリアル番号はどんなものでもかまいませんが、通常は重複しないように西暦＋月＋日＋3桁の数字などを指定します。

その次の4行は、ゾーン情報のキャッシュ期限などを指定しておくものです。今回は外部にDNSサーバを公開しないため、あまり気にする必要はありません。

その次の「server01」で始まる行が、server01のIPアドレスを指定している部分です。「IN　A」に続けてIPアドレスを指定しています。

次の「@　IN MX 10」は外部に公開するメールサーバを設定する部分ですが、今回は外部にメールサーバを公開しませんので形式的に登録しています。

次の行の「@　IN NS」はネームサーバの指定です。これら2つのアドレスはFQDNで記述していることに注意してください。

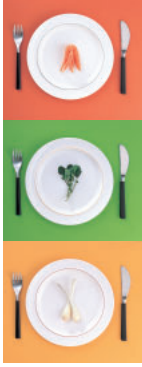
リスト2の残りの部分が実際に名前の解決を行うためのデータベースで、ホスト名とIPアドレスの対応を記述します。ここでは、pc01とpc02のIPアドレスをそれぞれ、192.168.0.11、192.168.0.12と定義しています。

▼リスト2　正引き用ゾーンファイル（/var/named/example.com）

```
@       IN SOA  server01.example.com    root.server01.example.com. (
                200102901       ; serial
                28800           ; refresh
                14400           ; retry
                3600000         ; expire
                86400           ; default ttl
        )

server01        IN A            192.168.0.1
@               IN MX 10        server01.example.com.
@               IN NS           server01.example.com.

pc01            IN A            192.168.0.11
pc02            IN A            192.168.0.12

localhost       IN A            127.0.0.1
```

▼リスト3　逆引き用ゾーンファイル（/var/named/192.168.0）

```
@       IN SOA  server01.example.com    root.server01.example.com. (
                200102901       ; serial
                28800           ; refresh
                14400           ; retry
                3600000         ; expire
                86400           ; default ttl
        )

1               IN PTR          server01.example.com.
@               IN NS           server01.example.com.

11              IN PTR          pc01.example.com.
12              IN PTR          pc02.example.com.3
```



▼リスト4　ローカルゾーンファイル（/var/named/named.local）

```
@       IN SOA  server01.example.com    root.server01.example.com. (
                200102901       ; serial
                28800           ; refresh
                14400           ; retry
                3600000         ; expire
                86400           ; default ttl
        )

1       IN      PTR     localhost.
@       IN      NS      localhost.
```

```
# host pc01
pc01.example.com has address 192.168.0.11

# host 192.168.0.12
192.168.0.12.in-addr.arpa. domain name pointer pc02.example.com
```

画面　名前解決の確認

■ルートキャッシュファイル

/var/named/local.caファイルは、ルートドメインのDNSサーバのキャッシュデータです。実際には、キャッシュデータではなくルートDNSサーバのアドレスが記述されているファイルで、通常は最新のファイルをインターネットから入手する必要がありますが、今回はルートDNSに対して問い合わせを行わないように設定していますので、ディストリビューションに付属しているlocal.caをそのまま使用します。

■逆引き用ゾーンファイル

/var/named/192.168.0は、逆引き用の設定ファイルです（**リスト3**）。ファイル名をLANのアドレスにしていますが、実際にはどんな名前でもかまいません（named.confで定義するため）。一般的には、どのネットワークのゾーンファイルなのかをわかりやすくするため、このような名前を付けています。

最初のブロックは正引き用のゾーンファイルと同じです。

逆引き用のゾーンファイルには、正引き用ゾーンファイルで設定したデータベースの逆の対応を記述します。正引き用ゾーンファイルと矛盾がないように記述する必要があります。

■ローカルゾーンファイル

/var/named/named.localはローカルループバック用のゾーンファイルです（**リスト4**）。ほとんどの部分はほかのゾーンファイルと同じで、127.0.0.1がlocalhostだという設定だけをしています。

■リゾルバの設定

BINDの設定が終わったら、リゾルバ（/etc/resolv.conf）の設定を行います。リゾルバは、ドメイン名やDNSサーバを指定するだけの簡単なものです。1行目でドメイン名を、2行目でDNSサーバのアドレス（自分自身）を指定しています（**リスト5**）。

■名前の解決の順序

最後に、host.confファイルを設定します。host.confファイルは、名前の解決をどのように行うかを指定するファイルです（**リスト6**）。

ここでは、まずhostsファイルを参照し、名前の解決ができなかった場合にDNSサーバに問い合わせを行うように設定しています。

▼リスト5　リゾルバの設定（/etc/resolv.conf）

```
domain example.com
namaserver 192.168.0.1
```

▼リスト6　名前解決の順序の設定（/etc/host.conf）

```
order hosts,bind
```

以上で設定は終了です。DNSサーバは外部のネットワークに迷惑をかける可能性があるので、もう一度設定ファイルが正しいかどうかを確認し、DNSサーバを起動しましょう。

```
# /etc/init.d/named start
```

正常に名前の解決ができているかどうかを確認するには、hostコマンドを使います。hostに続けて、ドメイン名またはIPアドレスを指定します。正常に名前の解決ができていれば、**画面**のように表示されるはずです。名前の解決が正常に行われていることが確認できたら、システム起動時にDNSサーバが起動するように設定しておきましょう。

```
# chkconfig named on
```

ここがポイント！

- **ゾーンファイルを変更した場合は、シリアルを変更するのを忘れないようにしましょう。また、変更した場合はDNSを再起動する必要があります。**
- **設定ファイルの「.」や「;」は重要です。ついつい忘れてしまいますので、十分注意しましょう。DNSサーバはほかのネットワークに迷惑をかける可能性が高いサーバです。DNSの設定や運用には細心の注意が必要です。**

実用Samba 2.2入門

Linuxで作るWindowsファイルサーバ

～前編～

Samba 2.2

複数のＰＣをＬＡＮで接続するときに、ファイルやプリンタの共有を行えると非常に便利だ。共有はWindowsｓ同士ではWindowsｓネットワークによって行い、Linux同士ではＮＦＳやＬＰＤを使用するのがふつうだ。このように標準で使われるネットワークプロトコルが違うため、LinuxとWindowsｓの間での共有は簡単ではなかった。しかし、Ｌｉｎｕｘ上でＳａｍｂａを動かすことでWindowsｓマシンとの間でも共有を行うことができる。今回はまず、Windowsｓネットワークの仕組みを理解し、そして最新版のＳａｍｂａで何ができるかを紹介する。実際のＳａｍｂａの設定方法は次号の後編でお届けする。

Samba運用のためのWindowsネットワーク入門

文：藤沢敏喜
Text :Toshiki Fujisawa

Windows 95の登場とともに本格的に導入されたマイクロソフトWindowsネットワークは、OSの標準機能として搭載された。これにより、クライアントPCにソフトウエアを追加することなくファイルやプリンタを共有することができるため、広く使われるようになった。クライアントマシンがWindows XPになったとしても、当分の間はこの共有方法が主流になると思われる。

一方ファイルを提供するサーバとしては、Windows NTやWindows 2000が使用されることが多かったが、最近では非常に安定した動作を誇るLinuxなどのPC-UNIXで、Sambaを動作させる事例が増えてきた。

Sambaサーバを安定して動作させるためには、当然のことながらマイクロソフトWindowsネットワークの挙動を詳しく知る必要がある。しかしながらUNIXユーザーの場合、ふだん自分ではWindowsを使っていないこともあり、UNIX側の設定方法はよく知っていても、クライアント側の設定方法やその動作はあまりよくわからないということも多いようである。

そこで、この記事ではマイクロソフトWindowsネットワークの基礎概念や、クライアントの設定、そしてトラブル時の解決方法など、Sambaの運用をするうえで知っておくべき、いくつかのトピックについて解説してみたい。

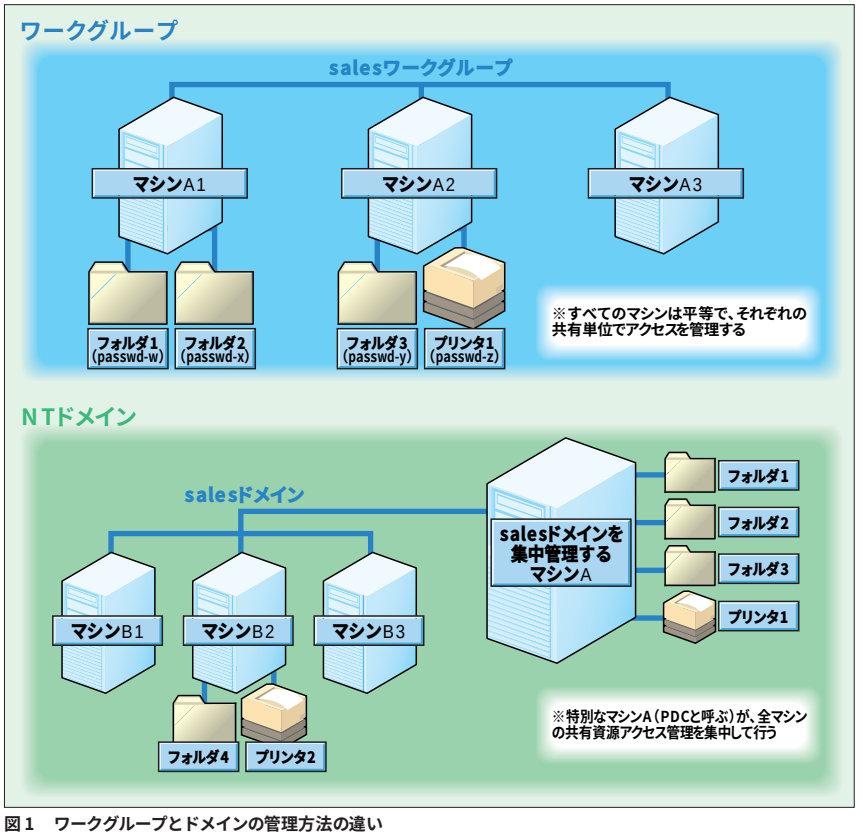


図1　ワークグループとドメインの管理方法の違い

ワークグループとNTドメイン

みなさんご存じのように、Windowsのデスクトップにある、ネットワークコンピュータのアイコンをクリックすると、ネットワークに接続されているコンピュータが表示される。表示されたコンピュータの数が数台であれば、グループ化する必要はないが、PCの数が数百台になると、たとえば設計部、製造部、販売部といった単位でコンピュータをグループごとに分類する必要が生じてくる。このためマイクロソフトWindowsネットワークでは、グループ化する手段として「ワークグループ」と「ドメイン」と呼ばれる2種類の方法を用意している。

「ワークグループ」では、アクセスの管理をそれぞれのマシンが個別に行うのに対し、「ドメイン」ではアクセス管理を1台の管理専用マシンで集中的に管理することが特徴となっている（**図1**）。

なお、ここでいうドメインとはメールアドレスやURLに使われるドメインとは、概念がまったく違う用語であり、大変紛らわしい。そのためこの記事では、このマイクロソフト用語のドメインを「NTドメイン」、普通のド

メインを「インターネットドメイン」と呼んで区別することにする。

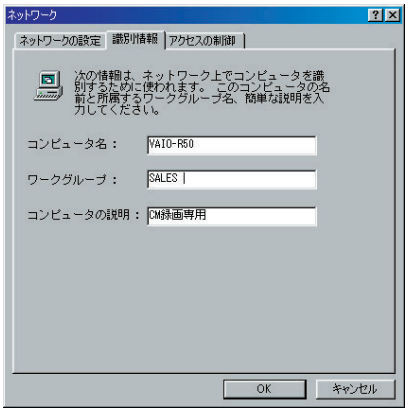
ワークグループ

ワークグループは、必ずしもサーバを必要としないグループ化の方法で、共有するフォルダやプリンタなどのリソース（資源）は、個々のクライアントコンピュータから直接参照される。このモデルでは、たとえば他人に共有させるためのフォルダなどの共有資源に対してパスワードを設定し、そのパスワードによってアクセスできるかどうかを決定している。この場合アクセス権限が共有ごとに行われることから、このアクセス方法は、「共有レベルのセキュリティ」と呼ばれる。なお、Sambaの設定ファイルsmb.confでは、この方式を使用する場合、

```
security = share
```

と設定することになる。

さて、ワークグループモデルでは、それぞれのクライアントマシンがどのワークグループに所属しているかを設定する必要がある。



画面1　Windows 98のワークグループの指定
コントロールパネルのネットワークを開き、識別情報タブの中で所属するワークグループ名を指定する。

たとえば、設計部、製造部、販売部のワークグループ名が、それぞれ、design、product、salesで、Windows 98のクライアントマシンをsalesグループに所属させることを考えてみる。この場合にはコントロールパネルの「ネットワーク」を開き、**画面1**のように識別情報のタブの中にある、ワークグループ欄にsalesを入力すればよい。

なお、Sambaサーバのsmb.confでワークグループ名をsalesに設定するには、

```
workgroup = sales
```

と指定することになる。

ワークグループではアクセスの管理が、それぞれのマシンの共有単位（フォルダやプリンタ単位）になり、特別な管理サーバを必要としないため簡単である。しかしながら、ワークグループで、共有レベルのセキュリティを使用した場合には、それぞれの共有単位（フォルダ単位）で多数のパスワードを必要とすることが問題になる。このため大規模で複雑なアクセス権限が必要になるようなネットワークでは、後述の「NTドメイン」を使用することになる。

NTドメインとPDC

NTドメインでは、アクセス管理を一手に引き受けるプライマリドメインコントローラ（PDC）と呼ばれる1台のマシンが存在する。

PDCには、SAM（Security Account Manager）と呼ばれるデータベースが存在し、この中には各ユーザーのパスワードなどがすべて保存されている。

SAMデータベースは、

```
c:¥winnt¥system32¥config¥
```

というディレクトリの中に保存され、レジストリエディタによりその構造を確認することができる。

ユーザーがログインする場合などのアクセス管理はすべて1つのPDCにある、SAMデータベースによって集中管理されることになる。

PDCとBDC

このように、1つのNTドメインにおけるすべての情報は1つのPDCに存在するため、PDCのディスクがクラッシュしたり、PDCが存在するネットワークセグメントに問題が起きたりすると大変困ることになる。

このため、PDCにあるSAMデータベースをバックアップするサーバを使用することができるようになっているが、このサーバのことを「バックアップドメインコントロール」、略してBDCと呼んでいる。

ここで、SAMデータベースは、PDCとBDCの間で同期がとられることになるが、データベースの変更が行われるのはPDC上だけで、BDC上にあるSAMデータベースは読み取り専用のコピーとなっている。

なお、大きな会社のネットワークでは、支店などが64Kbps程度の低速回線で結ばれることもあるが、このような場合に、PDCだけでアクセス管理を行おうとすると、SAMのサイズが大きすぎて問題が起こることもある。たとえば、NTサーバにおいて5000人のユーザーがいて、50のグローバルグループに平均100人が所属している場合、SAMのサイズは5Mバ

イト以上になってしまい、ログインに時間がかかるなどの問題が生じる。このような環境では、支店にBDCを置くとよい、ログオンなどの認証処理は支店のBDCが行うため、リソースへのアクセス時間が減り、ネットワーク全体のトラフィックも減少することになる。

プリンタサーバ機能

マイクロソフトWindowsネットワークでは、共有資源としてプリンタを扱うことができるが、プリンタに関しても先ほど説明した共有フォルダと同じように、アクセス制限をかけることが可能である。前述の共有レベルのセキュリティでは、それぞれのプリンタ単位にパスワードが設定でき、そのプリンタのアクセスパスワードを知っている人だけに、プリンタの使用を許可することができる。一方、NTドメインの場合は、PDCによってプリンタのアクセスが集中して管理されることになる。

マイクロソフトWindowsネットワークの印刷では、多くの場合、Windowsアプリケーションからの印刷要求をクライアントマシンで、各プリンタのプリンタドライバを使って各プリンタ独自の制御用バイナリデータを作成する。そしてそのデータを、プリンタサーバを提供するマシンへネットワーク経由で送ることになる。一方、プリントサーバはネットワークで送られてきたデータを、パラレルポートなどを通じ単純にプリンタへ転送することになる。

したがって、Sambaでプリンタ共有を実現する場合は、送られてきたデータを一般的なセントロニクス方式でパラレルポートにそのまま出力することになる。しかしながら、一部の昇華型プリンタなどでは、パラレルポートの端子を専用のプリンタドライバを用いて特殊な方法でリアルタイム制御し、ヘッドの加熱時間などを直接コントロールしているものもある。このようなプリンタではネットワークを通じた使用は不可能なので注意が必要である。

なお、最近では専用のプリントサーバが非常に安く買えるので、Sambaで共有するよりも、専用プリントサーバを用意したほうが管理上便利かもしれない。

漢字ファイル名の扱い

日本では、日本語のファイル名を使用することが当たり前のようになっているので、ネットワークでファイル共有を行う場合も漢字のファイル名のサポートは必須であろう。

筆者の知る限り、マイクロソフトネットワークでファイル共有を行う場合に、ネットワーク上でやりとりされるファイル名の漢字コードは、シフトJISの場合とUNICODEの場合がある。

ネットワークを流れるパケットを観測すると、Windows 95や、Sambaがサーバとなっている場合はファイル名を送信するのにシフトJISが使用され、Windows NTがサーバとなっている場合は、UNICODEでエンコードして通信が行われているようである。

なお、日本語版のWindows 95／98／Meの16ビットコード実行モジュールのシフトJIS解析ルーチンには、ネットワークドライブ上でシフトJISの漢字ディレクトリを正しく扱えないというバグが存在する。NTをサーバとする場合はUNICODEで通信が行われるため問題にならないが、Samba

Column

オープンなプロトコル

インターネットの普及に伴い、メールやWeb系のグループウェアがないと仕事にならなくなってきている。これらの機能を提供する、DNSサーバ、HTTP／FTPサーバ、SMTP／POPサーバなどはもともとオープンなUNIX上で開発されているため、これらを安定動作させるためには、LinuxなどのUNIX系OSを使用することが望ましい。そのため多くの組織ではUNIXに精通した技術者により、これらのサービスが提供されていることが多いようである。

一方、Windowsのファイル共有はオープンでないマイクロソフト独自の技術に基づいて実装されているため、オープンな技術に慣れているUNIX技術者には、その技術が「理解できない」ことも多い。特に大規模にSambaの運用をする場合は、かなり気合いをいれてWindowsネットワークを解析し、Sambaのソースコードにパッチを当てたりしないといけない場面もある。

このため、面倒なファイル共有は、NT技術者にまかせてしまうことも多いようである。しかし結局破綻してUNIX技術者が泣く泣く後始末をするという話も聞く。

このような事態を避けるためには、最初からWindowsのファイル共有を使わないというのもひとつの方法である。ファイルを共有する手段は、Sambaだけとは限らず、オープンなプロトコルを使うFTPやHTTPもある。最近のブラウザでは操作も簡単なので用途に応じて適切な手段を考えるとよいだろう。

Sambaのようなクローズドなファイル共有手段は、Windowsがバージョンアップすると役に立たなくなることもあるが、FTPやHTTPは不滅である。

▼リスト1　hexファイル名を日本語表示するスクリプト

```
#!/bin/sh

/bin/ls $* | perl -e '

while(<>){
        s/:([0-9a-f][0-9a-f])/pack('c',hex($1))/ge;
        print;
}' | nkf -eX
```

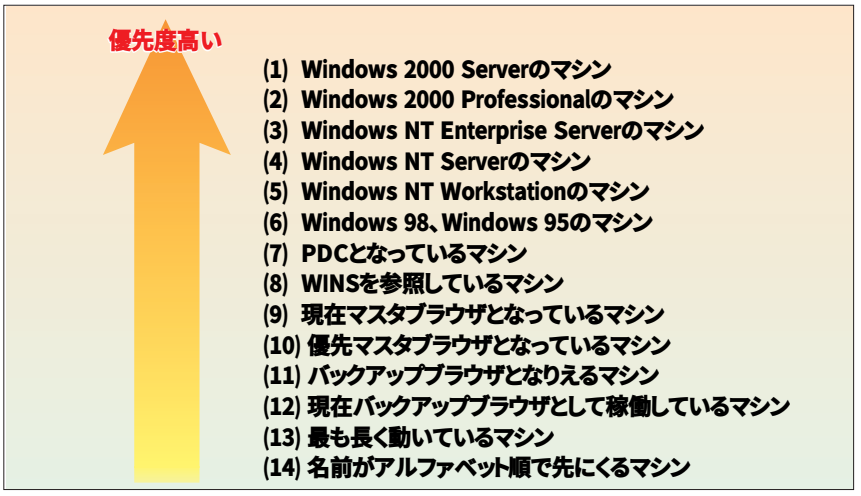


図2　マスタブラウザになる優先順位

を使用する場合にはシフトJISでの通信となり、漢字ディレクトリのアクセスができなくなる。このためSamba側にパッチをあてて、Windowsのバグを回避する必要がある。

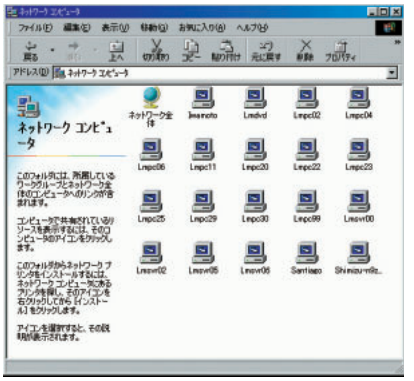
さて、上記のように、ネットワークでの通信では、シフトJIS、またはUNICODEが使用される。前述のようにSambaサーバを使用する場合にはシフトJISで通信が行われるが、送信されてきたシフトJISの漢字ファイル名を、Linux上のファイルとして保存するには、次のいずれかの形式を使用するように設定することができる。

(1) シフトJIS漢字
(2) EUC漢字
(3) jis7
(4) jis8
(5) junet
(6) hex
(7) cap

シフトJIS漢字を使用すると、2バイト目が特殊文字となる漢字を扱った場合に、Linuxからファイルをアクセスすることが難しくなることがある。EUC漢字の場合も、丸数字や半角カタカナが使用されると、Sambaを動作させるOSによってはlsコマンドなどで正しく扱えないことがある。

なお、マッキントッシュが混在するネットワークでNetatalkを使用する場合にはcapを使用する必要があるが、Macとの連携を考えない一般的な場合は、hexを指定するのがよいだろう。

hexを指定した場合、ネットワークで送られてきたシフトJIS漢字ファイル名を「:」と16進数2桁という形式に変換して、Linuxのファイル名とする。この形式であれば、lsやtarなどのコマンドが誤動作することがなく扱うことができる。また、この形式は非常に単純なので、たとえば**リスト1**に示すようなスクリプトを使用することにより、Linux上でも漢字ディレクトリの表示を行うことができる。



画面2　Windowsのネットワークコンピュータの表示
自分が所属するワークグループ／ドメインに参加しているコンピュータが表示される。

ただし、hexを使用する場合でも若干の注意が必要な場合がある。それはシングルクォート「'」などシェルが特殊文字として解釈する文字がファイル名として使われる場合がある。

Sambaサーバを運用していると、findコマンドなどで、不要ファイルの削除のために30日以上経過した特定のファイル名を得て、先頭に「rm」などの文字列を付加してシェルスクリプトとして実行することがあるが、この場合には「'」をバックスラッシュなどでエスケープしなければならない。または、findコマンドのオプションで-execを使用するなど、シェルを使わないようにするとよい。

シングルクォートは西暦の上位2桁の省略記号として頻繁に使用されるので注意が必要である。

ブラウジングの仕組み

デスクトップのネットワークコンピュータアイコンをクリックすると、画

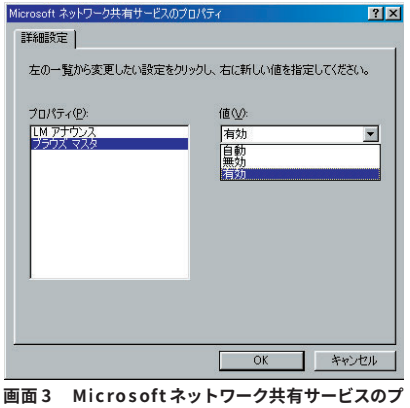
面2のように自分が所属するグループのすべてのコンピュータが表示される。このようなコンピュータのリスト表示をブラウジングリストと呼ぶが、このブラウジングリストは、ブラウザサーバと呼ばれるマシンが集中して管理している。

ブラウザサーバには、次のような4つの種類がある。

- **ドメインマスタブラウザ**
- **マスタブラウザ**
- **バックアップブラウザ**
- **優先マスタブラウザ**

1つのネットワークには最低1つのマスタブラウザを置く必要がある。たとえば、2台のWindows 95マシンだけからなるネットワークでは、最初に立ち上げたマシンがマスタブラウザになり、次に立ち上げたマシンがバックアップブラウザになる。

一方、NTサーバなどを含む大規模なネットワークでは、どのマシンがマスタブラウザになるかを決定するために、特定のアルゴリズムによる「選挙」が行われる。この選挙では、**図2**に示す優先順序でマスタブラウザが選出される。このような複雑なアルゴリズムを使う理由は、ネットワークコンピュータアイコンを開くときに、すべてのコンピュータに対し、ブロードキャストパケットを送ることが非効率であるからである。



画面3　Microsoftネットワーク共有サービスのプロパティ
ブラウズマスタを無効に設定することで、そのマシンがブラウズマスタになることを禁止できる。

しかしながら、上記のようにしてマスタブラウザが選出されるため、電源を入れた直後のサーバが見えなかったりすることがある。特にWindows 98などが、マスタブラウザになっていて、ハングアップなどで異常終了すると、回復するまで時間がかかる場合もある。

Sambaサーバを運用する場合、サーバがアイコンとして見えないという問題が発生することがあるが、この問題を解決するためには上記のブラウジングの仕組みをよく理解しなければならない。特にローカルマスタブラウザが存在するネットワークセグメントの変更が、ネットワーク全体に反映されるまでは、1時間以上かかることもあるので、ブラウジングを安定運用するためには、マスタブラウザの選定に関する設定が重要になる。

なお、Windows 98などのクライアントマシンでは、コントロールパネルのネットワークの設定にある、「マイクロソフトネットワーク共有のプロパティ」のウインドウ（**画面3**）で、ブラウズの選出を制御できる。この値を無効にしておけば、不安定なクライアントマシンがブラウズマスタとなってしまうことを避けることができる。

一方、Sambaサーバをマスタブラウザとする場合、マスタブラウザでの選挙に勝つためには、次のような設定をsmb.confで行う。

```
local master = yes
preferred master = yes
domain master = yes
os level = 65
```

Sambaにおいて、ブラウジング機能を実現しているのは、nmbdというプロセスであるが、local masterをyesにすると、nmbdはマスタブラウザの選挙に立候補することになる。逆にlocalmasterをnoにすると、nmbdは決してマスタブラウザとして選出されなくなる。

さらに、preferred masterをyes

Column

ブラウジングより便利なアクセス方法

本文で解説したように、ブラウジングのアルゴリズムはかなり難解である。特にタイムアウトしたときの挙動の詳細を把握することは困難で、トラブルがあったときに対処できないことも多い。

このためSambaサーバを運用する場合は、ネットワークコンピュータでサーバを表示してアクセスするのではなく、直接名前で指定する方法をユーザに告知しておくというやり方がある。

自動的に、前回の接続を回復するようにしてもよいし、ドメインログオンを使って、サーバの名前でドライブ割り当てをしてもよいだろう。

または、Windowsのスタートボタンを押して「ファイル名を指定して実行」を選び、直接、

\\サーバ名

を入力してもらうようにする方法もある。サーバ名を2文字くらいにしておけばキーボードが苦手な人でもさほど問題はない。

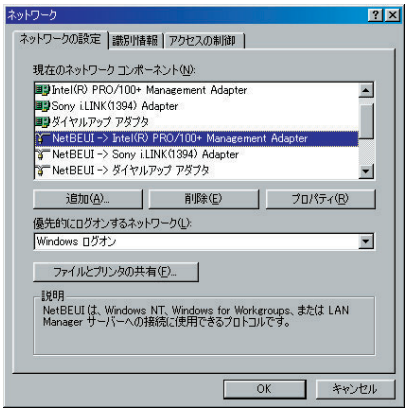
この方法では、前回入力したサーバ名が記録されるので、ネットワークコンピュータのたくさんのマシンの中から目的のサーバを探すよりも簡単にアクセスできるはずだ。

に設定すると、nmbdは前述の優先マスタブラウザになることができ、図2にある選出アルゴリズムの10番目に該当する場合、他のマシンに勝ってマスタブラウザとして選出される。

また、os levelは、OSの種類を決定するもので、Windows 95／98はレベル1、Windows NT4.0 Serverはレベル33というようにOSによってそのレベルが定められている。このため、smb.confで「os level = 65」とすることによりNTサーバに勝つことができ、必ずSambaサーバがマスタブラウザとして選出されることになる。

そして、「domain master = yes」とすると、nmbdは、複数のネットワークセグメントにあるNTドメインの中央マスタブラウザとして動作する。通常、マスタブラウザの選出は、ブロードキャストによって行われるため、ルータを介して違うセグメントにあるマシンのブラウジングには特別な設定が必要になり、WINSなどとの連携が必要になることがある。

なお、ブラウジングに関する情報については、マイクロソフトのサポートページで、「ブラウザ」と「オペレーション」をキーワードとして検索すると見つかる文書番号J030937のドキュメントが参考になる。このドキュメントでは、上で説明したオペレーティングシステムタイプや、優先マスタブラウザの情報が、32ビットのビット演算で表されていることが示されており、選出のアルゴリズムが解説されている。



画面4　コントロールパネルのネットワーク設定
NetBEUIプロトコルがインストールされていれば、TCP/IPを使わずにWindowsマシン同士で通信可能。

名前解決

マイクロソフトWindowsネットワークで、サーバにアクセスできない場合は、名前解決ができていないことが多い。

TCP/IPネットワークを使っている場合の名前解決とは、与えられたマシンの名前をIPアドレスへ変換することである。名前解決は単純な作業であるにもかかわらず、Windowsでの名前解決方法は、NetBIOS（Network Basic Input Output System）とWinsockが複雑に絡み合っている。そのうえ、ブラウジングやWINSの存在が名前解決の理解をさらに難しいものとしている。

■Winsock/DNSとNetBIOS/WINS

WindowsマシンからUNIXマシンへアクセスするには、MS-DOSプロンプトから、

```
C>telnet machine01
```

というようなコマンドを実行する。この場合machine01という名前が、たとえば192.168.0.10というようなIPアドレスに変換（名前解決）されて通信が行われる。

一方、machine01という名称のマシンのshare01という共有名を、xドライブに割り当てるためには、MS-DOSプロンプトで、

```
C>net use x: ¥¥machine01¥share01
```

というコマンドを実行する。この場合もmachine01という名称が名前解決され、192.168.0.10というようなIPアドレスに変換される。

上記の、telnetコマンドの例とnet useコマンドの例は同じように名前解決を行っているわけであるが、その名前解決の方法はかなり違い、それぞれ次のような名前解決方法が使われる。

- **telnetコマンドはWinsockで解決**
- **net useコマンドはNetBIOSで解決**

telnetは、いわゆるインターネット型のコマンドで、TCP/IPプロトコルが有効になっていないと実行できないのに対し、net useは、いわゆるマイクロソフトWindowsネットワーク型のコマンドで、必ずしもTCP/IPが有効になっていなくても実行できる。つまり**画面4**のようにNetBEUIプロトコルが有効になっていれば、TCP/IPの代わりにNetBEUIプロトコルでの通信が行われるのである。

ところで、NetBEUIはルータを越えられないため、非常に小さい範囲のネットワークでしか使用することができない。同様にIPXやAppleTalkも大規模な利用には向かないため、現

	Winsock	NetBIOS
アプリケーション	telnetやpingなど	net useやnbtstatなど
静的な名前解決	c:\windows\hosts	c:\windows\lmhosts
動的な名前解決	DNS	WINS

表1　WinsockとNetBIOSでの名前解決

在では、TCP/IP以外のプロトコルの使用を禁じている組織が多い。また、NetBEUIが有効になっていると、コンピュータが見える／見えないといった問い合わせが頻発したり、トラブルがあったときに、tcpdumpなどでの解析が難しくなる。NetBEUIは禁止して、TCP/IPのみでの運用が望ましいだろう。

さて、本題のWinsockとNetBIOSの名前解決方法であるが、これには**表1**に示すような方法が使われる。なぜ同じような名前解決方法が2つもあるのだろうと不思議に思う読者もいるかもしれないが、これはNetBIOSという前世紀の遺物を、いまでも捨てきれないでいるからである。

しかしながら、まだまだWindows 95が存在する環境は多く、そのような場合に、Sambaを安定運用するためには、NetBIOSでの名前解決についての詳細な理解が欠かせない。

■Winsock/DNS

インターネットの通信において基本となるTCP/IPプロトコルは、もともとBSD-UNIX上で開発されたものである。BSDにおいて、TCP/IPでのアクセスは、socketライブラリと呼ばれるライブラリによって実現されていた。このBSDソケットライブラリをそっくり真似してWindowsに移植したのがWinsockである。

telnet.exeや、ping.exeなどのいわゆるインターネット型のコマンドでの名前解決は、このWinsockライブラリを使って行われることになるが、このライブラリでのアルゴリズムをフローチャートにしたのが**図3**である。

このアルゴリズムでは、最初に、hostsファイルを検索し、次にDNSサーバを検索する。そして、DNSに存在しなかった場合は、WINSサーバへ問い合わせ、それでも解決できない場合はブロードキャストを行ったり、lmhostsファイルを参照することになる。

■NetBIOS/WINS

マイクロソフト社は、NetBIOSと呼ばれるAPIの上に独自のネットワークアプリケーションを構築してきた。

当初NetBIOSは、NetBEUIと呼ばれるプロトコルの上に実現されたが、このNetBEUIプロトコルは、ルータを越えられないなど、狭い世界での使用しか考えられていない場当たり的なプロトコルであった。

このため現在では、NetBIOSパケットを、TCP/IPにカプセル化して通信する方法が考え出された。これは、NetBIOS over TCP/IPと呼ばれ、NetBTや、NBTと略される（NetBIOSの状態を調べる、nbtstat.exeコマンドの最初の3文字がこの略である）。

このNetBIOSの名前解決手段を知っておかないと、Sambaを運用するうえでのトラブルが解決できないので、これを理解することは重要である。

さて、NetBIOSでの名前解決アルゴリズムをフローチャートにしたものが、**図4**である。

ここでは、IPアドレス形式、またはインターネットのドメイン形式（FQDN）などの特殊形式である場合は、それに基づいた解決が行われる。

しかし、一般的な名前の場合は、WinsockやDNSで解決する前に、WINSサーバや、ブロードキャスト、そしてlmhostsでの解決が優先されるのが特徴である。

なお、NetBIOSでの名前解決はIPアドレスだけでなく、サーバの種類を示す情報も必要なことがある。この情報は2桁の16進数で表され、たとえばドメインログオン可能なサーバの場合は0x1Cとなる。

■名前解決トラブルの原因調査

Windowsでの名前解決は大変複雑だが、大まかな概念は単純である。つまりtelnetなどのいわゆるインターネットタイプのアプリケーションは、Winsock/DNSを第一優先として検索し、見つからなかった場合はNetBIOS/WINSで検索するのである。

これと反対に、ファイル共有を行う、net useコマンドなどいわゆるNetBIOSタイプのアプリケーションの場合は第一優先がNetBIOS/WINSであり、その次にWinsock/DNSを用いることになる。

結局優先する順番が異なるだけであるので、どちらかでIPアドレスが解決できれば通信が可能になるのである。

したがって、ルータを越えたSambaサーバに接続できない場合に、まず最初に確認すべきことは、

```
ping サーバ名
```

としてアクセスできるかどうかを調べることである。ここで応答がない場合は、何らかの方法で名前解決をしなければならない。

Windows 98以降では、IPアドレスを直接記述できるので、192.168.2.3というIPアドレスのサーバには、

```
net use x: ¥¥192.168.2.3¥共有名
```

として、ドライブ割り当てができる。

または、

```
net use x: ¥¥xx.yy.zz.jp¥共有名
```

というように、サーバ名を完全なインターネットドメイン名（FQDN）で指定する方法もある。

しかしながら、Windows 95では上記のようなIPアドレス形式や、FQDN形式は受け付けない。このため、接続すべきサーバがクライアントマシンとは別のインターネットサブドメインにある場合には、コントロールパネルにあるネットワーク設定の「ドメインサフィックスの検索順」に、サーバのインターネットサブドメインを追加するとよい（**画面5**）。

または、クライアントの存在するインターネットサブドメインのDNSで、サーバマシンをCNAMEで定義するという方法もある。

なお、DNSでの解決ができない場合はWINSサーバを立てるか、hostsやlmhostsファイルに、サーバのIPアドレスを記述することになる。

上記のいずれの手段を用いて、とにかく、pingで、応答がくるようにすれば、ルータ越えでも問題なく接続が可能になるはずである。

また、Windowsに標準で添付される、nbtstat.exeもトラブルの解決には役に立つ。マスタブラウザの確認や、ドメインログオンが可能になっているかなどの情報（0x1Cなどの16進数で表示される）もこのツールを使用することによって調べることができる。

そして最も強力なツールは、UNIX

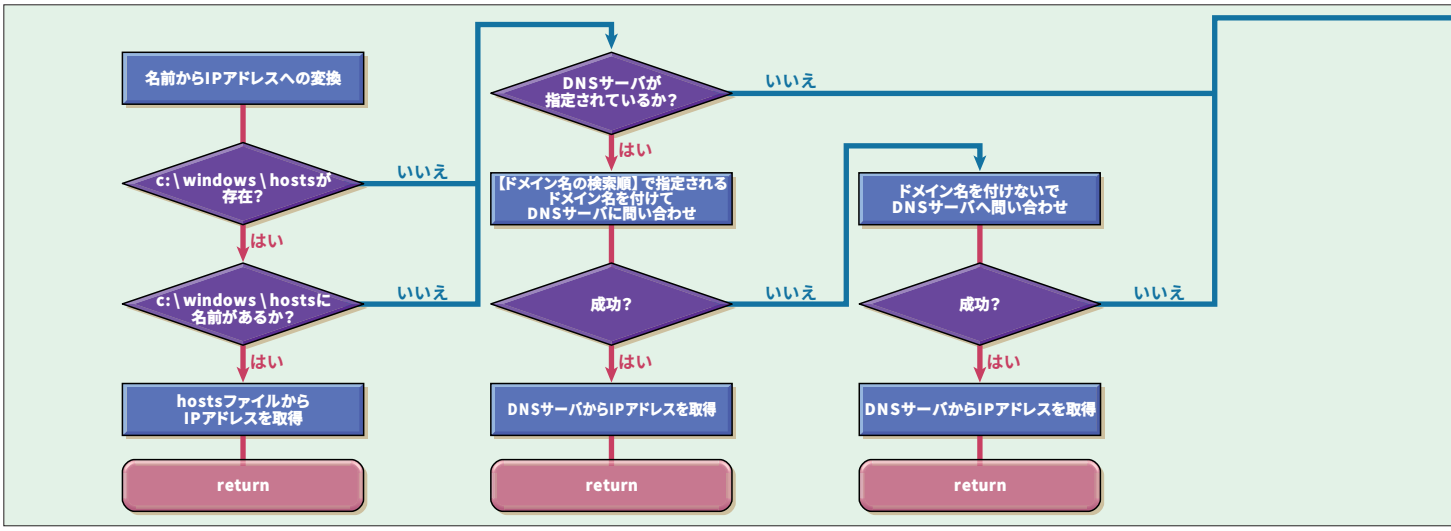


図3　Winsockにおける名前解決のアルゴリズム

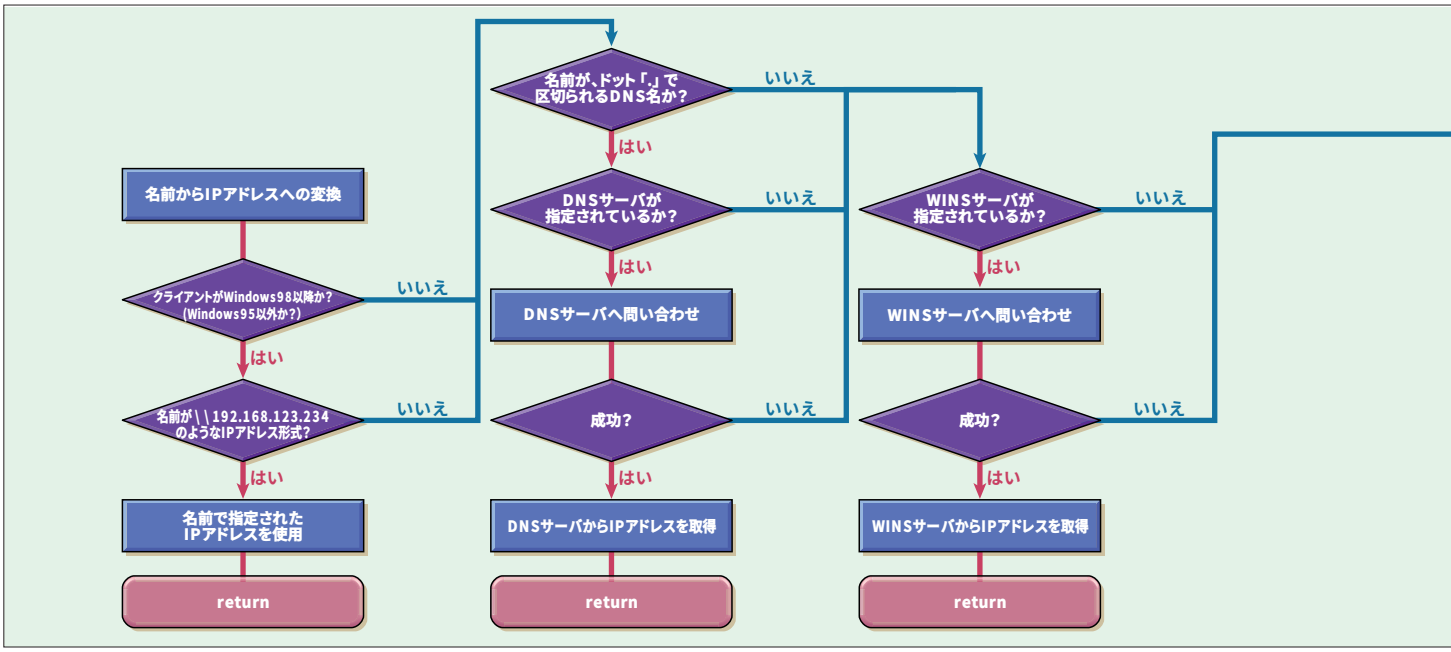


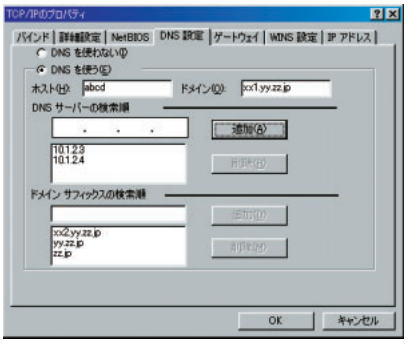
図4　NBT(NetBIOS over TCP/IP)での名前解決

上のtcpdumpである。それなりの知識は必要になるが、tcpdumpでパケットを解析することで、トラブルの原因がわかるはずである。

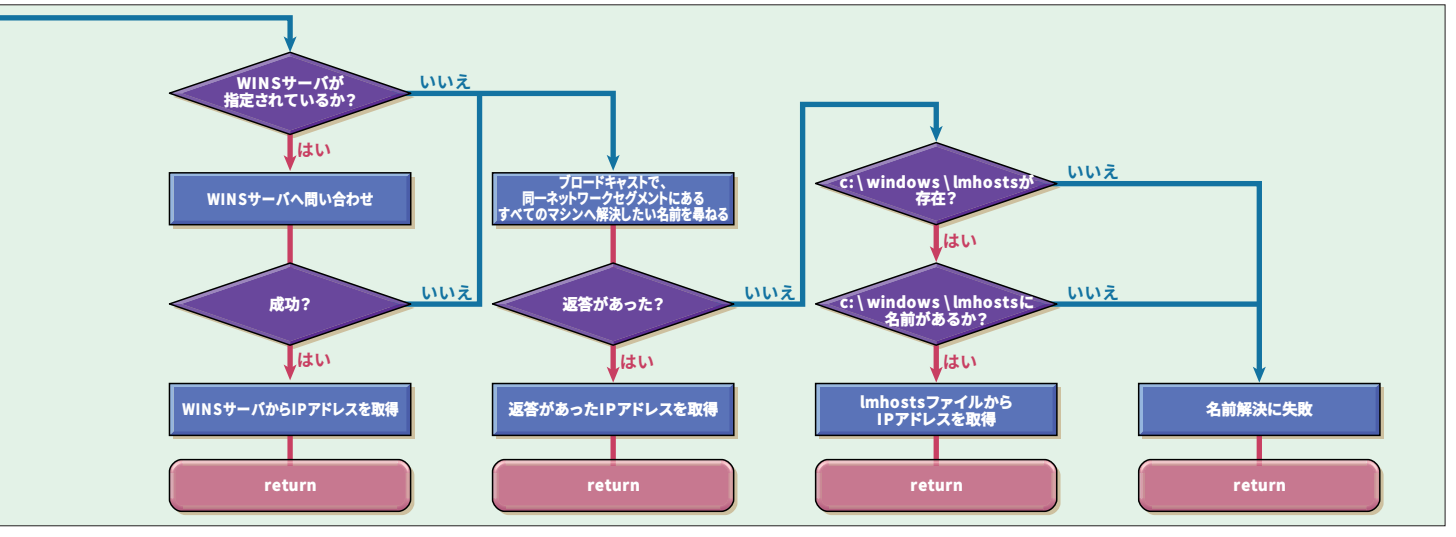
まとめ

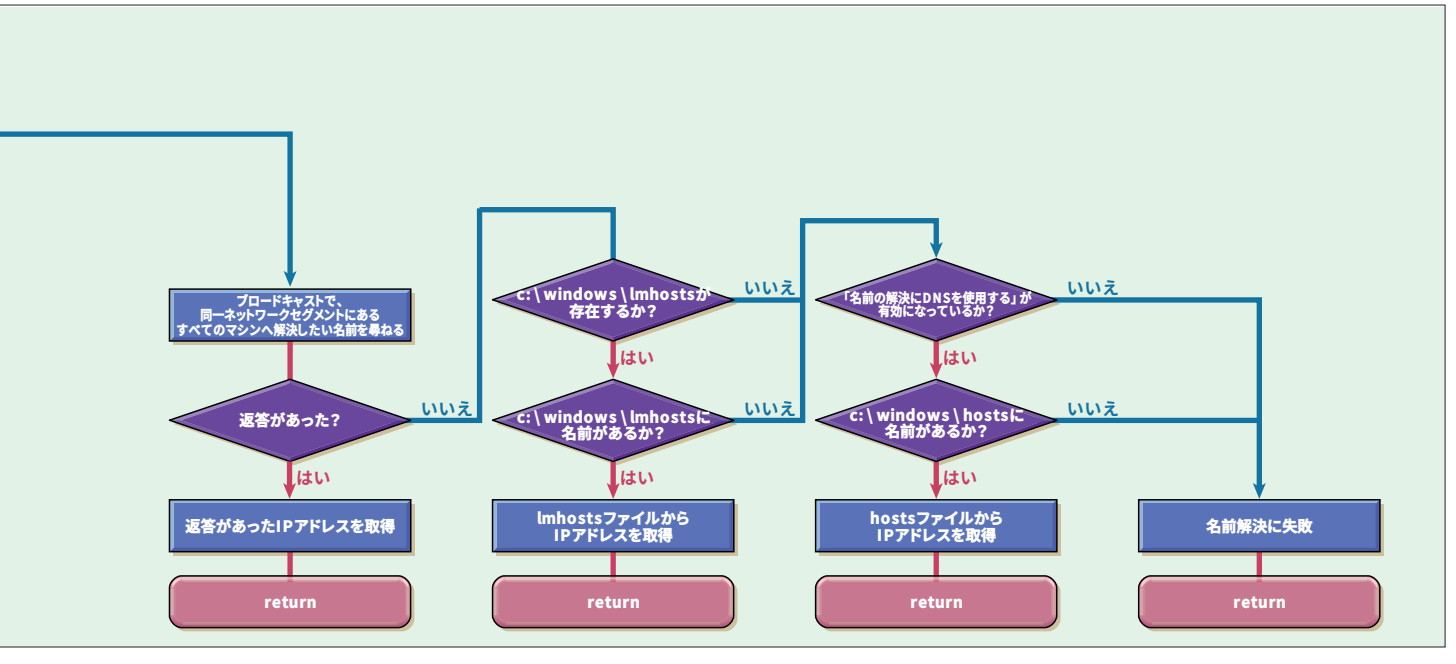
筆者はいままでSambaを運用してきてさまざまなトラブルに遭遇した。

NFS、FTP、HTTPといったUNIXベースのファイル共有手段と比べ、そのトラブルに費やす時間は比較にならないくらい多く、解決が非常に難しいものもあった。Windowsネットワークは、WINSのような過去の遺物を引きずっているため、意味もなく複雑な部分も多く、その完全な理解はかなり難しい。しかしながら、今回解説したようなWindowsネットワークにおける基本の動作が理解できれば、トラブルの解決にはかなり役に立つのではないかと思う。



画面5　ネットワークコンポーネントのTCP/IPのプロパティ
Windows 95で別のサブドメインのマシンに接続するには、ドメインサフィックスにそのドメイン名を記述する。





Sambaで何ができるか？

文：ミラクル・リナックス株式会社　小田切 耕司
Text : Koji Odagiri

Sambaは、UNIXやLinuxマシンをWindows NT互換のファイルサーバ／プリントサーバにするオープンソースソフトウェアだ。GPL（GNU General Public License）の元に無償で利用することができる。製品分類としては、その使用するプロトコルからSMBサーバ製品と言われていたが、最近はSMBの上位プロトコルを指すCIFS（Common Internet File System）サーバとして広く認識されている（参考：ftp://ftp.microsoft.com/developr/drg/CIFS/）。

SambaやマイクロソフトのWindows製品以外のCIFSサーバとしては、サン・マイクロシステムズのSun PC NetLinkやSyntaxのTAS（TotalNET Advanced Server）、IBMのFastConnect、SCO VisionFSなどがある。

SambaはオーストラリアのAndrew Tridgell氏らによって1992年に開発された。現在Sambaは、VA Linux Systemsの Andrew Tridgell氏やJeremy Allison氏らボランティアではなく、専任の担当者によって、開発／サポートが行われている（http://samba.org/samba/samba.html）。もちろん、日本人を含めた世界中のボランティアの方も多数参加しており、Sambaの日本語化は日本Sambaユーザ会（http://www.samba.gr.jp/）を中心に行われている。

ここでは、Sambaが備える機能を以下の3つに分類し、簡単に紹介してみよう。

(a) Windows NT／2000と同等のSambaの機能
(b) Windows NT／2000にはないSambaの機能
(c) Sambaの制限事項およびWindows NT／2000との相違点

ここでの紹介はSamba 2.2.1aをもとに解説する。Samba 2.0との違いは、次章で解説する。

Windows NT／2000と同等のSambaの機能

SambaはWindows用ファイル／プリントサーバとして必須といえる基本機能だけでなく、他のCIFS製品にはないPDCやWINSサーバの機能も有している。

■ファイルサーバ機能

Linux上にあるディレクトリを「共有」としてWindowsクライアントおよびSambaクライアントに公開できる（**画面1**）。

Sambaはその名前の由来通りSMB（Server Message Block）プロトコルを使用する。しかし、正確にはSambaの使用する通信プロトコルはNBT（NetBIOS over TCP/IP）であり、下位プロトコルはTCP/IPのみになる。Windows NT／2000のようにNetBEUIやIPX/SPXを使ってWindowsマシンと通信することはしない（**図1**）。

■プリントサーバ機能

Linuxに接続されているプリンタ「共有プリンタ」としてWindowsクライアントおよびSambaクライアントに公開できる。

プリンタドライバはLinux用のもの

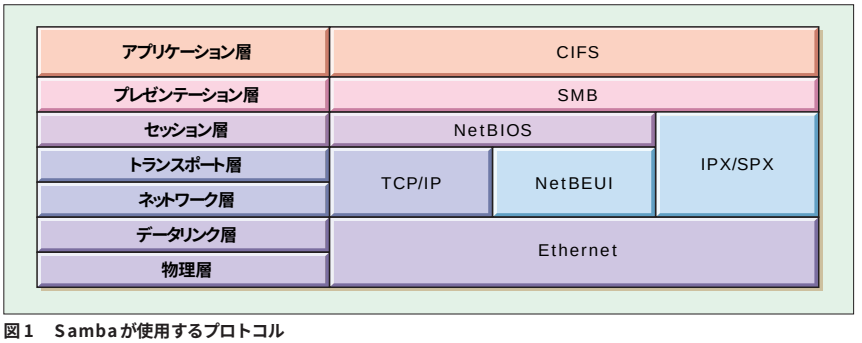


図1　Sambaが使用するプロトコル

画面1　Sambaのファイル／プリンタ共有機能
Windows NT／2000サーバと同様に共有ごとに使用できるユーザーを制限できる。

ではなく、Windows用のものをクライアントにインストールして使用する。

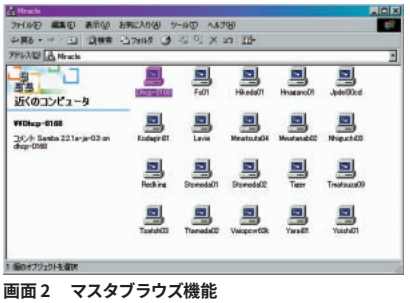
最近のプリンタはネットワークプリンタとしてSambaやWindowsサーバを介さずにWindowsクライアント間で共有することが可能だが、Sambaを使ってプリンタを共有するメリットはWindows 95／98／Me／NT／2000クライアントに対し、プリンタドライバの自動インストールができることだ。

■SMBクライアント機能

LinuxマシンからWindowsの共有ファイルにアクセスしたり、共有プリンタに印刷したりすることができる。

通常これはsmbclientコマンドを使用するが、mount（内部はsmbmount）コマンドを使用して、Windows共有をLinuxのファイルシステムとしてマウントすることも可能である。

LinuxからWindowsのプリンタドライバを使って印刷することはできない。LinuxからWindowsのプリンタに印刷する場合、プリンタドライバ（フィルタ）はLinux側に必要だ。Windowsに接続されているプリンタがポストスクリプトをサポートしているなら、Linux側で作成したPSデータをそのまま印刷できる。Windowsに接続されているプリンタがESCPage（エプソン）やLIPS（キヤノン）などのポストスクリプト以外の場合は、印刷可能なデータはLinux側で作成する必要がある。



画面2　マスタブラウズ機能
Sambaは、ネットワークコンピューター覧を提供するドメイン・マスタブラウザになることができる。

Linux側に適切なプリンタドライバ（フィルタ）がなく、Windowsのプリンタドライバしかない場合、Linux側でPSデータをPDFファイルに変換し、Windows側のAcrobatで印刷するという手もあるだろう。

■PDC（プライマリドメインコントローラ）機能

Windowsクライアントに対し、ドメインログオンを提供し、ログオン時にスクリプトを実行させることができる。

■認証サーバの統合

1つの認証サーバを複数のSambaサーバから利用できる。

■ドメインメンバ機能

WindowsドメインにSambaサーバをメンバサーバとして追加することができる。

ユーザー認証がすべてWindows NT／2000のドメインコントローラで行えるのでパスワードの二重管理が必要なくなる。

Windowsドメインにユーザーを追加した場合、自動的にSambaユーザーを追加する機能もある。

■WINSサーバ機能

WINS（Windows Internet Name Service）サーバになることができる。

■マスタブラウザ機能

ブラウズリストを保持し、Windowsマシンにネットワークコンピューター覧を提供する。LMB（ローカルマスタブラウザ）はWindowsマシンもSambaもなることが可能だが、DMB（ドメインマスタブラウザ）はWindows 95／98／Me、Windows NT／2000 Workstationはなることができず、Windows NT／2000サーバとSambaだけが可能だ（**画面2**）。

■サーバの別名登録

通常はLinuxのホスト名がネットワークコンピューター覧に表示される名前になるが、1つのマシンに複数の名前をつけてネットワークコンピューター覧に表示させることができる。

Windows NT／2000も同様のことができるが、特殊なレジストリを変更しないといけない。

Windows NT／2000にはないSambaの機能

SambaはLinux（UNIX）とWindowsの違いを吸収する仕組みや、NetwareにあってWindows NTにはないユーザーホーム機能やQUOTA機能を持っている。加えてファイルサーバ機能を拡張したり運用を効率化するためのさまざまな工夫が加えられている。

■隠し共有の扱い

Windows NT／2000では共有名に$を付けると隠し共有になるが、Sambaでは任意の共有名を隠し共有にできる。

■ユーザーホーム機能

共有の一覧表示時に、クライアントのユーザー名を共有名としてマッピングして表示する機能。

たとえば、yamadaというアカウントでSambaマシンをアクセス（ネットワークコンピュータをクリック）した場合、Linux上の/home/yamada

が共有名yamadaとして表示され、また、suzukiというアカウントでSambaマシンをアクセス（ネットワークコンピュータをクリック）した場合はLinux上の/home/suzukiが共有名suzukiとして表示される（**図2**）。

これによってユーザーごとの専用のディレクトリを提供し、他人のディレクトリをアクセスするのが防げる。

■リモートアナウンス機能

Windowsではセグメントをまたがったワークグループを構成できない（ネットワークコンピュータ一覧に表示できない）。

また、別セグメントにある別ドメインをネットワークコンピュータ一覧に表示するには以下が必要となる。

・ドメイン間で信頼関係を結ぶ
・別ドメインに所属するブラウズマスタになれるマシンを、同一セグメントに用意する

Sambaは上記を行うことなく、リモートアナウンス機能を使って別セグメントにあるワークグループやドメインをネットワークコンピュータ一覧に表示することができる。

■ユーザー名マッピング

Windows 95／98／Meは、Windows NT／2000サーバにアクセスするときのユーザー名を変更することができない。

Sambaはサーバ側でクライアントのユーザー名をマッピングして変更することができる（**図3**）。

■コマンド実行

クライアントが共有に接続した時、または共有との接続を切断した時に、Sambaサーバ側でコマンドを実行することができる。これを使ってCD-ROMやMO（光磁気ディスク）の共有などを行うことができる。

■クライアントホストの制限

共有ごとに使用できるクライアント（ホスト名やIPアドレス）を制限できる。

■クライアントごと、ユーザーごとに設定ファイルを用意可能

クライアントごとやユーザーごとにアクセスできる共有名やそれぞれの設定を変更できる（**図4**）。

■QUOTA（ユーザー／グループ単位の容量制限）機能

LinuxのQUOTA機能を利用することでWindows NTでは標準サポートされていないQUOTA（ユーザー／グループ単位の容量制限）機能がSambaで利用することが可能だ（ただし、Windows 2000サーバでは利用可能）。

Sambaの制限事項およびWindows NT／2000との相違点

Sambaは、Windowsのふるまいに合わせて改良されてきたが、すべてを解決しているわけではない。外国人

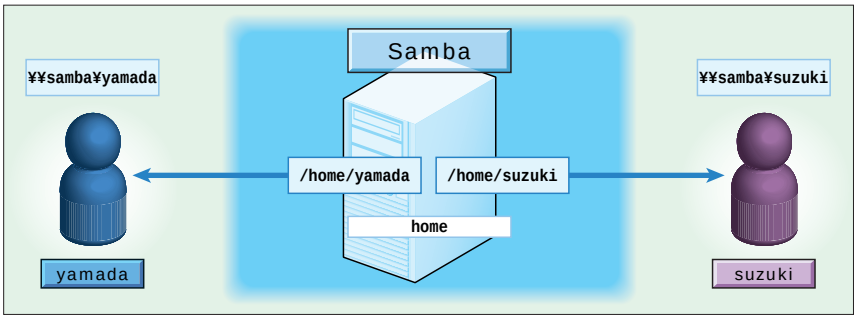


図2　ユーザーホーム機能

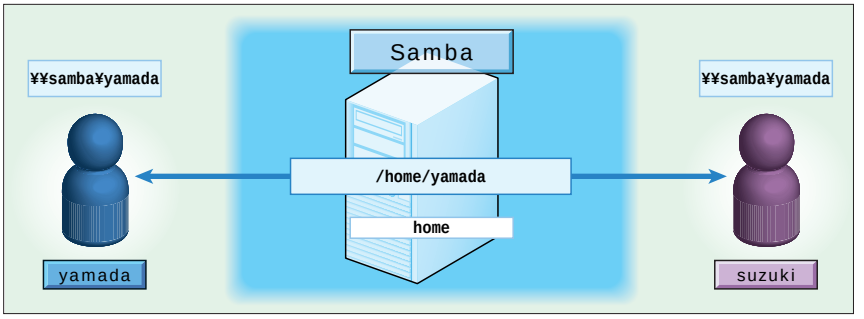


図3　ユーザー名マッピング

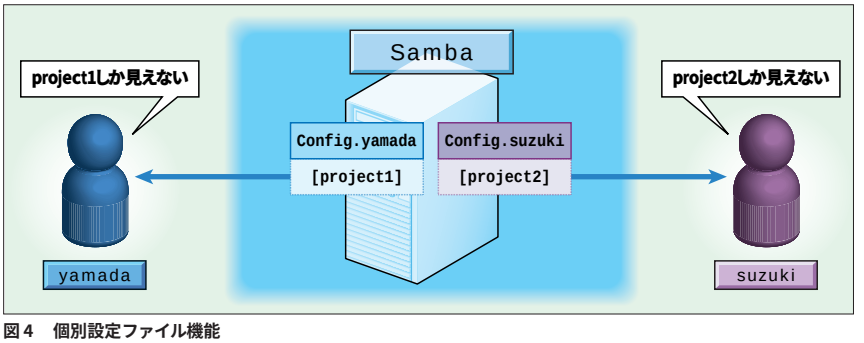


図4　個別設定ファイル機能

には理解しにくい日本語の問題や、マイクロソフトが仕様を公開しないドメインコントローラ機能などの問題が残っている。しかし、これらは日本Sambaユーザ会やSamba Teamの努力で少しずつ解決してきており、今後制限事項は減少していくはずだ。

■NetBEUI、IPX/SPXは使用しない

先にも述べたが、Sambaが使用するプロトコルはNBTのみのため、TCP/IPが組み込まれていないWindowsクライアントと接続することはできない。

逆にクライアントにNetBEUI、IPX/SPXが組み込まれていると問題が発生するので、これらは削除したほうがいい。

■日本語の問題

Sambaの設定ファイルsmb.confに、coding systemパラメータで指定したコードを使って以下の日本語パラメータを記述できる。

- **共有名**
- **プリンタ名**
- **共有やプリンタの説明（コメント）**
- **NetBIOS（コンピュータ）名**

もちろん、日本語ファイル名（日本語ディレクトリ名）にも対応している（ファイルの中身の漢字コード変換機能や改行コード変換機能は持っていない）。

coding systemパラメータには、主に以下のものが使用できる。

- **sjis（シフトJIS）**

Windowsが使用するものとほぼ同じコードのため、Windowsとの互換性が高い。しかし、Windowsと完全に同一ではないし、一部のUNIXではsjisは使用できない。Linuxにおいてもすべてのコンポーネントがsjisを取り扱えると検証されたわけではない。

- **euc（拡張UNIXコード）**

Linux（UNIX）で一番安全なコード体系である。しかし、オリジナルのSambaでは、①や㈱などのWindowsでしか使用できない機種依存文字や外字を扱うことができない。Samba日本語版では、機種依存文字や外字をEUCの外字領域にマッピングすることでほぼ対応している（もちろん、これらはLinux上で表示することはできない）。

- **cap**

CAPやNetatalkなどを使用してマッキントッシュとファイル共有する場合に使用する。Vine Linuxなどのディストリビューションのように、NetatalkでもEUCを使用するように改変されているものもあるので使用するコードには注意してほしい。

- **hex**

ファイル名を16進変換して格納する。eucやsjisが使用できない場合でも安全に使用できる。ただし、Linux上では16進表記になるため、ファイル名がわかりにくいが、Samba日本語版の開発にも尽力している白井氏によるFDcloneのようにCAPやHEXのファイル名に対応したツールがあるので活用すると良いだろう。

どのcoding systemを使用するかはユーザーの運用形態に合わせて選択すべきだが、OSがサポートしていない漢字コードを使用すると、バックアップ機能などで障害が起きる可能性がある。

LinuxではSamba日本語版でEUCを使用するのが一番安全だ。

■ドメインコントローラ機能

SambaはWindowsドメインのドメインコントローラになることができる。しかし、なれるのはPDCだけで、BDC（バックアップドメインコントローラ）にはなれず、またSambaのPDCに対し、WindowsサーバをBDCとすることもできない。

■WINSサーバ機能

プライマリのWINSサーバにだけなることができる。

別な（Windows NTやSambaの）WINSサーバと同期（プッシュとプル）を取ることはできない。

このため、同一ドメインおよび同一セグメントに複数の（Sambaによる）WINSサーバを置くことはできない。

■暗号化パスワードの問題

SambaはWindows NT／2000と同様に平文パスワード（plain text password）と暗号化パスワード（encrypt password）の両方で認証することができるが、暗号化パスワード（MD4ハッシュ、LANMANハッシュ）は、Linux（UNIX）のpasswdファイル（MD5ハッシュ）と互換性がないため、別々に管理しなければならない。

しかし、この不便を補完するためにLinuxとのパスワード同期機能や平文パスワードから暗号化パスワードへの移行機能が提供されている。

■大文字と小文字の問題

Windowsも表示に関しては、ファイル名の大文字小文字の区別があるが、アクセス時には区別がない。そのため、同一ディレクトリにABCというファイルとabcというファイルを作成することはできない。

ところが、Linux（UNIX）には正確な大文字小文字の区別があるため、同一ディレクトリにABCというファイルとabcというファイルを作成することが可能だ。Sambaサーバにこうしたファイルが存在すると、アクセス時に予期できない結果になることがある。

■16ビットプログラムのための8.3形式の短いファイル名の問題

Windowsは16ビットプログラムのために8.3形式の短いファイル名と255バイトまでの長いファイル名をファイルシステム（FAT、NTFS）に保持している（ファイル作成時に決めて、2つを保存）。

しかし、Linux（UNIX）のファイルシステムは長いファイル名しか保持しないため、Sambaは16ビットプログラムのために8.3形式の短いファイル名をアクセス時に自動生成してクライアントに返す（これをName Manglingという）。

このため、似たような長いファイル名が複数あるとSambaはこれを正しく区別できないことがある（クライアントから短い名前でアクセスした時、間違ったファイル名を返すことがある）。

■ファイル保護機能（セキュリティ）

各共有に対するファイル保護機能（セキュリティ）は、SambaとWindows NTでほぼ同等で、ホスト（クライアント）名に対する制限がかけられる分Sambaのほうが有用なのだが、ファイル単位の保護機能ではACL（アクセスコントロールリスト）をサポートするWindows NT／2000のほうが細かく設定できる（ただし、NTFSを使用した場合のみ）。

Samba上のファイルもWindowsのエクスプローラで各ファイルのプロパティを使ってセキュリティタブが利用できるが、「所有者（owner）」「グループ（group）」「それ以外（other）」の3種類でしか制限できない（**画面3**）。

■ファイル属性の取り扱い

Windowsではファイルに「読み取り専用」「隠しファイル」「アーカイブ」「システム」といった属性があるが、Linux（UNIX）にはこうした属性はない。

また、Linuxでは、.（ドット）で始まるファイルが隠しファイルとなるのが一般的だが、SambaにはLinuxの特定ファイルをWindowsのファイル属性にマッピングする機能がある。

これらを正しく設定すれば、ファイルのプロパティで属性を表示したり、設定したりできる。

■隠しファイルの取り扱い

Windows NT／2000サーバにマッキントッシュサービスを設定するとMacintoshとWindowsでファイル共有などができるように、Linuxマシン上でNetatalkとSambaを動作させると、MacintoshとWindowsでファイル共有することができる。

こうした場合、Windows NT／2000ではMacintoshのリソースフォークファイルなどが見えないのだが、Linux上では見えてしまう。

こうしたファイルをSambaではクライアントから隠すことができる。

また、Samba 2.2.1aからはユーザーにアクセス権限のないファイルを見せなくするオプションも追加された。

■ファイル所有者の取り扱い

Windowsの場合、ディレクトリの下にファイルを作成すると誰でもアクセスできる属性が付くか、上位のディレクトリの属性を引き継ぐのだが、Linuxの場合ファイルは作成者が所有者になるため、ある人が作成したファイルを別の人が更新できないといったことが起きることがある。

Sambaは、共有ファイルの所有者を作成者に関係なく固定ユーザー（グループ）に設定できる。

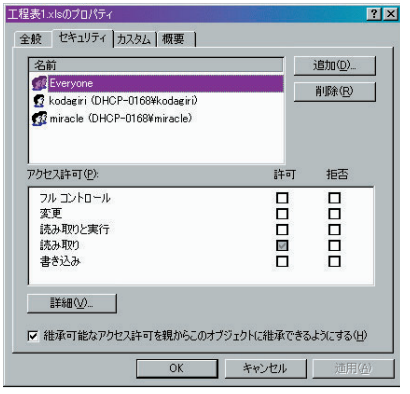
■Windows管理ツール

Windows NTサーバに付属する「ドメインユーザーマネージャ」「サーバマネージャ」「WINSサーバマネージャ」はSambaサーバに対しては利用できない。ただし、一部の機能は参照機能だけ利用できるものもある。

Samba 2.2の新機能

1999年1月に公開されたSamba 2.0も2001年4月にSamba 2.2が公開され、US Sambaチームによる開発は終了してしまっている。

2.2.0リリース当初は2.0.7より品質が良くなかったため、ディストリビューションが採用していなかったが、7月の2.2.1aリリースにより実用レベルになり、9月にTurbolinux 7 Workstationが、10月にRed Hat Linux 7.2がSamba 2.2を採用した。しかし、これらはオリジナルUS版の



画面3　Samba上のファイルのセキュリティタブ

ため、日本語の取り扱いに問題があったり、ドキュメントが英語であったりする。

現在、Samba 2.2.1aの日本語版はミラクル・リナックスで開発したコードをもとに日本Sambaユーザ会で開発されており、10月に発売されたMiracle Linux Standard Edition V2.0で採用されている。

ここでは、Samba 2.2.1aおよびその日本語版の機能を紹介する。

■PDC機能

Samba 2.0ではWindows 95／98／Me／NTクライアントに対し、ドメインログオンを提供し、ログオン時にスクリプトを実行させることができるが、Samba 2.2.1aではこれにWindows 2000も加わった。Windows 2000からSambaへのアクセスはSamba 2.0.7以降で可能であったが、ドメインログオンは従来できなかった（**図5**）。

■Windows 2000／NT系列でのプリンタドライバの自動インストール機能

従来Samba 2.0では、Windows 95／98／Meのプリンタドライバの自動インストール機能が提供されていたが、Samba 2.2ではWindows NT／2000もサポートされるようになった。しかし、プリンタドライバのサーバへの配置方式は従来のWindowsのinfファイルをprinterdef形式に変換する方法も残っているが、Windows NT／2000用のドライバを配置するには、Windowsクライアントを使用した新しい方法で行わなければならない。

■MS-DFSルートのサポート

MS-DFS（マイクロソフト分散ファイルシステム）を使用すると、複数台のSambaサーバ／Windowsサーバを1台の仮想ファイルサーバとしてクライアントに見せることができ、ファイルサーバの追加などの構成変更に柔軟に対応できるようになる。

従来のSamba 2.0でMS-DFSを構築するには、Windows 2000をDFSルートにし、SambaをDFSのツリーにする必要があったが、Samba 2.2はWindows 2000がなくてもDFSルートになり、ツリーを構成することが可能になった（**図6**）。

■hide unreadableのサポート

ユーザーが読めないファイルはユーザーに見えないように隠してしまうオプションをサポートした。

■enhanced browsingのサポート

WINSサーバにNetBIOS名に<1B>副修飾子を持つDMB（ドメインマスタブラウザ）を自動的に登録する、拡張Sambaブラウジング機能をサポートした。ネットワークセグメントを越えたブラウジングを改善するだろう。

■UNICODE対応

Windowsクライアントとの通信コード（wire）がUNICODEになった。

■USRMGR対応

Windowsのドメインユーザーマネ

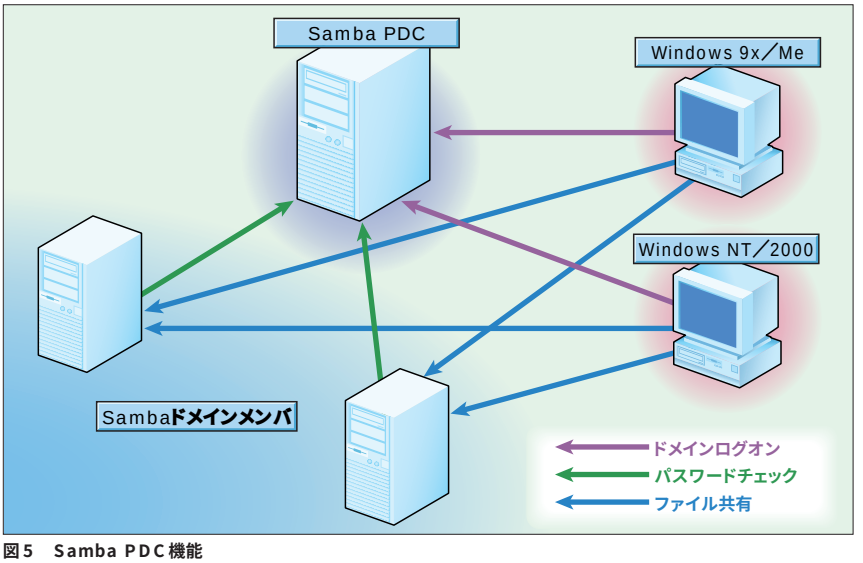


図5　Samba PDC機能

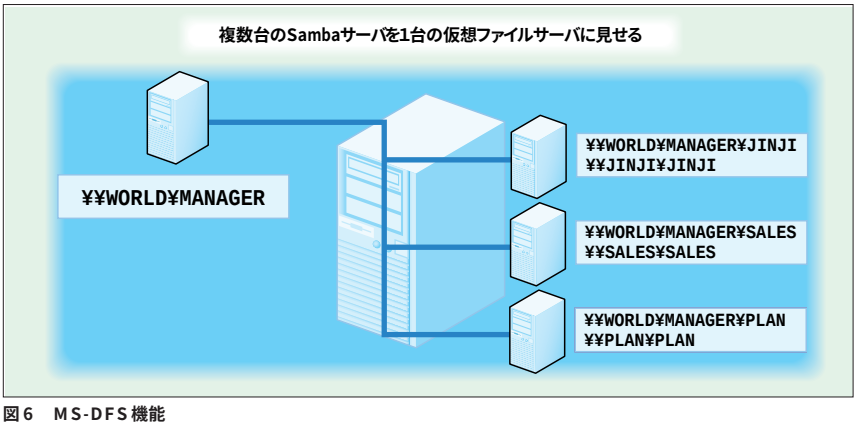


図6　MS-DFS機能
\\JINJI\JINJIというSambaサーバの共有を\\WORLD\MANAGERというDFSルートの下層ツリー\\WORLD\MANAGER\JINJIとして見せることができる。

ージャ（USRMGR.EXE）でSambaのユーザーを見ることが可能になった（設定変更は未対応）。

■libnss_winsによるWINSを使った名前解決の提供

libnss_winsを使用することで、Linuxでのホスト名の名前解決でDNSやhostsファイルに加え、WINSを使用して名前解決ができる。

通常、Linux（UNIX）でのホスト名の名前解決はDNSやhostsファイルやNISが用いられ、WINSを参照しない。

そのため、Windowsのpingコマンドではを参照して名前解決できるマシンも、Linuxのpingではを参照しないため、名前解決できないことがある。LinuxやWindowsサーバをDNSだけで管理すれば問題はないように見えるが、WindowsクライアントをWINSやDHCPのみで管理している場合、LinuxマシンからWindowsクライアントの名前解決ができなかったり、古いWindowsマシンではDNSによるNBT名前解決ができない、動的DNSが利用できないなど、制限事項が多かった。

libnss_winsは、Linuxの機能であるnss（ネームサービススイッチ）機能を使いLinux上でWINSでの名前解決を提供する（**図7**）。

■pam_smbpassによる/etc/passwdとsmbpasswdの統合

pam_smbpassは、Linuxが管理するユーザー管理ファイル/etc/passwdをSambaのユーザー管理ファイルsmbpasswdで統合する機能を提供する。これにより以下の管理作業が軽減される。

- **pam_smbは認証サーバのSambaもしくはWindowsサーバが動作していないと認証できなかったが、pam_smbpassはファイルのみで認証するのでネットワークがダウンしていても認証可能である。**
- **unix password syncを使って、Linuxのパスワードと Sambaのパスワードを同期させる必要がない（unix password syncを使用するには常にSambaが起動している必要がある）。**
- **smbpasswdコマンドだけでパスワードが変更できる。ただし、passwdコマンドで変更しても使用されないし、ユーザーの追加はuseraddしたうえでsmbpasswd -aしなければならない。**

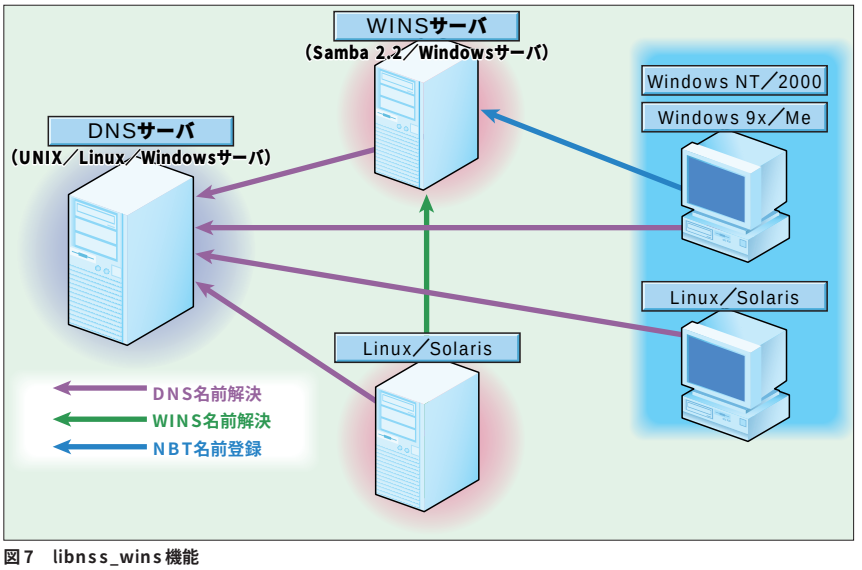


図7　libnss_wins機能
UNIX／Linux／Windowsマシンの名前解決がWINSで統合可能になる。

■Windows oplockとNFS（telnet、FTP）との統合

従来はSambaで更新した内容がNFSクライアントやFTPで参照できないという問題があったが、Linuxカーネル2.4との組み合わせでSambaが提供するoplock（Windowsクライアントにファイルをキャッシュする機構）とNFS（telnet、FTPなどのLinuxマシン上でのアクセス）が、同じデータファイルにアクセスするとき、同じロックが使えるようになる。そのため、NFSとSambaでファイルを共有するときの整合性が保たれるようになった。

ただし、これはLinuxカーネル2.4にoplockパッチを適用する必要があり、日本のディストリビューションで対応しているものはない。

■ACL（アクセス制御リスト）のサポート

POSIX準拠のACLをもとにSambaへWindows NT／2000のようなACLが使えるようにする機能。しかし、これはACLをサポートしたファイルシステム（たとえばSGIのXFS）やLinuxカーネルにACLパッチが必要なため、日本のディストリビューションで対応しているものは現在ない。

■Samba 2.2.1a日本語版の機能

Samba 2.2はSamba 2.0で日本Sambaユーザ会が解決した日本語（マルチバイト）の問題をいくつか取

り入れているため、オリジナルのSamba 2.2でも大きな支障なく運用は可能だ。**表1**にオリジナルのSamba 2.2.1aでの日本語対応状況を挙げる。しかし、機種依存文字などの扱いが完全でなかったり、ドキュメントが英語だったりするため、Samba日本語版を使うのが望ましい。

表2にSamba 2.2.1a日本語版での日本語対応状況を挙げる。

Samba 2.2.1a日本語版ならばsmb.confに以下のような設定をすれば、機種依存文字も外字もサーバ側でもクライアント側でも問題なく日本語を扱えるであろう。

```
coding system = euc
client code page = 932
```

加えてコンピュータ名やワークグループ名、ドメイン名に日本語が使えるようにSamba日本語版では改修しているが、現10月時点ではまだ完全とはいえない。使用はあまり勧めない。

■Windows／Linux／UNIXの認証機能の統合（現在開発中）

次版のSamba 3.0で提供が予定されていたwinbindという新しい認証デーモンがSamba 2.2.2でPDC機能を除く部分が提供された。これは、Windows／Linuxでのユーザー認証を統合し、Windowsクライアントの認証はもちろんPAMとしてNISと同様なLinuxユーザーの認証も可能にする。

従来はSambaでドメインを構築すると複数マシンでのLinuxのUID、GIDの統一が難しかったが、Winbindによって、複数のLinux（UNIX）マシンでのユーザー管理が統合される（**図8**）。

	最大長（バイト）	日本語対応			空白を含む名前
			半角カナ	機種依存文字	
ファイル名	255 （注1）	○	○	×	○
ディレクトリ名	255 （注1）	○	○	×	○
共有名	12 （注2）	○	○	×	○
コンピュータ名	15 （注3）	×	×	×	×
Windowsワークグループ名	15 （注3）	×	×	×	×
Windowsドメイン名	15 （注3）	×	×	×	×
説明（詳細／コメント）	47 （注4）	○	○	×	○
ユーザー名	12 （注5）	×	×	×	×
グループ名	20	×	×	×	×
パスワード	14 （注6）	×	×	×	×

表1　オリジナルSamba 2.2.1aの日本語対応
（注1）共有名、ディレクトリ名を含めたファイルパスの最大長も255バイト。coding system=hex／capを使用すると漢字1文字が6または4バイトになるため、最大長は短くなる。
（注2）Windows NT／2000ではもっと長い共有名が使用できるが、Windows 95／98／Me／3.1でアクセスできる共有名の最大長は12バイト。
（注3）コンピュータ名にピリオドを入れるとSambaが起動しないことがある。ワークグループ名、ドメイン名もピリオドを入れないほうがいい。
（注4）コンピュータの説明を48バイト以上にするとWindowsの問題でネットワークコンピュータが見えなくなることもある（マイクロソフト文書番号J041615より）。
（注5）Windowsでは数字のみのユーザー名が作成できるが、Linux（UNIX）では数字のみのユーザーは作成できない。同様に日本語や空白を含むユーザー名もLinuxでは使用できない。ただし、user name map fileを使用すると使うことができる。
（注6）encrypt passwords=noとした場合やLinux（UNIX）とのパスワード同期機能を使用する場合、使用できる文字や長さはLinuxと同様の制限を受ける。古いLinuxでは8バイトのものもある。

	最大長（バイト）	日本語対応			空白を含む名前
			半角カナ	機種依存文字	
ファイル名	255 （注1）	○	○	○	○
ディレクトリ名	255 （注1）	○	○	○	○
共有名	12 （注2）	○	○	×	○
コンピュータ名	15 （注3）	○	△	×	×
Windowsワークグループ名	15 （注3）	△	△	×	×
Windowsドメイン名	15 （注3）	△	△	×	×
説明（詳細／コメント）	47 （注4）	○	○	○	○
ユーザー名	12 （注5）	△	△	△	○
グループ名	20	×	×	×	×
パスワード	14 （注6）	×	×	×	×

表2　Samba 2.2.1a日本語版の日本語対応

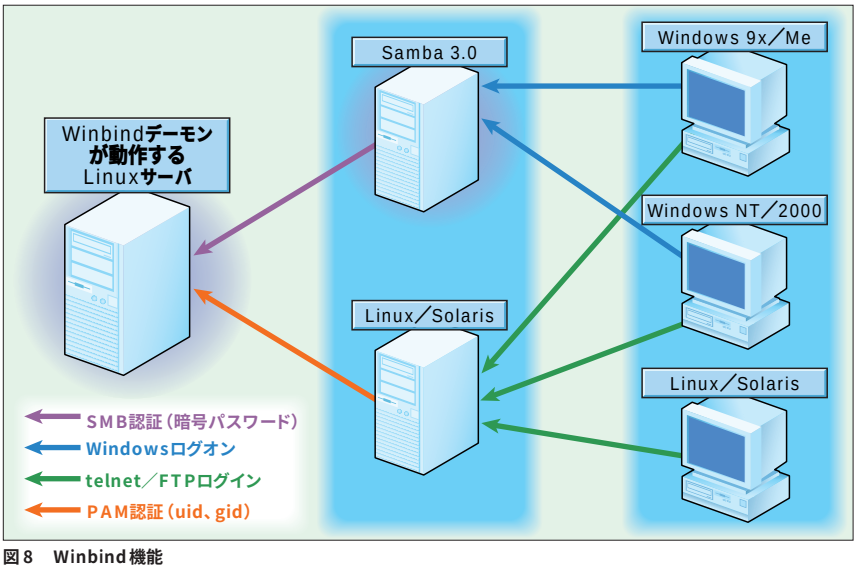


図8　Winbind機能
Windowsドメイン認証やUNIXの認証がすべてWinbindで統合可能になる。

Borland Kylixで始める

Linuxビジュアルプログラミング

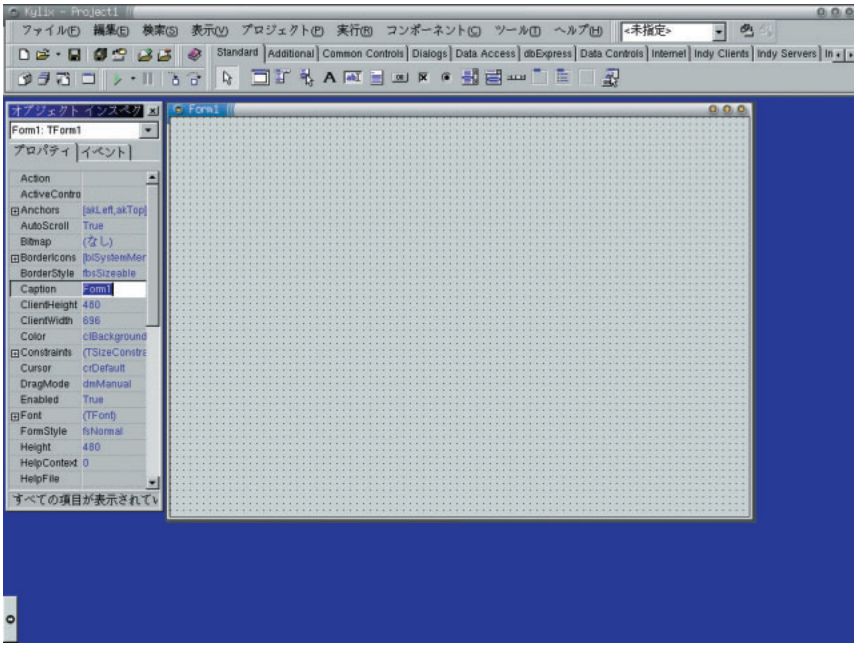
第1回　コンポーネントの使い方を覚えよう（1）

Borland Kylixは、Windows上の開発環境として人気を博している「Borland Delphi」のLinux版で、すぐれたビジュアル開発環境を提供します。部品を組み合わせる感覚でプログラミングができる点が特徴です。今回は、Borland Kylixの簡単な使い方とビジュアル開発の基礎となるコンポーネントについて解説します。

文：井上 勉
Text : Tsutomu Inoue

Borland Kylix（以下、Kylix）は、ビジュアルな環境において、短時間でLinuxアプリケーションを開発できるツールです（**画面1**）。Kylixには、フォームデザイナやオブジェクトインスペクタなどのビジュアル設計ツールや、頻繁に使われるような処理や、関連するデータをカプセル化した形態のプログラム部品を多数用意しています。そのため、フォームに「コンポーネント」と呼ばれるプログラム部品を配置し、コードエディタで最小限のコードを記述するだけで、実用的なアプリケーションが極めて短時間で作成できます。

Kylixは、Windows上の開発ツール「Borland Delphi」を元にしています。Borland Delphi（以下、Delphi）は、わかりやすいビジュアル開発機能と、高速なネイティブコードコンパイラを両立させた、Windows上では定評ある開発ツールです。このDelphiのプログラミングスタイルと高い生産性をLinux上のプログラミングの世界でも実現させた開発ツールがKylixです。



画面1　Kylix Server Developer Editionの統合開発環境（IDE）

Object Pascal について

Kylixは、プログラム言語として「Object Pascal」を採用してます。Object Pascalは、Delphiを使用したことがある方以外にはあまりなじみがないかもしれません。Object Pascalという言語は、一般的によく知られたプログラミング言語「PASCAL」に、オブジェクト指向言語としての機能を拡張したものです。したがって、基本的な文法はPASCALに基づいています。

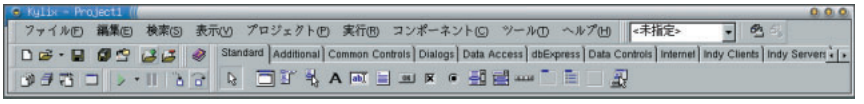
初期のPASCALは教育用の言語として考案され、その機能はあまり高いものではありませんでした。しかし、ボーランドは教育用であったPASCALを、マイクロコンピュータやPC上で幅広く使うことのできる汎用プログラミング言語にするために、

さまざまな拡張を行ってきました。その過程で、分割コンパイルやオブジェクト指向プログラミングといった機能に関する拡張も行われてきました。オブジェクトの永続化や実行時型情報のような、ビジュアル開発ツールを構築するうえでの重要な機能の拡張は、1995年に最初のDelphiである「Delphi for Windows」に合わせて行われたものです。また、Kylixに搭載された最新バージョンのObject Pascalは、ボーランドのPASCAL言語の集大成とも言えるものです。

Object Pascal言語は、「厳格で洗練された構文を持つ」という、元々のPASCAL言語の特長を引き継いでいますが、それにとどまることなく、C言語と同等以上の細かいプログラミングを行うための機能や、C++言語に匹敵する柔軟性と拡張性を持たせた実用的な言語になっています。よって、プログラミングの初心者はもとより、すでにC/C++言語で高度なプログラミングを行っているプログラマーに対しても、必要十分な機能を提供します。また、言語の習得のしやすさと実用性（高速コンパイルの実現など）の向上のために、C++言語で一般に難解で扱いにくいとされている機能（プリプロセッサ命令、演算子の多重定義、クラスの多重継承、独立参照、テンプレート）は搭載していません。

コンポーネントの概要

コンポーネントと呼ばれるプログラム部品は、Kylixでさまざまなプログラミングをするうえで使用できるオブジェクトです。Kylixのコンポーネントは、プログラムの設計時にビジュアルに扱えることが特徴です。そのため、プログラムのユーザーインターフェイスの外観や、プログラムの動作の一部を、設計時にビジュアルに確認しながらプログラミングできます。



画面2　コンポーネントパレット

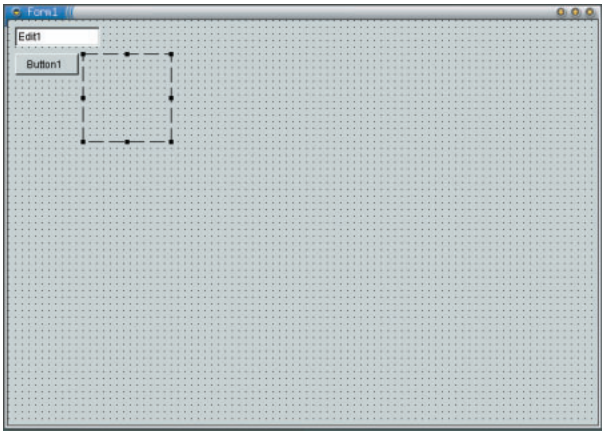
コンポーネントは、Kylixのメインウィンドウにあるコンポーネントパレットに配置されています。コンポーネントパレットには、コンポーネントがカテゴリー別にページ分けして格納されています（**画面2**）。

これらのコンポーネントは、プログラムの設計時にフォームと呼ばれるコンテナオブジェクトに貼り付けて使うことができます（**画面3**）。

CLXとは

Kylixで使用することのできるコンポーネントの集まりを、「CLX（クリックス）」と呼んでいます。CLXは「Component Library for Cross-Platform」という意味を持っており、元々Windows上のDelphiのコンポーネントライブラリであった、「VCL」のコンセプトを元に、Kylixのために新たに開発されたものです。VCLは、Windows APIやWindows上のさまざまなミドルウェアの機能をカプセル化した、いわば「Windows依存の」フレームワークですが、CLXはLinuxとWindowsでソースコード互換のクロスプラットフォーム開発を実現するためのフレームワークです。KylixはCLXを搭載する最初のツールとして登場しましたが、Windows上のDelphiの最新バージョンであるDelphi 6には、従来のVCLと共にCLXも搭載されています。そのため、DelphiとKylixのプログラマーは、WindowsとLinuxの上でまった同じスタイルで、また同一のソースコードで双方のプラットフォーム向けのアプリケーションの開発が可能になります。これは、ネイティブコードコンパイラを搭載したビジュアル開発ツールとしては画期的なことです。

なおCLXは、Kylixのプログラム言語であるObject Pascal言語でそのほとんどすべてが記述されています。つまり、Kylixがそれ自身のみで、ツールに対するほとんどすべてのカスタマイズが可能であることを意味します。これは、Kylixが非常にポテンシャルの高い開発ツールであることの裏付けになっています。



画面3　フォームに貼り付けたコンポーネント

このCLXは、大きく表のように分類されます。

Kylixの基本的な操作

Kylixで、フォームやコンポーネントを利用したビジュアルプログラミングを行うとき、最初のステップとして、まずフォーム上に必要なコンポーネントを配置します。

プロパティの設定

フォームや、フォーム上に配置されたコンポーネントは、オブジェクトインスペクタの［プロパティ］ページでプロパティを設定することができます。設計時にプロパティを設定することにより、フォームやコンポーネントの外観などの基本的な属性を設定できます（**画面4**）。

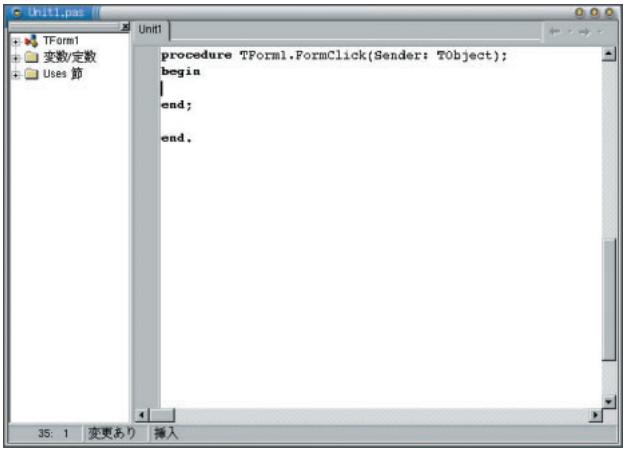
プロパティの値の設定方法は、プロパティの型によっていくつかのパターンがあります。「Caption」のような文字列型のプロパティや、「Height」のような整数型のプロパティは、直接キーボードで値を設定します。また、Colorプロパティのように、ドロップダウンリストから値を選択できるものもあります。プロパティ名の前に「＋」が表示されているプロパティは、「サブプロパティ」を持ちます。「サブプロパティ」とは、プロパティそのものがオブジェクトであるか、もしくは集合型（論理値の集合に対して、要素ごとにTrue/Falseを設定する型）である場合に、そのプロパティに含まれる要素に対して個別に値を設定できる状態を意味します。

このほか、プロパティの値の設定欄をクリックすると「…（省略記号マーク）」が表示されるものは、そのプロパティを設定するための専用のダイアログボックスが用意されていることを意味します。これには、ビットマップ（TBitmap）、ピクチャー（TPicture）、フォント（TFont）、文字列リスト（TStrings）のような型のプロパティがあります。

画面4　オブジェクトインスペクタでのプロパティの設定

イベントハンドラの作成

Kylixは、主にイベントドリブン型のGUIアプリケーションをターゲットとしているので、作成されるアプリケーションのほとんどすべてのコードは、



画面5　コードエディタで表示されたイベントハンドラのスケルトンコード

分類名	説明
VisualCLX	ネイティブなクロスプラットフォームGUIコンポーネントと、グラフィックスの機能が含まれる。この部分では内部的にTroll TechのQtライブラリを使って、LinuxとWindowsのGUIシステムの違いを吸収している
DataCLX	クライアント側でのデータアクセスの機能が含まれる。この部分はLinuxとWindowsで共通
NetCLX	Apache DSOやCGI Webブローカーを含む。この部分もLinuxとWindowsで共通
RTL	RTL（Run-Time Library）のうち、Classes.pas以上の部分。この部分のコードはWindowsとLinuxで共通（RTLのうち、Classes.pasより下の低レベルのコードの一部はプラットフォーム依存のアセンブリ言語で記述されている。この部分はCLXには含まれないとされる）

表　CLXの分類

何らかの「イベント」の発生をきっかけとして、直接的、または間接的に実行されます。この場合の「イベント」とは、ユーザー、またはシステムが発生させるさまざまなアクションに対して、特定の処理を関連付けできる仕組みのことを指しています（もちろん、Kylixはコンソールやターミナルウィンドウで実行するような、Linuxで一般的なキャラクタベースのプログラムも作成できます）。

このイベントを処理するためのコードはイベントハンドラと呼びます。イベントハンドラを作成するには、オブジェクトインスペクタの［イベント］ページで、イベント名の右の空欄をダブルクリックします。これによって、コードエディタが自動的に開かれ、イベントハンドラのスケルトンコード（空枠）が表示されます。**画面5**は、フォームオブジェクトのOnClickイベントハンドラが、コードエディタで表示されたところです。

コードエディタに表示されたイベントハンドラのスケルトンは、Kylixが自動的に生成したものです。フォームオブジェクトのOnClickイベントハンドラのデフォルトの手続き名は「FormClick」ですが、変更は可能です。イベントハンドラの手続き名を変更するには、オブジェクトインスペクタのイベントページで、該当するイベントハンドラの名前を修正します。

通常、イベントハンドラはフォームクラス（デフォルトではTForm1クラス）のメンバメソッドとして自動的に宣言されます。コードエディタを上方にスクロールすると、その宣言を見ることができます（**リスト1**）。キーワード「type」は、C/C++言語の「typedef」に相当します。**リスト1**の宣言は、フォームの基本クラス「TForm」を継承した新しいクラス、「TForm1」を宣言するものですが、Kylixでフォームをビジュアルにデザインしているときは、そのような文法的なことにあまり注視する必要はありません（しかし、無作為に修正したりすると、コンパイルエラーが発生します）。

ここで、イベントハンドラ「FormClick」のbegin～end内に、**リスト2**のようなコードを記述してみます。

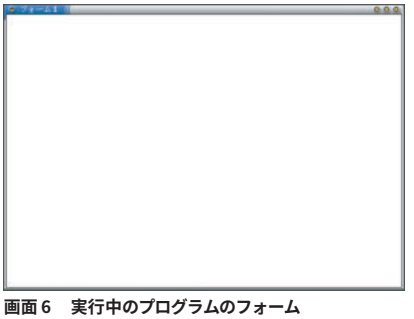
リスト2のコードは、フォームのColorプロパティの値を状況に応じて変更するコードです。Object Pascalでは、「：＝（コロンイコール）」の記号は代入演算子で、if文の分岐条件で使われている「＝（イコール）」は、両側の値が等しいかどうかを判定するための演算子になります。Object PascalにはC/C++言語の「＝＝（連続する2個のイコール）」という演算子はありません。また、フォームのColorプロパティは、もちろんフォームの設計時にオブジェクトインスペクタで値を設定することもできますが、このようにコードから値を代入して設定することもできます。ここで使われている「clBackground」や「clWhite」のような名前は、Colorプロパティに設定できる値として、あらかじめKylixで定数宣言されているものです。これらの値の一覧は、Colorプロパティのドロップダウンリストで確認できます。

プログラムのコンパイルと実行

作成したプログラムを実行するには、キーボードのF9キーを押すか、ツールバーの［実行］ボタンをクリックします。F9キーは、メニューバーの［実行］メニューの［実行］コマンドに付けられたショートカットキーです。［実行］コマンドによって、ソースプログラムと関連するフォームは自動的にコンパイル＆リンクされ、ELF形式の実行可能ファイルが生成されます。

また、プログラムの実行は行わず、コンパイルのみを行いたい場合は、キーボードのCtrl＋F9キーを押します。Ctrl＋F9キーは、メニューバーの［プロジェクト］メニューの［コンパイル］コマンドに付けられたショートカットです。

先ほどのイベントハンドラを記述したプログラムを実行し、実行中のプログラムのフォームをクリックすると、



画面6　実行中のプログラムのフォーム

▼リスト1　TFormクラス宣言

```
type
  TForm1 = class(TForm)
    procedure FormClick(Sender: TObject);
  private
    { Private 宣言 }
  public
    { Public 宣言 }
  end;
```

▼リスト2　フォームオブジェクトのOnCreateイベントハンドラ

```
procedure TForm1.FormClick(Sender: TObject);
begin
  if Color = clBackground then
    Color := clWhite
  else
    Color := clBackground;
end;
```

フォームの表面の色が変化します（**画面6**）。

プログラムの終了

Kylixの統合開発環境（IDE）から実行したプログラムを終了させるには、通常どおりにウィンドウのタイトルバー右端の「閉じるボタン」をクリックします。プログラムミスによりプログラムが無限ループしてしまったり、デバイスドライバのエラーなど、さまざまな理由によりプログラムがフリーズしたときには、メニューバーの［実行］メニューの［プログラムの終了］を選択します。しかし、この方法でプログラムを終了させた場合、フリーズしたプログラムをデバッガから強制終了させているので、そのプログラムが実行時に確保していたリソースが完全に解放できない場合があり、それ以後、Kylixの動作が不安定になることがあります。KylixのIDEから実行したプログラムを強制終了させた場合は、速やかにプログラムを保存してKylixを再起動することをお勧めします。

作成したプログラムの保存

作成したプログラムを保存するには、メニューバーの［ファイル］メニューから［プロジェクトに名前を付けて保存］を選択します。初めてプログラムを保存する場合は、作成したプログラムに含まれるソースコードのディレクトリ位置とファイル名を指定して保存できます。このとき保存するファイルのうち、名前を指定しなければならないのは、「ユニットファイル」と「プロジェクトファイル」です。ユニットファイル（拡張子.pas）は、フォームと関連するソースファイルで、フォームと同じ数だけ存在します（デフォルトでは1個）。プロジェクトファイル（拡張子.dpr）は、作成したプログラムに必要なすべてのソースファイルの関連情報と、プログラムのエントリポイントを含むメインソースファイル（**リスト3**）で、1個のプログラムにつき必ず1個だけ作成されます。通常、コンパイル＆リンク後の実行形式のファイル名は、このプロジェクトファイルの名前から拡張子「.dpr」を除いたものになります。

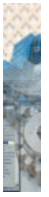
一度保存したプロジェクトに対して修正を行ったあとで、再度保存したい場合は［ファイル］メニューから［すべて保存］を選択します。

▼リスト3　プロジェクトファイル

```
program Project1;

uses
  QForms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

▼リスト4　Button1のOnClickイベントハンドラ

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Close;
end;
```



目的別のコンポーネントの利用

Kylixのコンポーネントは、コンポーネントパレット上で、カテゴリー別に分類されて配置されていますが、ここではプログラミング上の処理目的別にコンポーネントを分類して、利用方法を見ていきます。

ボタンを使う

「ボタン」のようなコンポーネントは、もっとも頻繁に利用されるコンポーネントのカテゴリです。ボタンはクリック操作をきっかけにして、何らかの処理を実行するために使います。

■Buttonコンポーネント

ボタンとして機能するコンポーネントのうち、一番標準なものはコンポーネントパレットの［Standard］ページにある「Button」コンポーネントです。「Button」のCaptionプロパティは、ボタンの表面（フェイスと呼ばれる）に表示する表題や「キーアクセラレータ」を設定できます。キーアクセラレータとは「&（アンパサンド）」とそれに続く1文字を組み合わせ、実行時にAltキーを押しながら、その文字のキーが押されたときにOnClickイベントを発生させる機能です。

たとえば、フォーム上に［Standard］ページの「Button」コンポーネントを配置し、OnClickイベントハンドラを**リスト4**のように記述します。

リスト4のコードは、フォームオブジェクトのCloseメソッドを呼び出し、フォームを閉じるためのコードです。

このプログラムを実行し、実行中のプログラムのフォーム上の「Button1」をクリックすると、フォームが閉じられ、プログラムは終了します。Kylixでは、デフォルトで用意される最初のフォームを「メインフォーム」と呼び、メインフォームが閉じられるとプログラムは自動的に終了します。

またボタンには、［Standard］ページの「Button」コンポーネントのほかに、［Additional］ページの「BitBtn」と「SpeedButton」があります（**画面7**）。

■BitBtn

まず「BitBtn」ですが、このコンポーネントの特長はKindプロパティがあることです。Kindプロパティは、あらかじめ用意された選択肢（bkAbort、bkAllなど）から特定の値を設定すると、設定した値に応じたビットマップと動作が設定されるボタンです。たとえば、Kindプロパティに「bkClose」を設定すると、［閉じる］ボタンのためのビットマップと動作がボタンに設定されます。

■SpeedButton

「SpeedButton」は、ウィジェット（ウィンドウ）ではなく、完全にグラフィックスとして実装されているボタンで、キーボードフォーカスを受け取ることはできませんが、軽量でリソースの消費が抑えられることが特長です。SpeedButtonは、ボタンバーのような数多くのボタンを並べるユーザーインターフェイスを作成するときに便利です。このSpeedButtonの特長として、「ボタンのグループ化」機能があげられます。複数のSpeedButtonのGroupIndexプロパティを「0」以外の値に設定すると、実行時にそれらの複数のSpeedButttonのうち、1つだけが押された状態にすることができます。特定のボタンが押された状態にするには、Downプロパティを Trueに設定します。

これら3種類のボタンは、それぞれ「Caption」プロパティと「OnClick」イベントを持っており、使い方自体は非常に似ています。どの種類のボタンを選択するかは、ボタンにどのような振る舞いや使い勝手を求めるかに依存します。

ボタンにはこれらのほかに、［Common Controls］ページの「ToolBar」コンポーネントの上にのみ配置できる「ToolButton」があります。ToolButtonは、フォーム上にToolBarコンポーネントを貼り付けてから、ToolBarを右クリックしてポップアップメニューを表示させ、ここから［ボタン新規作成］を選択することにより生成します（**画面8**）。

画面にテキストを表示・入力する

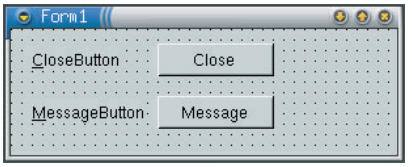
画面にテキストを表示するコンポーネントは非常に数多くあるので、ここではすべて紹介することはできません。よく使うものと特徴あるものだけを取りあげて紹介します。

■Label

フォーム上に単純なテキストを表示するには［Standard］ページの「Label」コンポーネントを利用します。Labelコンポーネントは、フォームやButtonと同じようにCaptionプロパティを設定することで文字列を表示できます。Captionプロパティにはキーアクセラレータを設定することも可能で、実行時に押されたキーアクセラレータは、FocusControlプロパティで関連付けられたコンポーネントにフォーカス移動するときに有効になります。**画面9**は、2個のLabelコンポーネントのCaptionプロパティにキーアクセラレータを含む文字列を設定し、FocusControlプロパティでButtonコンポーネントに関連付けたものです。

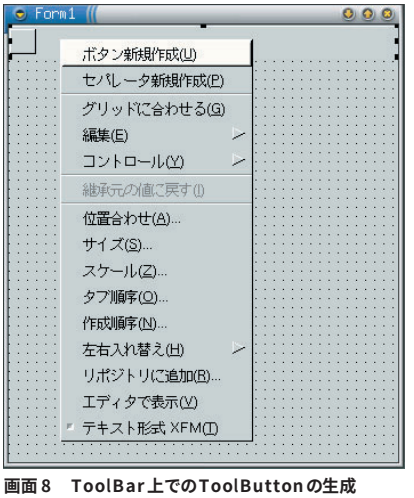
画面9のCaptionが「CloseButton」に設定されたLabelのFocusControlプロパティは、［Close］ボタンに関連付けられており、Captionが「Messa

画面7　「Button」「BitBtn」「SpeedButton」



画面9　LabelコンポーネントとFocusControlプロパティの関連付け

画面10　EditコンポーネントのEchoModeプロパティを「emPassword」に設定



画面8　ToolBar上でのToolButtonの生成

geButton」に設定されたLabelのFocusControlプロパティは、［Message］ボタンに関連付けられています。このとき、実行時のフォームの上でAlt＋Cキーを押すと、キーボードフォーカスは［Close］ボタンに移動します。Alt＋Mキーを押すと、フォーカスは［Message］ボタンに移動します。

■Edit

単一行のテキストを入力、または表示することに利用できる、最も一般的なコンポーネントが［Standard］ページの「Edit」コンポーネントです。Editコンポーネントは、実行時にキーボードから入力された文字列がTextプロパティに格納されます。また、Textプロパティに文字列を設定すると、設定された文字列がそのまま表示されます。また、EchoModeプロパティを「emNone」や「emPassword」に設定することにより、入力された文字をまった表示しないようにしたり、「＊（アスタリスクマーク）」で隠したりすることができます（**画面10**）。また、ReadOnlyプロパティをTrueに設定することにより、読み取り専用にすることもできます。

■Memo

［Standard］ページの「Memo」コンポーネントは、Editコンポーネントが常に単一行のテキストを扱うのに対して、複数行のテキストを入力したり表示することができます（**画面11**）。

Memoコンポーネントは、Editコンポーネントと同様なTextプロパティのほかに、Linesプロパティによっても表示中のテキストにアクセスできます。Linesプロパティは、「TStrings型」のプロパティと呼ばれ、この型は改行コードで区切られた複数行の文字列で構成される、文字列リストを扱うためのオブジェクトとして実装されています。Linesプロパティから、リスト中の単一行の文字列にアクセスするには、Stringsプロパティを利用します。たとえば、**リスト5**のように記述します。

このコードは、Memoコンポーネントで扱われる複数行のテキストのうち、最初の行の文字列をLabelに表示するコードです。また、このコードは**リスト6**のように記述してもまったく同じ意味を持つので、通常はそのように記述します。

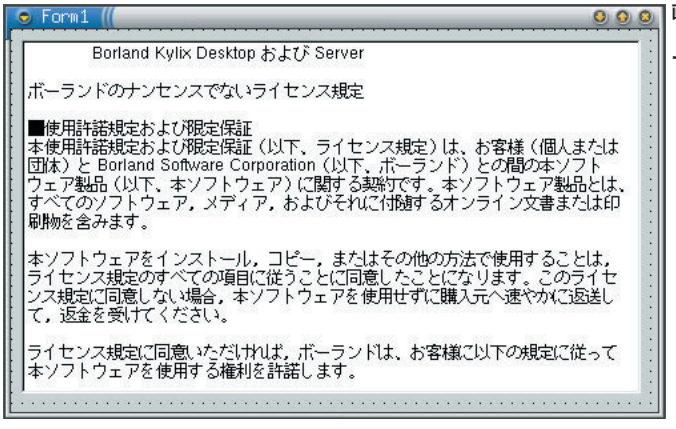
これはStringsプロパティが、TStrings型の「デフォルトプロパティ」として定義されているためです。Kylixの場合、「デフォルトプロパティ」とは、プロパティ名が省略できるプロパティであり、必ずインデックス付けが必要な配列型のプロパティになります。

またTStrings型は、文字列リストの内容をテキストファイルへ書き込んだり、また逆にファイルから読み込んだりするためのメソッドである「SaveToFile」や「LoadFromFile」を持っています。そのため、Memoコンポーネントで扱うテキストをファイルに書き込んだり、ファイルから読み込んだりするのは非常に簡単です。SaveToFileとLoadFromFileの2つのメソッドは、唯一のパラメータとしてファイル名を必要とします。たとえば、Memoコンポーネントに表示中のテキストをファイル「MyProg.txt」へ書き込むコードは、**リスト7**のようになります。

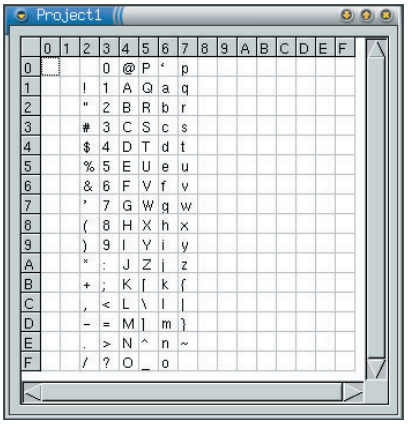
なお、Object Pascalでは文字列や文字を「'（単引用符）」でくくります。

■StringGrid

［Additional］ページの「StringGrid」コンポーネントは、表計算ソフトのような2次元の表形式に並んだ



画面11　複数行のテキストを扱えるMemoコンポーネント



画面12　StringGridコンポーネント

▼リスト5　Memo1で表示中のテキストの最初の行をLabel1に表示

```
Label1.Caption := Memo1.Lines.Strings[0];
```

▼リスト6　デフォルトプロパティ「Strings」を省略

```
Label1.Caption := Memo1.Lines[0];
```

「セル」に、テキストを表示したりキーボードから入力したりできるコンポーネントです（**画面12**）。

StringGridコンポーネントの「セル」は、Cellsという2次元の配列型プロパティによってアクセスできます。CellsプロパティはCells[Column, Row]というように、最初に列番号、あとに行番号を指定する形で特定のセルのテキストを取得したり設定したりします。たとえば、Editコンポーネントから StringGridの列番号「1」行番号「2」のセルにテキストを設定するコードは**リスト8**のようになります。列番号と行番号の開始点は、ともに「0」です。この特性は、Kylixのすべての配列型プロパティの整数インデックスの指定方法に共通するものです。

StringGridの列数や行数を指定するには、ColCountやRowCountのようなプロパティを使います。StringGridの（デフォルトでは灰色の）固定セルの列数や行数を設定するには、FixedColsとFixedRowsプロパティを設定します。また、StringGridのセルはデフォルトでは読み取り専用で、キーボードから入力できませんが、これを可能にするにはOptionsプロパティのgoEditingサブプロパティをTrueにします。Optionsプロパティのサブプロパティには、goEditingのほか、Tabキーによってセル間を移動可能とするためのgoTabsや、特定の行や列全体の選択を可能とするためのgoColSelect、goRowSelectがあります（**画面13**）。

■SpinEdit

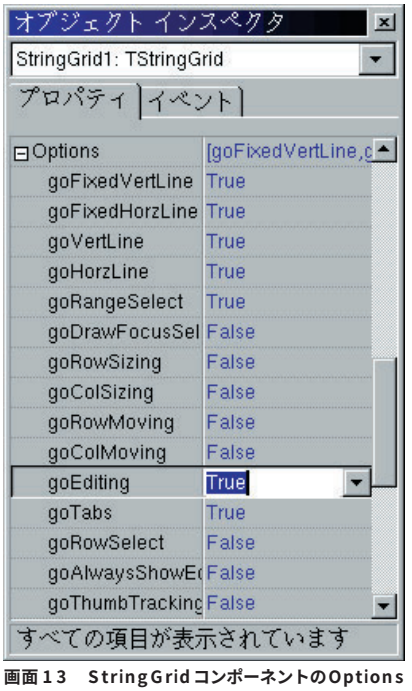
［Common Controls］ページの「SpinEdit」コンポーネントは、整数値を入力するための便利なコンポーネントです。SpinEditには、整数型のValueプロパティがあり、表示する値を整数値で取得したり設定したりできます。SpinEditには、値を増減するための矢印があり、これをマウスやキーボードの↑、↓キーによって操作することができます。また、通貨記号の特殊な文字を数値の前後に表示するためのPrefixや、Sufixのようなプロパティもあり非常に便利です（**画面14**）。

■TextViewer、TextBrowser

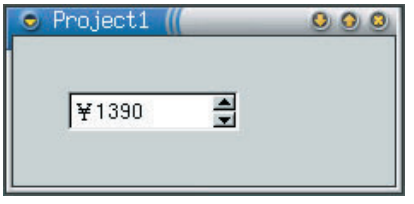
［Common Controls］ページの「TextViewer」コンポーネントと「TextBrowser」コンポーネント（**画面15**）は、ともにプレーンなテキストファイルや、HTML形式のハイパーテキストを表示できるコンポーネントです。2つのコンポーネントの大きな違いは、「TextViewer」コンポーネントが単にファイルを表示するだけなのに対して、「TextBrowser」コンポーネントは、表示されたハイパーテキストに含まれたハイパーリンクをクリックしてたどってゆくことができる点です。2つのコンポーネントとも、表示するためのファイルを指定するには FileNameプロパティを使います。なお、TextBrowserだけには、「Backward」「Forward」「Home」のようなハイパーテキストをナビゲートするためのメソッドがあります。

●

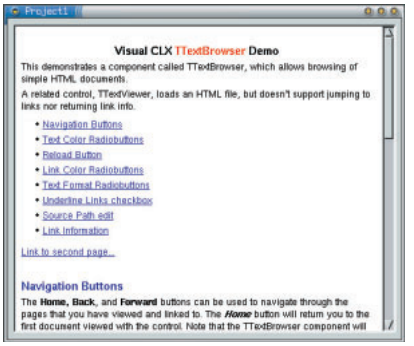
今回は、Kylixの概要とCLXコンポーネントの基本的な利用法を紹介しました。このように、Kylixではコンポーネントをフォームに配置することで、簡単にインターフェイスを設計することができます。次回は、選択肢から候補を選択するリストコンポーネント、グラフィックスを扱うコンポーネントの利用法を紹介します。



画面13　StringGridコンポーネントのOptionsプロパティ



画面14　SpinEditでPrefixプロパティを「¥」に設定



画面15　TextBrowserコンポーネント

▼リスト7　文字列リストのSaveToFileメソッドの呼び出し

```
Memo1.Lines.SaveToFile('MyProg.txt');
```

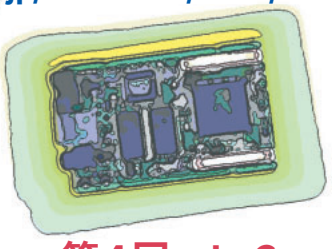
▼リスト8　StringGridの列番号「1」行番号「2」のセルにテキストを設定する

```
StringGrid1.Cells[1, 2] := Edit1.Text;
```

レーザーファイブ L-Card+

カード型Linuxマシンで遊ぼう

レーザーファイブ　L-Card+
http://www.laser5.co.jp/embedded/lcard/



文：吉田 功
Text : Isao Yoshida

第4回　L-Card+を赤外線リモコンにしてみよう（前編）

「家にあるビデオやエアコンを外出先からコントロールしたい！」と思ったことはないでしょうか？　そこで今回は、L-Card+を使って、LANやインターネットからビデオやエアコンを操作できる赤外線リモコンを作ってみることにしました。

L-Card+なら、ずっと電源を入れっぱなしにしておいても消費電力は少ないですし、ハードウェア仕様が詳しく公開されているので、赤外線LEDなどの部品の取り付けやコントロールが簡単にできます。リモコン機能をもつハードウェアをL-Card+に接続すれば、telnetなどでインターネットやLANを介した遠隔操作も容易にできそうです。

赤外線リモコンの動作原理

赤外線リモコンは、赤外線を飛ばしているらしいことは想像つくのだけど、具体的にどういう仕組みで動いているかを知っている人は少ないでしょう。そこで、赤外線リモコンを作る前に、動作原理を調べてみましょう。

写真1は、テレビの赤外線リモコンを分解して、中から取り出した基板です。先端にレンズが透明なLEDが付いているのがわかるでしょう。このLEDを光らせてテレビをコントロールして

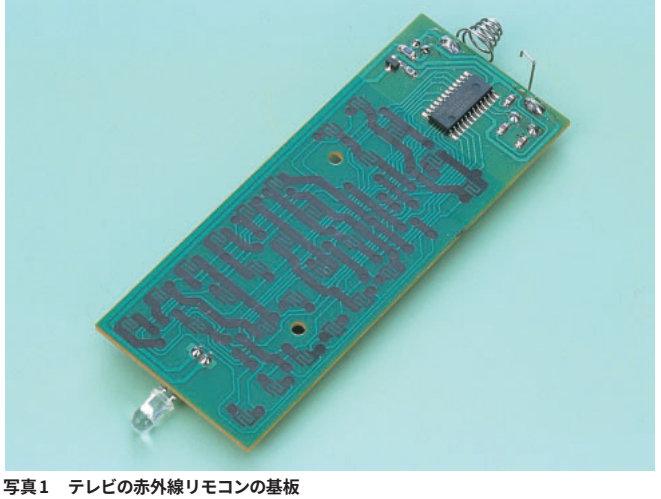


写真1　テレビの赤外線リモコンの基板

写真2　デジカメで赤外線LEDの状態を見る
肉眼では見えない赤外線LEDの状態を確認できる。

いるようです。

このLEDは可視光ではなく、赤外線を出すために、光っているようすを肉眼で確認することはできません。しかし、デジカメを使ってこの赤外線LEDを見ると、**写真2**のように紫色に光っていることがわかります（機種によっては緑など、別の色になる場合もある）。どうやら点滅しているようで、これによって何かの信号を送っていると思われます。

じっとLEDを見ていても信号の内容はわかりませんので、信号の波形を見ることができるオシロスコープという装置をつないでみました。**図1**がその出力です。線が上のほうにあるときにLEDが点灯し、下のほうにあるときは消灯している状態を示しています。

この図を見ると、小刻みに上下（点滅）している状態と、しばらく消灯している状態があることがわかります。小刻みに点滅している間隔は約25μ秒（40kHz）ほどで、点滅状態と消灯している状態の幅は約600μ秒程度です。もちろん、こんな高速に点滅している状態が目で見えているわけではなく、このような点滅／消灯の信号が数10m秒程度続いて、さらに、この信号が1秒間に数回程度発生しているようです。

■赤外線リモコンを作るには？

L-Card＋を赤外線リモコンにするときも、本物の赤外線リモコンと同じように、赤外線LEDを使えばよいでしょう。しかし、リモコンが送出するデータフォーマットがわからないとテレビなどをコントロールをすることができません。リモコンには数多くの種類があり

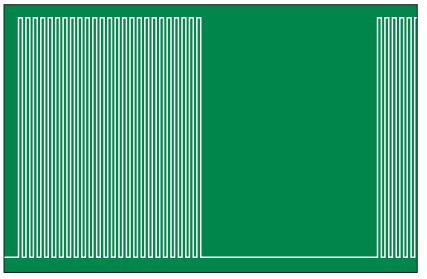


図1　オシロスコープで見た赤外線リモコンのLEDの状態
HI＝点灯、LOW＝消灯を示している。

Column

オシロスコープってどんな装置？

オシロスコープは信号の波形を目で見られる形で表示してくれる測定器です。たとえば、同じ周波数（音程）の音でもいろいろな音があり、それらを比べるのに手軽な測定器のひとつです。今回は、リモコンが送信する信号の波形を表示させましたが、目や耳では区別できないような信号を比べるにはもってこいの装置です。

Column

赤外線リモコンの周波数

赤外線リモコンの信号は、信号を単純なON／OFFで送るわけではなく、40kHz程度で高速に点滅している状態（ON／OFF）と、完全に消えている状態（OFF）という2つの状態を組み合わせて送信されています。なぜ、こんなことをするのでしょうか？

世の中には、赤外線リモコン以外にも赤外線を出すものがたくさんあります。たとえば、太陽光や白熱電灯などがそうです。これらは、赤外線リモコンのLEDよりも強い赤外線を出します。たとえ赤外線リモコンの受信部に当たらなくても、反射光だけでかなりの光量になるほどです。しかし、これらが40kHzという高い周波数でON／OFFを繰り返すことはありません。赤外線リモコンは、高い周波数の点滅があってはじめて制御信号だと認識することで、外来の赤外線による誤動作を回避しているのです。

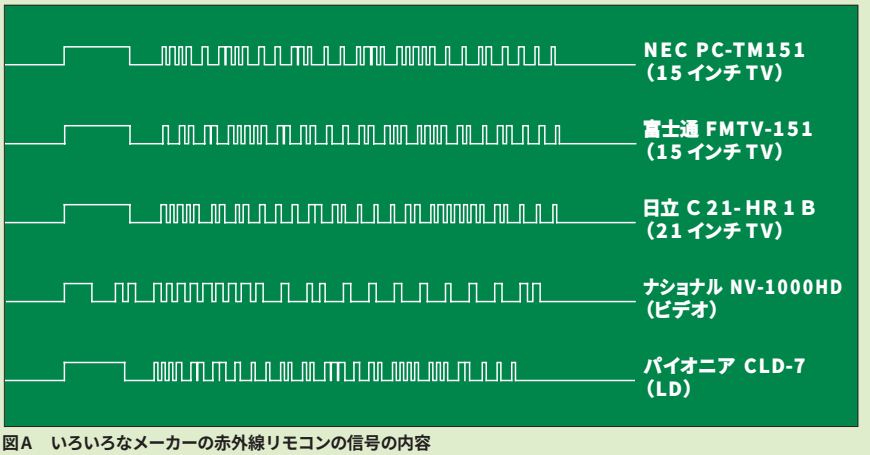
Column

データフォーマットのいろいろ

テレビやビデオ、エアコンなど、赤外線リモコンを使う機器が1つの部屋に何台もあることは珍しくありません。しかし、それぞれの機器のリモコンは誤動作することもなく動きます。一体なぜなのでしょう？

図Aは、さまざまな機器で使われる赤外線リモコンで「電源ON」のボタンを押したときの信号です。同じ「電源ON」のボタンを押しても、それぞれのリモコンが別々の信号を出力するようすがわかります。

赤外線リモコンの受信部は、40kHzなどの点滅だけでなく、送られてくるデータフォーマットを認識し、自分のリモコン装置の信号だけを分離して、機器を制御するしくみになっているのです。



図A　いろいろなメーカーの赤外線リモコンの信号の内容
メーカーによって信号の内容はまちまちであることがわかる。

ますし、そのすべてのデータ内容が公開されているわけではありません。データフォーマットを自分で調べるのは簡単な作業とはいえません。

そこで、テープレコーダのように、録音（受信）した波形を、そっくりそのまま再生（送信）して、テレビなどを操作するという戦略を立てました。こうすれば、リモコンから発信される信号をプログラムで解析する必要はなくなります。

さて、仕掛けを考えましょう。テレビなどの赤外線リモコン受信用に使われている「赤外線リモコン受信モジュール（**写真3**）」をL-Card＋に取り付ければ、リモコンが送信するデータを取り込むことができそうです。赤外線リモコン受信モジュールは、約40kHzの変調をカットして、データ信号のみを出力するようになっています。今回のように生のデータを記録して、それを赤外線LEDで送信する場合には、データに40kHzの変調を加えた状態でLEDをON／OFFしなくてはなりません。

それでは、L-Card＋を赤外線リモコンにできるか検証してみます。外付けするハードウェアや、製作するプログラムなどはできるだけ簡単に作れることを重視して考えていこうと思います。

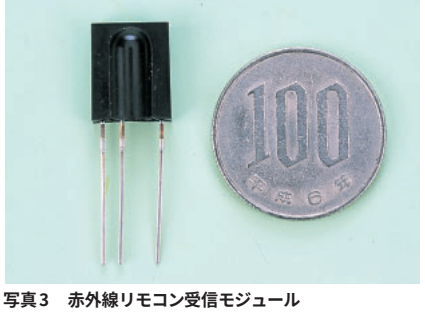


写真3　赤外線リモコン受信モジュール
リモコンの赤外線を受信する部品。

■赤外線LEDのON／OFFのためには

送信には赤外線LEDを使います。L-Card＋に搭載されたLEDでは赤外線が出ませんので、汎用I/Oポートに赤外線LEDを接続して制御することになります。

LEDのON／OFFは、リモコン制御信号のほかに、約40kHzで変調を掛けなくてはなりません。VR4181内蔵の機能で、これを行うのは難しいので、ソフトウェアで行うか、点滅用ハードウェアを外付けするかのどちらかになります。しかし、Linuxはリアルタイムではないので、これらのLEDの点滅を決められた時間でできるのか、どこまでできるのかを実験してみることにしました。

■リアルタイムOS

Linuxはマルチタスクシステムになっているのですが、同時といっても、1つのCPUが複数のプログラムを順番に処理してゆくので、別のプログラムを実行中は、そのプログラムは停止していることになります（**図2**）。

また、定期的にシステム管理のプログラムが、ユーザープログラム実行中に割り込んでくることになります。さらには、複数のタスクが動いていると、別のタスクが動いている間は、自分のプログラムの連続性が失われることになります。そこで、L-Card＋で赤外線リモコンを実現するために、どの程度システム管理にCPUパワーを使っているのか調べてみることにしました。

Linuxはリアルタイムではないとよくいわれます。リアルタイムOSの定義はいろいろあるようですが、簡単にいえば、決まった時間（スケジュール）ごとに、確実に特定の処理ができるかどうかということのようです。たとえ

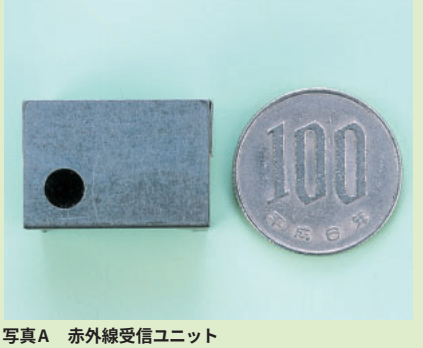
Column

赤外線リモコン受信モジュールってなに？

赤外線リモコンの赤外線ON／OFF信号を、デジタル信号に変換してくれるのが、赤外線リモコン受信モジュールという装置です。リモコン式のテレビやビデオの中には必ず内蔵されています。

昔のテレビには、**写真B**のような少し大きめの赤外線受信ユニットが使われていました。金属ケースの中には、いろいろな部品が入っています。最近のテレビには**写真3**のような小型で一体化されたものが使われています。外見は違いますが、どちらも同じように使えます。

赤外線リモコン受信モジュールは、リモコンが発振した約40kHzの変調がかかった信号しか受け取らないので、太陽光などの赤外線は排除してくれます。この変調周波数にはいろいろな種類があって、今回使用したモジュールは30kHz～56kHzまで7種類が用意されていました。試しに買ってみたのは38kHzと40kHzのものですが、38kHzのリモコンの赤外線を受信させると、どちらも同じように反応します。つまり、周波数が38kHzでも、40kHzでも、どちらでも使えるということです。厳密には、受信と送信の距離が離れて赤外線の光量が減ってくると、到達距離に違いがでてくるのですが、1m未満の至近距離での実験ではまったく関係なく動作しました。



写真A　赤外線受信ユニット
昔のテレビに使われていたユニットは、シールドケースに入っていて、今のものに比べると少し大きい。

ば、40kHzでLEDを点滅させるなら、約25μ秒ごとにLEDをON／OFFさせる必要があるわけですが、通常のLinuxにはこれを指定する方法がありません。

そこで、実践的テスト法として、高速にI/Oポートをアクセスするプログラムを走らせ、その出力をオシロスコープで確認し、ポートアクセスがない時間（CPUがシステム処理をしている）を見てみました。

結論としては、バックタスクで一切のプログラムが動いていない状態、LANやシリアルの処理も行われない状態で、約10m秒ごとに100μ秒程度のシステム処理が行われるだけで、あとはすべてフォアグランドの処理をしていることがわかりました（**図3**）。ただ、telnetでL-Card＋をアクセスしたり、シリアルポートからコマンドをたたいたりすると、数10m秒単位でほかのタスクの処理が実行されるようになり、フォアグランドの連続性が失われてしまいます。

L-Card＋をリモコンにするときは、裏でデーモンを動かしたり、制御用以外のシェルを動かしたりすることはなさそうですので、デバイスドライバ形式をとらず、普通のプログラム形式でも問題なく動くであろうという想定で製作を始めました。逆にいうと、バックグランドで何かのデーモンを動かしたりしたい場合は、このプログラムは使えません。

■変調用には外付け回路を使う

先の検証結果から、10m秒に一度、100μ秒程度プログラムの実行が止まることになります。約40kHz（25μ秒ごと）でLEDを点滅中にこれが発生すると、その周波数は大きく狂うことになります。

10m秒に一度だけですので、なんとかなるかもしれませんが、ちょっと気持ち悪いので変調にはハードウェアを使うことにしました。ハードウェアに任せることで、プログラムの負担が軽くなるし、プログラミングも楽になるというメリットもあります。

次に、リモコンのデータ信号への影響を考えてみましょう。データは600μ秒で1符号なので、16符号に一度の割で、影響を受けることになります。符号も、600μ秒が700μ秒になりますが、その差は15%程度で収まります。制御信号の受信／再生に関しては、なんとなるかなといった感じです。

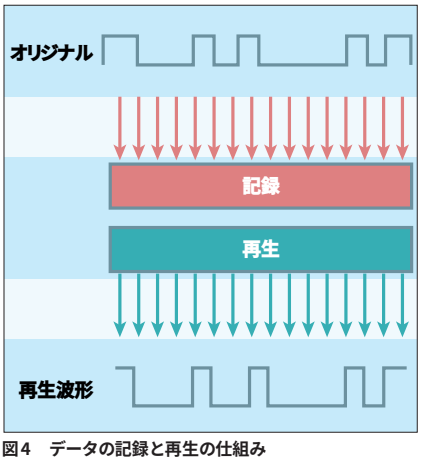


図4　データの記録と再生の仕組み
一定間隔でサンプリングしたデータを記録し、同じ間隔で再生するとオリジナルに近い波形が復元できる。

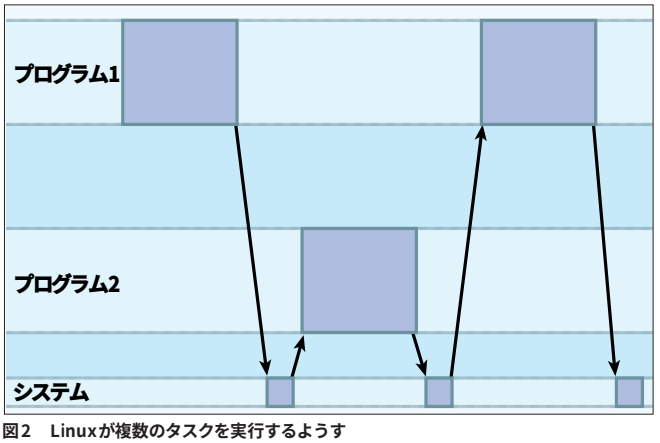


図2　Linuxが複数のタスクを実行するようす
1つのタスクは連続して実行されない。

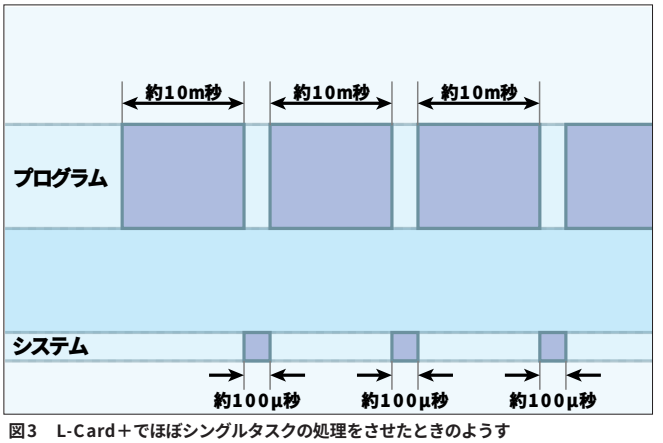


図3　L-Card＋でほぼシングルタスクの処理をさせたときのようす
10m秒に1度、100μ秒程度システムの割り込みがある。

Column

学習型リモコン

赤外線リモコンのデータを取り込んで、それをそのまま使って、他社のリモコンになりきる「学習型リモコン」なるものが市販されています。

これを使えば、テレビでも、エアコンでも、複数の装置を1個のリモコンで操作することが可能になります。

ビデオ用のリモコンには、他社製のテレビを操作できるものもありますが、これは、各社のテレビで使用しているリモコンのデータフォーマットをあらかじめ内蔵していて、それを切り替えることで、テレビのリモコンになりきるタイプのものです。学習しているわけではありませんが、他社のリモコンになりきるところはいっしょですね。

■サンプリングについて考える

リモコンからデータ信号を取り出すために、赤外線リモコン受信モジュールを使います。これは、変調信号をカットして、制御信号だけ出力してくれるので、その信号を、VR4181の汎用I/Oポートを使って読みとり、そのまま記録することにします。

信号の記録は、**図4**のように、一定間隔ごとに信号レベルを調べ、それを保存します。サンプリングの間隔は、1つの信号の10分の1くらいあれば十分実用になるようです。赤外線リモコンの信号は1符号で600μ秒程度なので、60μ秒間隔で記録します。

リモコンのボタンを押したとき、どのくらいの長さの信号が出るのかは、機種によって違うのでなんともいえませんが、今回は1万ポイントとしてみました。これは、60μ秒×1万＝60万μ秒なので、0.6秒になります。

このようにサンプリングしたデータを再生するには、サンプリングしたときと逆に、60μ秒ごとにデータを1つずつ出力させます。サンプリング間隔の間に信号変化があると、それを再現することはできませんが、サンプリング間隔を高速にすれば、出力される波形は、よりオリジナルに近づくわけです。

今回は、この60μ秒という周期を、プログラムによるループで作っています。この周期は、かならずしも正確である必要はないので、カット＆トライで適当に決めてしまっても大丈夫です。たとえば、40μ秒でサンプリングしてしまったとしても、40μ秒で再生すれば、出力される波形に違いは出ません。この場合、むしろ高速な信号変化にも対応できて、よりオリジナルの波形に近づくことになります。ただし、全体の記録時間が0.4秒と少し短くなってしまいます。

リモコンを作ってみよう（ハードウェア編）

回路図は**図5**のようになります。赤外線LEDを使った送信部分と、赤外線リモコン受信モジュールを使った受信部分に分かれます。電源は、＋5VのL-Card＋用ACアダプタを使うことを前提にしています。＋5V以外の電源を使

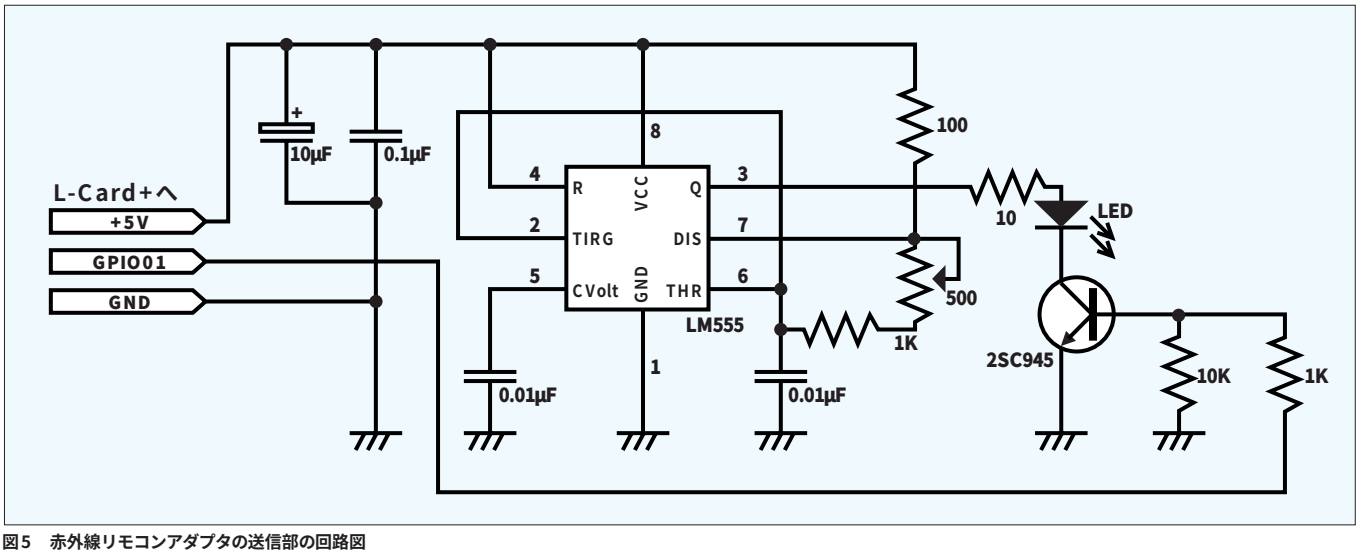


図5　赤外線リモコンアダプタの送信部の回路図

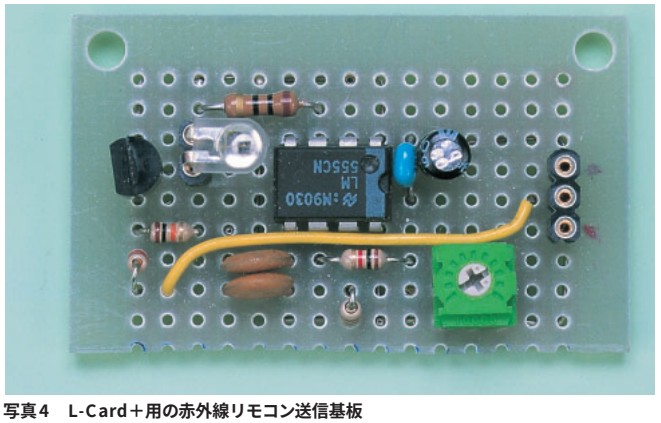


写真4　L-Card＋用の赤外線リモコン送信基板

写真5　LM555
40kHzを発振させるために使ったタイマIC。

用する場合は、回路を見直す必要があります。

■送信のためのハードウェア

送信部分（**写真4**）は、赤外線LEDをVR4181の汎用I/Oポートで制御しますが、40kHzの変調をかけるために、LM555というタイマIC（**写真5**）を使って40kHzを発振させます。タイマICはずっと発振したままなので、トランジスタを使ってON／OFFさせるという回路にしてみました。

タイマICは、コンデンサと抵抗を接続するだけで、発振してくれるICです。コンデンサや抵抗の値で発信周波数を決めることができます。ただ、ICの個体差などの影響で、多少発振周波数がずれてしまうので、これを補正するために、抵抗値を変えることができる可変抵抗器を使って、発振周波数を調整できるようにしています。

LM555の3番ピンから40kHzの出力が出てきます。ここの電圧が0Vと5Vの交互に変化するので、LEDをつなげば40kHzで点滅させることができます。

点滅させるだけでは、信号が送れないので、LEDのON／OFFはトランジスタを使います。L-Card＋のI/Oポートの電圧をHI（3.3V）にすると、トランジスタのE（エミッタ）とC（コレクタ）間の抵抗が低くなり、LM555の3番ピンが5VになったときにLEDが点灯するしくみです（**図6**）。

LEDと直列につながっている抵抗は、LEDに流れる電流を制限するものです。LEDを連続的に点灯させる場合は10mA程度の電流を流して使うのが普通です。しかし、リモコンの場合は、断続的にしか電流が流れないので発熱が少なく、連続点灯させるときより多めの電流が流せます。電流を多く流すことで、より明るく発光させることができ、リモコンを利用できる距離が長くなります。ここでは10Ωと低い抵抗で制限しています。

■受信のためのハードウェア

受信部分（**写真6**）は、赤外線リモコン受信モジュールを使うので簡単な回路（**図7**）になっています。ただ、受信モジュールは大変高感度な設計になっているので、電源ラインにノイズが入ると誤動作してしまいます。L-Card＋の5V電源はノイズが多いので、これを軽減させるために、電源ラインに簡単なフィルタを入れてあります。

■電源電圧の違いに注意

また、大きな問題になるのは信号の電圧です。受信モジュールの電源電圧が5Vなので、出力電圧も5Vです。しかし、L-Card＋の入力電圧は3.3Vなので、5Vから3.3Vに変換しなくてはなりません。受信モジュールに付属する説明書には、モジュールの内部回路が書かれていて、これを見ると100kΩの抵抗とスイッチング用のトランジスタによる出力回路になっているようです。そこで、OUT（出力）とGNDの間に200Ωの抵抗を入れてやることで、出力電圧を3.3Vに下げることにしました

Column

I/Oポートってなに

I/Oポートの「I/O」とは、Input／Outputという意味です。CPUが外部の装置からデータの入力／出力をするときに仲介をしてくれるのがI/Oポートです。広い意味では、LANのコントローラやシリアルコントローラもI/Oポートの仲間ですが、よく使われるI/Oポートが意味するものは、デジタル（2値）で、パラレル（たとえば8ビットとか）にデータの入出力を行えるポートです。

L-Card＋に搭載されているマイクロプロセッサ（VR4181）に内蔵された、汎用I/Oポートは、全部で32ビット分あり、最大32個のON／OFFの信号を入出力することができます。

赤外線リモコンでは、この汎用入出力I/Oポートを使って、赤外線LEDや、赤外線リモコン受信モジュールを制御します。

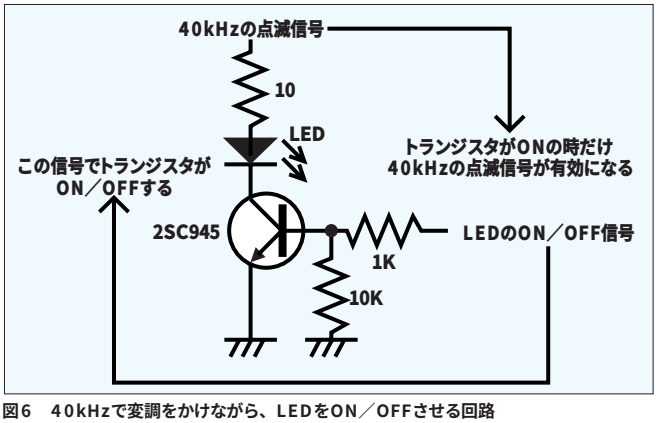


図6　40kHzで変調をかけながら、LEDをON／OFFさせる回路

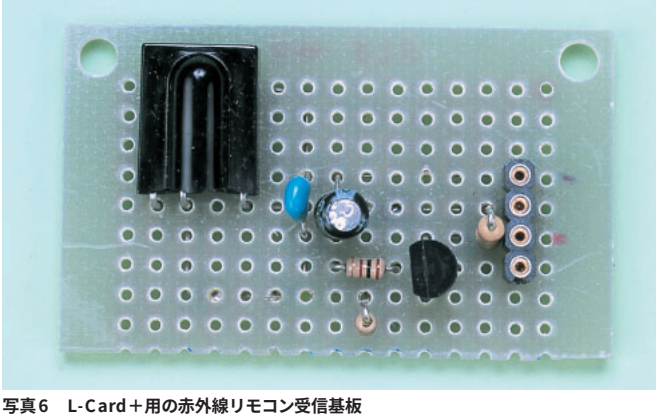


写真6　L-Card＋用の赤外線リモコン受信基板

（**図8**）。ただし、簡単な抵抗分圧による回路なので、出力電圧が3.3Vを超えてしまう可能性がないとはいえません。念のために、I/Oポートとの間にも抵抗を挟むことにしました。

■信号出力の論理の違い

赤外線リモコン受信モジュールの出力は、赤外線を検知したときにLOWレベル（0）に、何も検知できないときにHIレベル（1）になります。送信回路とは論理が反対になっているので、プログラムを作る際にも、これを考慮しないとなりません。

リモコン中継器を作って実験してみよう

L-Card＋用プログラムを作る前に、試作した送受信用ハードウェアのテストをしてみることにしました。L-Card＋を通さず、受信用回路と送信用回路を直結します。これで、リモコンからの赤外線信号を受信して電気信号に変換します。受信回路と送信回路を直結すると、その電気信号を元に送信回路から赤外線信号として出力され、TVなどを操作できるハズです。

ただし、送信回路は信号がHI（1）のときにLEDが点灯するのに対し、受信回路はLEDの信号を検出したときに出力信号がLOW（0）になります。直結するには、この信号を反転させる必要があります。そこで、**図9**のような回路を付け加え、送信回路につなぎます。

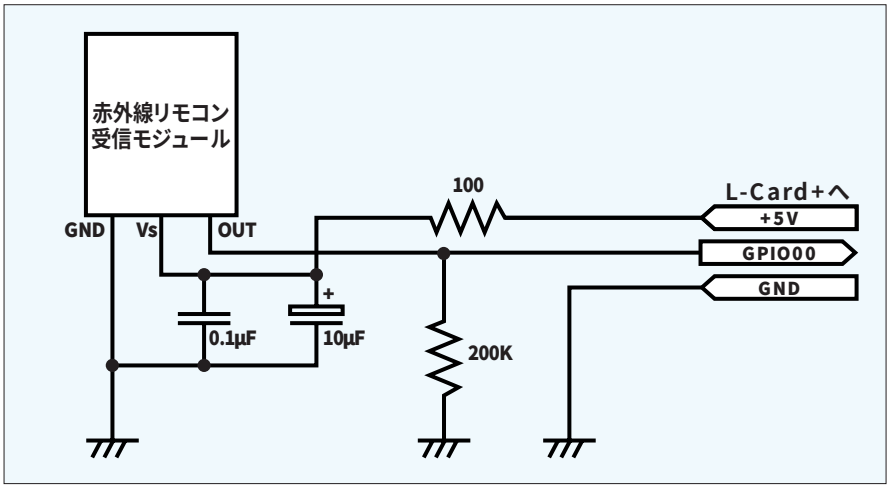


図7　L-Card＋用赤外線リモコンアダプタの受信部の回路図

■変調周波数の調整方法

送信回路の変調周波数を、使用するテレビやビデオなどの周波数（約40kHz）に合わせて調整しなくてはなりません。周波数カウンタを持っているなら、この調整も簡単なのですが、持っていない場合は、コントロールしたいテレビとリモコンを持ってきて、実際に操作しながら調整します。

まず、送信回路と受信回路を少し長めの電線でつなぎ、送信回路はテレビ

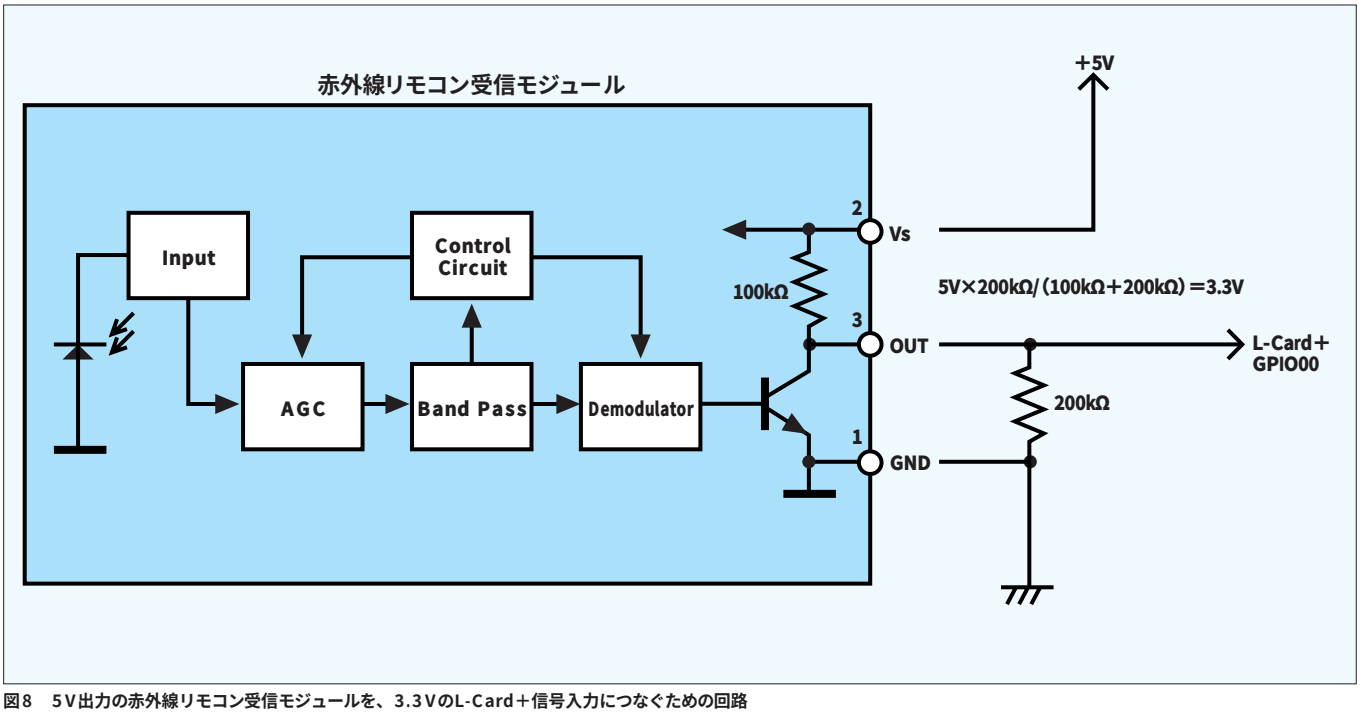


図8　5V出力の赤外線リモコン受信モジュールを、3.3VのL-Card＋信号入力につなぐための回路

の前に、受信回路はテレビから少し離れた、赤外線が届かない場所に置きます。変調周波数が適正なら、赤外線リモコンを受信部に向けて操作したときに、テレビも反応するはずです。

送信部とテレビの距離が近ければ、多少周波数がずれていても反応するので調整しやすいかもしれません。赤外線を遠くまで飛ばしたいなら、少し離れたところで再調整するとよいでしょう。

■リモコン中継装置で遊ぶ

送信回路と受信回路の間は電気信号になるので、長めのケーブルを使えば見通しできない隣の部屋にも設置できます。ですから、離れた部屋のテレビを、いながらにして手元のリモコンで操作することもできるようになります。

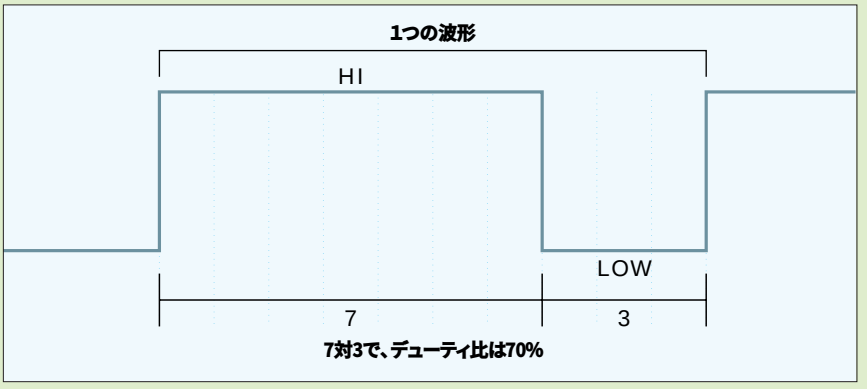
もうひとつの特徴は、テレビとビデオの赤外線リモコンの変調周波数が近い場合、1つの中継装置で、TVとビデオの両方を、それぞれのリモコンでコントロールできるということです。ただし、送信部の赤外線LEDが向いている方向にある装置しか制御できないことになりますが、複数の装置で使えるのは便利です。

次回、後編では、送信部と受信部の基板の製作、プログラムについて、使用結果などをご紹介しましょう。

Column

タイマIC、LM555の設定条件

LM555のデータシートには、発振用コンデンサと抵抗の発振周波数の計算式が載っています。式の中には抵抗が2種類（Ra／Rb）あって、Rb／(Ra＋2Rb)という式にしたがってデューティ比（図B）を可変できるのですが、赤外線LEDの点滅のデューティ比を50%近くにしておきたいので、Raの値は小さく（100Ω）、Rbの値は大きくなるように（1.5kΩ）抵抗値を選んでいます。



図B　デューティ比とは、信号がHIのときと、LOWのときの時間の比率

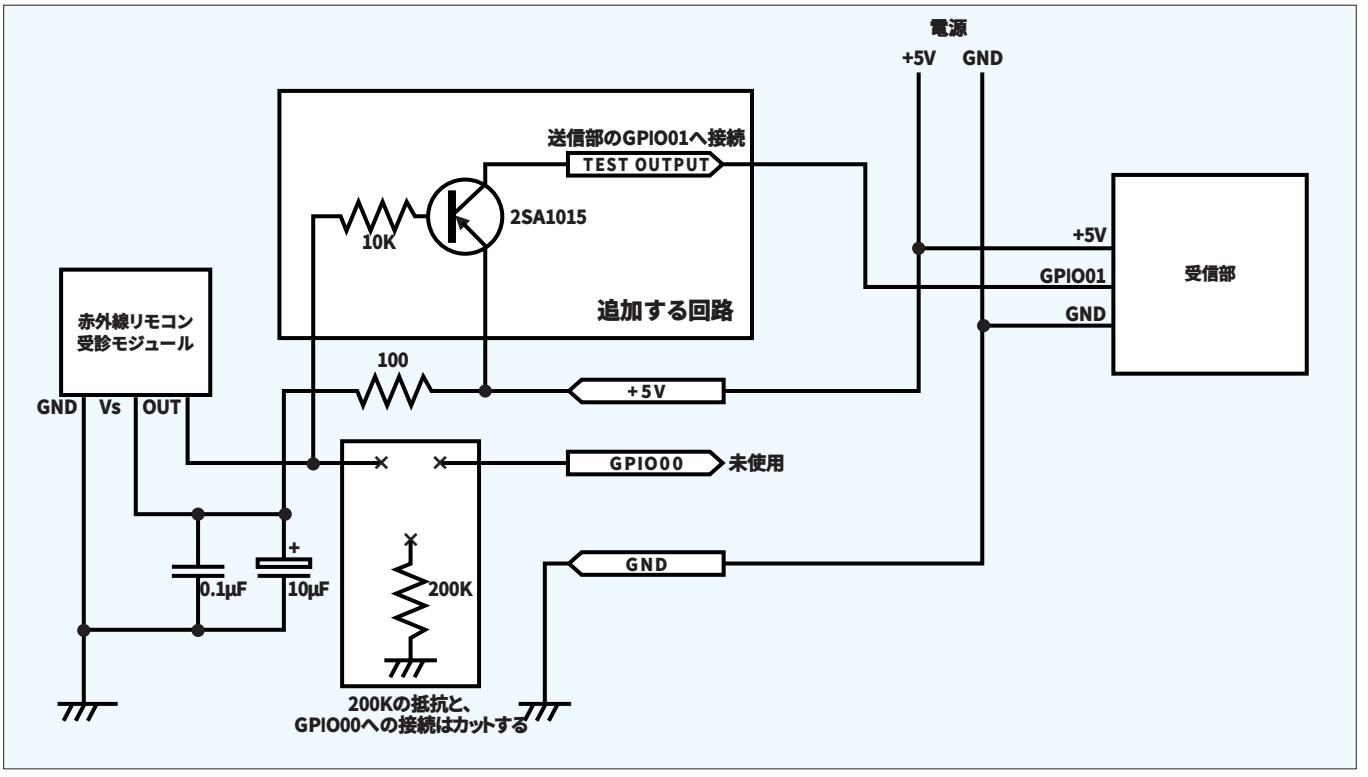


図9　送信部と受信部をつないで、赤外線リモコン中継器を作るための回路

携帯電話 de Javaプログラミング

今回は、iアプリが提供している描画機能について説明し、キャンバス上に図形や文字がどのように表示されるかを解説します。

第3回 iアプリの作成その2

文：加藤大受
Text : Daiju Kato　dkato@jcom.home.ne.jp

前回説明したように、Windows環境でiアプリを作成するのであれば、NTTドコモのWebサイトからJ2ME Wireless SDK for DoJaをダウンロードして使用すればいいのですが、Linux環境で行う場合は次の2点に注意しなければなりません。

- **J2ME CLDCが必要**
- **日本語はすべてシフトJISを使用し、コンパイルのエンコーディングをシフトJISにする**

J2ME（Java2 Micro Edition）には、CDCとCLDCというコンフィグレーションがあります。これについても前回解説しましたが、CDCはセットトップボックスやPDAなど、CLDCはCDCよりも小さいメモリ環境で使用される携帯電話などをサポートします。なお、Linuxでiアプリを作成するには、J2ME CLDCが必要になります。

J2ME CLDCのセットアップ

J2ME CLDCはサン・マイクロシステムズのWebサイトからダウンロードすることができます（**表1**。ユーザー登録が必要）。

サン・マイクロシステムズのWebサイトでは、J2ME CLDC 1.03 Reference Implementationと呼ばれる、CLDCの仕様をインプリメントしたソースコード一式（j2me_cldc-1_0_3-fcs-src-b17-winunix-14_sep_2001.zip）をダウンロードすることができます。ダウンロードしたソースコードはコンパイルする必要があります。ソースコードのコンパイルは**画面1**のように行います。ここでは/tmpディレクトリで作業を行っています。なお、コンパイルにはJDKのツールがあるディレクトリにパスが設定されている必要があります。

```
# unzip j2me_cldc-1_0_3-fcs-src-b17-winunix-14_sep_2001.zip   ZIPファイルの展開
# cd j2me_cldc/build/linux/                                  Makefileのあるディレクトリに移動
# export PATH=/usr/java/jdk1.3.1/bin:$PATH                   JDKのツールにパスを設定
# make                                                       makeの実行
# mv /tmp/j2me_cldc /usr/local                               J2ME CLDCを/usr/localに移動
```

画面1　J2ME CLDC 1.03 Reference Implementationのコンパイル

事前承認について

iアプリ用に作成したclassファイルは、正式にJ2ME CLDCの仕様に従っているかどうか事前承認する必要があります。前回は、コンパイルしたiアプリのclassファイルの事前承認は行いませんでしたが、本来はJ2ME CLDCに含まれているpreverifyというツールを使用し、作成されたclassファイルが仕様に従っているかどうかの確認作業を行います。このツールはj2me_cldc/

bin/linuxディレクトリ内に納められており、このツールで事前承認を行ったあと、jarコマンドを使用して配布用のjarファイルを作成します。

日本語エンコーディング

前述したように、日本語文字列はすべてシフトJISを使用する必要があります。JBuilderのLinux版でシフトJISを使用するには、プロジェクト作成後、［ツール］－［プロジェクトプロパティ］で表示される「プロジェクトプロパティ」ダイアログボックスの［一般］ページで、［エンコーディング］をシフトJISに設定します（**画面2**）。これで、このプロジェクト内のソースコードが使用する2バイト文字はすべてシフトJISになります。

図形・画像の描画

前回は、グラフィックスを描画する領域であるキャンバスに文字を描画しました。キャンバスはCanvasクラスによって作成されます。このクラスは画面全体をひとつのビットマップとして扱うことができ、このビットマップの上に文字、図形、画像を描画することができるようになっています。前回、文字を表示するiアプリを作成しましたが、文字データとして表示されているのではなく、文字が図形としてこのビットマップの上に描かれているのです。

Canvasクラスには**表2**のようなメソッドが用意されています。

iアプリでは各機種によって動作が変わることを説明しましたが、Canvasクラスでもこのメソッドを呼び出した場合、新規に新しいグラフィックスオブジェクトを作成するかどうかは機種に依存します。また、repaint()

画面2　エンコーディングの設定

http://www.java.sun.com/products/cldc/

表1　J2ME CLDCのダウンロードサイト

パッケージ	com.nttdocomo.ui	
親クラス	com.nttdocomo.ui.Frame	
メソッドの種類	メソッド	機能
親クラスから継承されているメソッド	public int getHeight()	キャンバスの高さを取得
	public int getWidth()	キャンバスの幅を取得
	public int setBackground(int c)	キャンバスの背景色を設定
	public int setSoftLabel(int key, java.lang.String label)	ソフトキーのラベルの文字列を指定
Canvasクラス独自のメソッド	public Graphics getGraphics()	キャンバスに描画するためのグラフィックスオブジェクトを取得
	public int getKeypadState()	キーパッド状態を返す
	public abstract paint(Graphics g)	キャンバスに表示
	public void processEvent(int type, int param)	低レベルイベントが通知されたときに呼ばれる
	public void repaint()	キャンバス全体を再描画
	public void repaint(int x, int y, int width, int height)	キャンバスの一部の矩形領域を再描画

表2　Canvasクラス

Column

Javaのクラスとメソッド

表2のように、Javaのクラスの解説では必ず親クラスがどれであるのか、また親クラスから継承されたメソッドとそのクラスで定義されているメソッドを分けて表記します。

メソッドは他のクラスから呼ばれることになりますので、必ずメソッドを公開していることを示すpublicというキーワードが宣言されています。公開しない場合はprivateというキーワードを使用しますが、privateを使用した場合、同一クラス内からしか呼び出せないという制限があります。

また、メソッドの説明では必ずメソッドの返り値が明記されています。getHeight()メソッドの場合だと返り値がintになっていますので、数値を返すことを意味しています。

paintメソッドにはabstractというキーワードが付いていますが、abstractは抽象であるという宣言です。抽象というと非常にわかりづらいですが、abstractのキーワードが付いているメソッドは中身がサブクラスで定義されていることを意味しています。つまり、Canvasクラスを継承したクラスで定義しなければならないということになります。このため、前回のリスト2ではmyCanvasクラス内にpaint()メソッドを定義して、paint()メソッド内でキャンバスへの書き込みを行うコードを実装しています。

メソッドを呼び出して再描画する際、アプリケーション部だけを再描画するか、あるいはタイトル部とアプリケーション部全体を再描画するのかは機種に依存します（**画面3**）。

getKeypadState()メソッドは、どのキーが押されているかを調べることができます。キーがiアプリ内でどのように管理しているかについてはまた別の機会に説明します。

キーを押すとイベントが発生しますが、このイベントを調べるために使用するのがprocessEvent()メソッドです。processEvent()メソッドでは、低レベルのイベントを監視することができます。このメソッドの使い方についてももう少し高度なサンプルを作成する段階で説明します。

前述したように、Canvasクラスを使用してキャンバスオブジェクトを生成することによって、画面全体をひとつのビットマップとして扱うことができます。実際に、このビットマップに描画を行うのがGraphicsクラスとなります。描画指示はGraphicsクラスのメソッドを使用して行います。

Graphicsクラスには、図形の描画用に5種類のメソッドが用意されています（**表3**）。

それでは、前回作成したmyCanvas.

画面3　キャンバスのタイトル部とアプリケーション部

機能	メソッド名	備考
線の描画	drawLine(int x1, int y1, int x2, int y2)	
矩形の描画	drawRect(int x1, int y1, int width, int height)	
線分の描画	drawPolyline(int[] xPoints, int[] xPoints, int nPoints)	nPointsは頂点の数
矩形の塗りつぶし	fillRect(int x, int y, int width, int height)	
多角形の塗りつぶし	fillPlygon(int[] xPoints, int[] yPoints, nPoint)	nPointsは頂点の数

表3　Graphicsクラスのメソッド

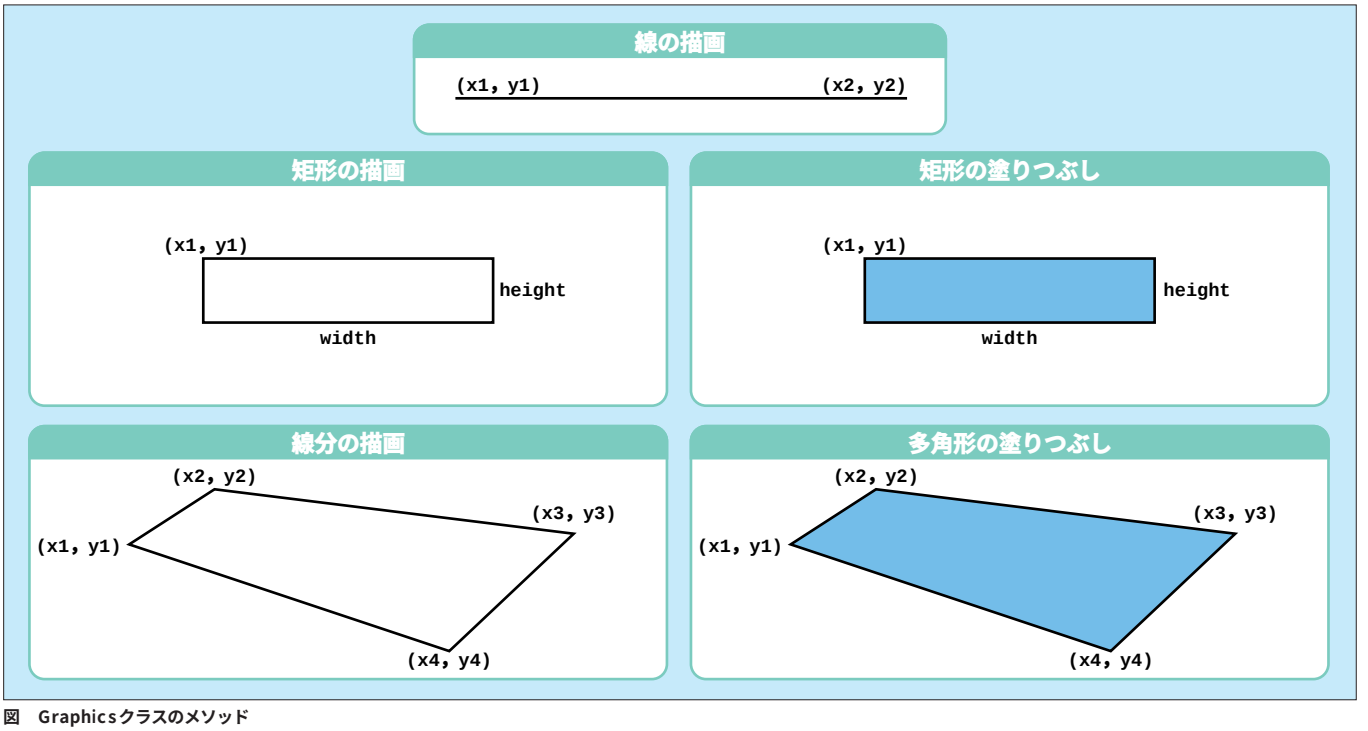


図　Graphicsクラスのメソッド

javaに図形の描画のコードを追加し、どのように描画されるか確認してみましょう。**リスト1**のiApplidemo.javaは、前回作成したままで修正する必要はありません。前回の**リスト2**（myCanvas.java）の文字列の描画のあとに、先ほど説明したメソッドを利用して直線、矩形、矩形の塗りつぶし、線分、多角形の塗りつぶし図形を描画しています。線分を描画するdrawPolyline()メソッドの部分では、(x1，y1)と(x4，y4)の座標を同じにして、三角形を描画しています（**リスト2**の①）。このソースコードの実行結果は**画面4**のようになります。

画面4ではすべての線が黒で、また塗りつぶしも黒で行われていますが、iアプリでは描画する色を指定して図形を描画したり、塗りつぶしたりすることができます。色の指定には、GraphicsクラスのsetColor()メソッドを使用します。setColor()メソッドの引数には、使用する色の数値を指定します。この数値は、GraphicsクラスのgetColorOfRGB()メソッドを使用してRGBから取得するか、getColorOfName()メソッドを使用してGraphicsクラスで定義している色名から取得する必要があります。それぞれのメソッドは次のような引数が必要となります。

①getColorOfRGB()メソッド

```
getColorOfRGB(int r, int g, int b);
```

②getColorOfName()メソッド

```
getColorOfName(int name);
```

getColorOfNameで指定できる色名は**表4**のものです。実際に、数値を取得する例が**リスト3・4**です。

ただし、使用できる色数は機種によって異なります。各機種ごとの色数は**表5**のようになります。

それでは、先ほど修正したmyCanvas.javaに色の指定を行い、描画する図形の線の色や塗りつぶしの色を指定して

画面4　図形の描画

▼リスト1　iAppliDemo.java

```
import com.nttdocomo.ui.*;

public class iAppliDemo extends IApplication {

  //キャンバスの定義
  myCanvas canvas;
  public void start() {

  //キャンバスの作成
  canvas = new myCanvas();
  //キャンバスの表示
  Display.setCurrent(canvas);
  }
}
```

▼リスト2　myCanvas.java

```
import com.nttdocomo.ui.*;

public class myCanvas extends Canvas {

 int width;
 int hight;

  public myCanvas() {
    //画面サイズの取得
    //携帯電話のディスプレイの領域を取得
    width= Display.getWidth();
    hight= Display.getHeight();
  }

  //キャンバスへ書き込み
  public void paint(Graphics g) {
    //画面の初期化
    g.clearRect(0,0,width,hight);
    //文字列の描画
    g.drawString("iアプリデモ",5,20);
    //直線
    g.drawLine(1,25,100,25);
    //矩形を描く
    g.drawRect(10,30,20,20);
    //矩形の塗りつぶし
    g.fillRect(50,30,20,20);
    //線分を描く
    int xDrawValue[] = {10,80,110,10};  ①
    int yDrawValue[] = {60,65,105,60};
    g.drawPolyline(xDrawValue,yDrawValue,4);
    //多角形の塗りつぶし
    int xFillValue[] = {10,85,105,75};
    int yFillValue[] = {90,100,125,120};
    g.fillPolygon(xFillValue,yFillValue,4);
  }
}
```

みましょう。**リスト5**は色の指定を行った、myCanvas.javaのpaint()メソッド部分のソースコードです。文字や図形の描画の前で、setColor()メソッドを呼び出して色の指定を行っていることがわかると思います（**画面5**）。

線形の描画と、多角形の塗りつぶし部分では、16進数でRGBの値を設定しています。Java言語では16進数の値を指定するときは前に「0x」を付けます。16進数をわざわざ10進数に変更する必要はありません。Webページの作成時に指定している、FONTタグのCOLORパラメータの値をそのまま指定することができます。

画像の表示

キャンバス上には、文字や図形だけでなく画像を表示することもできます。画像を表示する場合は、MediaManagerクラスのgetImage()メソッドを使用します。MediaManagerクラスではイメージの登録、イメージのオブジェクトへのロードを行います。

Java言語では標準でJPEG形式とGIF形式をサポートしていますが、iモードで使用できる画像フォーマットは、GIFファイルとなります。GIFファイルは最大256色まで使用できますが、画像ファイルはjarファイル内に含めなければなりませんのでサイズに注意しなければなりません。画像で使用する色数をできる限り減らして、容量を削減するなどの工夫が必要となります。

色名	色	色名	色
AQUA	シアン	NAVY	ネイビー
BALCK	黒	OLIVE	オリーブ
BLUE	青	PURPLE	紫
FUCHSIA	ピンク	RED	赤
GRAY	灰色	SILVER	銀色
GREEN	緑	TEAL	深い水色
LIME	ライム	WHITE	白
MAROON	茶色	YELLOW	黄色

表4 getColorOfNameで指定できる色名

▼リスト5　色の指定

```
//キャンバスへ書き込み
public void paint(Graphics g) {
  //画面の初期化
  g.clearRect(0,0,width,hight);
  //文字列の描画
  g.setColor(g.getColorOfName(g.BLUE));     // 青色を指定
  g.drawString("iアプリデモ",5,20);
  //直線
  g.setColor(g.getColorOfName(g.GREEN));    // 緑色を指定
  g.drawLine(1,25,100,25);
  //矩形を描く
  g.setColor(g.getColorOfName(g.SILVER));   // 銀色を指定
  g.drawRect(10,30,20,20);
  //矩形の塗りつぶし
  g.setColor(g.getColorOfName(g.YELLOW));   // 黄色を指定
  g.fillRect(50,30,20,20);
  //線形を描く
  int xDrawValue[] = {10,80,110,10};
  int yDrawValue[] = {60,65,105,60};
  g.setColor(g.getColorOfRGB(0xFF,00,0xCC)); // ピンク系を指定
  g.drawPolyline(xDrawValue,yDrawValue,4);
  //多角形の塗りつぶし
  int xFillValue[] = {10,85,105,75};
  int yFillValue[] = {90,100,125,120};
  g.setColor(g.getColorOfRGB(0xCC,00,00));    // 赤系を指定
  g.fillPolygon(xFillValue,yFillValue,4);
}
```

機種名	色数
P503i	256色
F503i	256色
N503i	4096色
SO503i	65536色
D503i	4096色
P503iS	256色
N503iS	4096色
F503iS	4096色
D503iS	4096色
SO503iS	65536色

表5　各機種で使用できる色数

画面5　色を指定したところ

▼リスト3　RGB値を利用して色の数値を取得（緑色を指定）

```
getColorOfRGB(0,255,0);
```

▼リスト4　色名を指定して色の数値を取得（赤色を指定）

```
getColorOfName(g.RED);
```

なお、読み込まれた画像はフォーマットに依存しない内部形式となっています。このため、画像を描画する手順は複雑です。まず、画像ファイルの情報（リソース情報と呼ぶ場合もある）を取得したあと、use()メソッドでファイルを使用できるように準備します。この状態ではまだオブジェクトに格納されてもいません。単に、準備作業が行われるだけです。続いて、イメージオブジェクトという、Java言語内で使用できる形式で格納します。その後、格納された画像データをキャンバス上に描き出すことで画像を表示することができます。

リスト6は、画像を表示するimgCanvas.javaのソースコードです。それでは、**リスト6**を見ながら画像を表示する流れを見てみましょう。

まず、画像を格納するイメージオブジェクトの変数を定義します（**リスト6**の①）。この変数に画像データを格納することになります。

続いて、imgCanvasクラスのコンストラクタであるimgCanvas()メソッド内で、画像データをメディアイメージオブジェクトに格納します（**リスト6**の②）。メディアイメージオブジェクトに格納するのは、必要があったときにすぐに画像の表示を行うなどの処理を行えるようにするためです。つまり、画像ファイルのデータをメモリ上に格納し、画像表示などの準備を行っているのです。画像ファイルは、jarファイル内、またはクラスファイルが格納されているディレクトリ内のimagesディレクトリに納められています。この連載ではJBuilderで開発していますので、classesディレクトリ（jbproject/＜プロジェクト名＞/classes）内にimagesディレクトリを作成し、その下に呼び出したい画像ファイルを格納します。

次に、取得したイメージをロードし、準備作業を終わらせます（**リスト6**の③）。

さらに、メディアイメージオブジェクトに格納された画像データをイメージオブジェクトに格納します（**リスト6**の④）。これで、画像を表示する準備がすべて終了しました。いつでも格納されたデータをキャンバス上に描き出すことができます。

キャンバスへの書き込みはpaintメソッド内で行います。GraphicsクラスのdrawImage()メソッドを呼び出すことで画像を表示することができます（**リスト6**の⑤）。grawImage()メソッドでは、表示する画像の左上の座標を指定し、どこに表示するかを指定します。

このように、iアプリ内で画像を使用するときは、画像をメモリ上に呼び込み、使用できる状態に整え、再度イメージオブジェクトとして取り出してからキャンバスに描き出すことになります。少しややこしい手続きとなりますが、この流れがあることにより画像フォーマットに依存することなく、同じ手続きで画像を扱うことができるよう

▼リスト6　画像の表示(imgCanvas.java)

```
import com.nttdocomo.ui.*;

public class imgCanvas extends Canvas {

 int width;
 int hight;

 //イメージオブジェクトの定義
 Image image;                                          ①

  public imgCanvas() {
    //画面サイズの取得
    //携帯電話のディスプレイの領域を取得
    width= Display.getWidth();
    hight= Display.getHeight();
    //画像データをメディアイメージオブジェクトに格納
    MediaImage mi =
        MediaManager.getImage("resource:///images/demo.gif");   ②
    try {
      //取得したイメージをロード
      mi.use();                                        ③
    } catch (Exception e) {}

    //メディアイメージからイメージオブジェクトを取得
    image = mi.getImage();                             ④
  }

  //キャンバスへ書き込み

  public void paint(Graphics g) {
    //画面の初期化
    g.clearRect(0,0,width,hight);
    //画像の表示
    g.drawImage(image,2,2);                            ⑤
  }

}
```

```
$ cd jbproject/iAppliDemo1/classes        ← classファイルがあるディレクトリに移動
$ /usr/local/j2me_cldc/bin/linux/preveriy -classpath /usr/local/i-jade/i-jade-ps.jar:/usr/local/j2me_cldc/api/classes.zip
:. iAppliDemo1 imgCanvas                  ← preverifyツールの実行
$ cp output/*.class .                     ← 承認されたclassファイルをコピー
```

画面7　preverifyツールによる事前確認

```
$ /usr/java/jdk1.3.1/bin/jar cvfM iAppliDemo1.jar iAppliDemo1.class imgCanvas.class images
```

画面8　jarコマンドの実行

になっています。

プログラムの実行結果は**画面6**のようになります。

事前承認と配付

i-JADEでiアプリの動作を確認することができたら、J2ME CLDCに付属するpreverifyツールを利用して作成されたclassファイルが正しいかどうかの事前承認を行います。事前承認はコマンドラインで行います（**画面7**）。

preverifyツールを使用するときは、-classpathオプションでi-JADEのjarファイルと、冒頭でコンパイルしたJ2ME CLDCのAPIのクラスファイル、事前承認を行いたいclassファイルの場所を指定する必要があります。

事前承認が正常に終了すると、outputというディレクトリ内に承認されたclassファイルが作成されます。iアプリとして配布する場合は、これらのファイルをjarコマンドを利用してjarファイルにまとめます。画像を伴う場合はjarコマンドでクラスファイルと一緒に画像ファイルもjarファイルに格納します（**画面8**）。

最後に、jamファイル（iアプリでは、ADF： Application Descriptor Fileと呼ぶ）を作成します。

jamファイルには、作成されたjarファイルのサイズをAppSizeに指定します。**リスト7**～**9**は実際に作成したiアプリを配布するために必要なjamファイルとHTMLファイルです。HTMLファイルには、iアプリ以外の携帯からアクセスしたときにエラーが表示されるように、<A>タグがエラーの場合のファイルを指定しておきます。

これらのサンプルは筆者の個人のページで実際にダウンロードすることができます（**表6**）。

コンポーネントの利用

文字や図形の描画を行うAPIについて説明しましたが、今まで説明したも

画面6　画像を表示したところ

▼リスト7　iAppliDemo1.jam

```
PackageURL = iAppliDemo1.jar
AppName = iAppliDemo1
AppClass = iAppliDemo1
AppSize = 2989
SPsize = 0
LastModified = Wed, 17 Oct 2001 01:34:20
```

▼リスト9　error.html

```
<html>
503i以外の携帯ではiアプリのアクセスできません。<BR>
</html>
```

▼リスト8　iAppliDemo.html

```
<html>
<OBJECT declare id="iAppliDemo" data="iAppliDemo.jam" type="application/x-jam">
</OBJECT><A ijam="#iAppliDemo" href="error.html">iAppliDemo</A>
</html>
```

のはどちらかというと低レベルなAPIです。これとは別に、高レベルなAPIとしてテキストボックスやボタンなどを簡単に作成できる、Componentクラスというフレームワークが用意されています。iアプリの環境設定や、ビジネス系のプログラムなどを開発するときはこちらのコンポーネントを利用すると便利です。

Componentクラスは、高レベルAPIのコンポーネントを表示する抽象クラスとなっており、**表7**のようなコンポーネントが用意されています。

Componentクラスには、作成するコンポーネントの設定や情報を取得するためのメソッドが用意されています(**表8**)。

Componentクラスで用意されているコンポーネントは、Panelクラスに追加していきます。つまり、パネル上に各コンポーネントを配置していくことになります。各コンポーネントのPanelへの追加は、add()メソッドで行います。パネルのサイズは携帯電話の表示領域のサイズです。

■イベントリスナー

コンポーネントを選択したときや、押したときにイベントが発生します。Java言語はイベントドリブンな言語ですので、何らかのアクションがあるとイベントが発生します。イベントはイベントリスナーと呼ばれる、イベントを監視する機能によって、イベントに関連付けされている処理が実行されます。イベントリスナーには、5種類のインターフェイスが用意されています(**表9**)。イベントリスナーを使用する場合は、クラスの定義の際に使用するリスナーを、implementsキーワードのあとに指定しなければなりません。これは、インターフェイスの宣言といい、イベントリスナーを使用するときは、クラス定義のあとに定義を行うと覚えてください。

ComponentListenerはボタンを押したときや、リストボックスの値が変わったときなどに発生するイベントを監視します。

SoftKeyListenerは、携帯電話のソフトキーを押したときに発生するイベントを監視します。一般的に、SoftKeyListenerにはiアプリの終了などの処理を設定します。

KeyListenerは、携帯電話のキーを押したときに発生するイベントを監視します。

MediaListenerは、指定されたメディアデータ(グラフィックスデータや音楽データ)を再生したり、中断したときに発生するイベントを監視します。

TimeListenerは、タイマーが指定された時間になったときに発生するイベントを監視します。

これらのイベントリスナーが監視しているイベントが発生したときに実行されるメソッドを結びつけることで、イベント処理を実現することができるようになっています。

●

次回は、Componentクラスの使い方を詳しく解説し、Componentクラスを使用したiアプリを作成します(今月号の付録CD-ROMには、Componentクラスによるiアプリを収録しています)。

http://www.ne.jp/asahi/luna/dkato/iAppliDemo.html

表6　作成した画像を表示するiアプリのサンプルのダウンロードサイト

コンポーネント	用途
Panel	Componentクラスのコンポーネントを管理するコンポーネント
Label	テキストを表示するコンポーネント
ImageLabel	画像を表示するコンポーネント
Button	プッシュボタンを表示するコンポーネント
ListBox	リストボックス、チェックボックス、ラジオボタンを表示するコンポーネント
TextBox	テキストボックスを表示するコンポーネント
Ticker	ティッカーテープ表示(マーキー)のコンポーネント
VisualPresenter	スクリーンに表示するメディアデータの再生をするためのコンポーネント

表7　iアプリで用意されているコンポーネント

メソッド名	用途
getHeight()	コンポーネントの高さの取得
getWidth()	コンポーネントの幅を取得
getX()	コンポーネントのX座標の取得
getY()	コンポーネントのY座標の取得
setBackground()	コンポーネントの背景色の設定
setForeground()	コンポーネントの前景色の設定
setLocation()	コンポーネントの位置の設定
setSize()	コンポーネントの大きさを設定
setVisible()	コンポーネントの表示・非表示の設定

表8　Componentクラスの各メソッド

リスナー名	対応するコンポーネントやキーなど
ComponentListner	Button、Text、ListBox
SoftKeyListener	SoftKey
KeyListener	0-9、*、#
MediaListener	AudioPresenter、VisualPresenter
TimeListener	Timer

表9　イベントリスナー

箱の中のペンギンたち

文：みわよしこ
Text : Yoshiko Miwa miwachan@pp.iij4u.or.jp

第10回　オープンソースビジネスの多様な姿

筆者は最近、『Linuxはいかにしてビジネスになったか コミュニティ・アライアンス戦略（NTT出版、佐々木裕一／北山聡著、国領二郎監修）』という本を読みました。この本では、主にレッドハットとVA Linux Systemsが取り上げられ、オープンソースコミュニティの産物であるLinuxでのビジネスが、これまでのビジネスとどのように異なるかが詳細に検討されています。

レッドハットとVA Linux Systemsは、Linuxディストリビューション、あるいはLinuxを搭載したサーバなどの販売、さらにその後のサービスによって収益を得ているという点で共通しています。VA Linux Systemsはディストリビューションそのものの開発や販売は行っていませんが、同社のサーバに搭載するディストリビューションに独自の改良を施して付加価値を高めています。この2社に限らず、Linux企業の多くは、ディストリビューションに関係する何らかのロイヤリティによって収益を得ています。しかしこれだけでは、「コミュニティの産物を商品化することで利益を得ている」という批判を免れません。

そこで両社は、本格的にビジネスを展開するのに前後して、開発者コミュニティとの協調関係・補完関係も構築しています。さらに、両社の利潤が開発者に不完全ながらも還元されるように注意を払っています。また、ソフトウェア開発は楽しいことばかりではありません。「好きだからやる、イヤだったらやらない」というのが一般的なハッカーの気質であり、彼らは往々にしてバグ修正やメンテナンスなど、技術も時間も必要なのに、光が当たりにくい仕事を避けて通りがちです。両社は、その「どちらかと言えば楽しくない」部分を企業活動として行うことにより、オープンソースコミュニティの産物であるLinuxに、製品として十分な市場価値を与えてきました。前出の本ではそのようすが歴史的に、克明に解説されています。

「オープンソースビジネスは難しい」と一般には言われています。ソースを公開せず完成品を売るという方法ではなく、ソースを公開したうえで十分な収益をあげられるビジネスモデルを構築することは容易ではありません。実際、Linuxディストリビューターの多くは、若干なりともディストリビューションのロイヤリティで収益をあげています。

しかし、今回紹介するモンタビスタソフトウェア（以下、モンタビスタ）は、ディストリビューションのロイヤリティとは違うビジネスモデルで成功している企業です。

モンタビスタのビジネスモデル

よく知られているように、Linux自体はソースコードがGPLに基づいて完全に公開されています。モンタビスタは、同社の組み込み向けLinuxディストリビューション「Hard

Hat Linux」を、Linuxそのものと同様にロイヤリティフリーで、GPLに基づいて公開しています。それでは、モンタビスタはどうやって収益を得ているのでしょうか。

モンタビスタは、「Hard Hat Linux」のバイナリパッケージと年間サポート契約をセットにして「サブスクリプション」という形で有料提供しています。「Hard Hat Linux」のユーザーは、自社での組み込み製品の開発のために安心して使えるバイナリパッケージとサポートを購入するということになります。このビジネスモデルは本社のある米国だけでなく、ヨーロッパや日本の市場でも高く評価されました。その結果、モンタビスタの日本法人は設立後最初の一年間で、予定通りの売上をあげています。

日本の会社員とLinuxコミュニティの微妙な関係

一般的な日本の企業の中にいると、自分の実名を出してLinuxコミュニティにコミットするのは容易ではありません。村社会の中で、個人が「目立つ」行動を取ることは暗黙のタブーでもあります。筆者自身もかつて会社員だったころ、Linuxコミュニティでの活動によって社内での立場が危険にさらされるという経験をしています。

さらに技術的なコミットメントの場合、どこまでがコミュニティ活動なのか、どこまでが業務なのかが判然としないといったことがコミットメントを難しくしています。コミュニティの一員としての発言なのか、それとも会社の業務の一環としての発言なのか、質問を受ける場合に個人として対応するのか、それとも会社の一員としての立場で対応するのか、どちらかの立場で対応するとして、どこまでの責任を持つのか、どこまでの責任が持てるのか……。

業務に熱意を持って取り組んでいればいるほど、また会社の中でより大きな権限や強い立場を持つようになればなるほど、概して「公私の別」の切り分けは難しくなってゆくものです。たとえば、コミュニティのメーリングリストに会社のメールアドレスで投稿する場合、投稿の署名に「この発言は個人としての発言であり、会社はまったく関係がありません」とただし書きをしておいたところで、投稿を読む人が発言者とその所属する企業をまったく無関係とみなすことはありえません。筆者自身は会社員時代、「あくまで個人としての行動なので、会社ではメーリングリストを読みも投稿もしない」という原則を貫き通しました。しかし、その結果としてタイムリーに発言できる機会が減り、コミュニティの人間であると自認する人々から、「貢献が少ない」という批判を口頭で浴びたこともありました。日本で会社に所属しながらLinuxコミュニティに関わっている方々は、個人の立場と組織の立場の矛盾という問題に、多少なりとも悩んでいることが多いでしょう。

個人レベルでのLinuxコミュニティとの「協調」

モンタビスタの社員には、以上のような悩みは無縁です。Linuxの開発プロジェクトに参加し、Linuxの完成度を高めることが業務の一環となっているからです。Linuxカーネルはインテルのx86だけでなく、Power PC、MIPS、StrongARM、SH、SPARCといった多くのアーキテクチャに対応していますが、そのそれぞれのアーキテクチャのプロジェクトにモンタビスタ社員が参加し、貢献しています。もちろんカーネルだけではなく、ツールやアプリケーションのプロジェクトにも参加者を出しています。Linuxの完成度を高めることが、すなわちモンタビスタのHard Hat Linuxの完成度を高めることになるというシナジー効果があるからこそ可能な形態です。

これに対して、日本の半導体メーカーがLinuxを自社のCPUに対応させたいと考えた場合、モンタビスタのようにコミュニティ活動に自社の社員を直接送り込むということは、日本の企業社会の性格から困難です。また、自社に質問やクレームが来た場合にどう対応すべきか、どこまでの責任を負うべきかという問題も障壁となるでしょう。

そこで最近では、NECや東芝など日本の大手半導体メーカーが、自社製品向けに社内で開発したコードをモンタビスタに託し、モンタビスタの技術者がそれをLinuxに反映するという状況が発生しています。モンタビスタは日本企業、あるいは日本企業で働く技術者たちがLinuxコミュニティへ貢献するための「窓口」としての役割も果たしているということになります。

現状を考えれば、このような「窓口」が存在することによって、日本の企業がLinuxへ貢献できるということは大いに評価すべきでしょう。

それにしても、日本企業で働く数多くの優れた技術者が、個人としてコミュニティ活動をすることが難しい現状自体がなんとかならないものかと筆者は思います。

オープンでないビジネスとの「協調」

100%オープンのソリューションのみで顧客のニーズに完

全に応えることは、ときに困難なこともあります。1つの会社の中にオープン・非オープンという2つの矛盾する立場が存在することは、企業活動を困難にすることもあります。

モンタビスタでは、数多くの非オープンソースビジネスを行うパートナーと協業し、それぞれの分野の製品・ソリューションで補完し合うことで、この問題に対応しています。とりわけ日本では、非オープンソースのツールを扱う兄弟会社「イーエルティ」と密接に協業することで開発ソリューションを強化しています。

具体的には、イーエルティは国内主要ツールベンダーが提供する実績ある開発ツールや開発環境をHard Hat Linuxに対応させ、販売しています。これよって、これまで開発に国産ツールを使用してきたユーザーは、日本国内で高いシェアを占めてきた実績のあるツール、慣れ親しんだツールをそのまま使用して、Hard Hat Linuxを用いた開発ができるわけです。

Hard Hat Linuxはなぜ評価されるのか

オープンソースで成功するビジネスモデルを構築し、Linuxコミュニティと協調体制を構築するだけで成功するほどLinuxビジネスは甘くありません。製品そのものに十分な魅力がなければ、ビジネスは先細りになるだけでしょう。モンタビスタの「Hard Hat Linux」がなぜ評価されたのかをここで改めて検討してみましょう。

モンタビスタ自体は1999年3月に設立された非常に若い会社です（日本法人の設立は2000年7月）。しかし、モンタビスタを設立したのは、20年以上にわたって現在でも組み込みリアルタイムOSとして利用されている、VRTXの開発者であり、「組み込みOSの父」とも呼ばれるジム・レディ氏です。ジム・レディ氏は、1992年ごろにLinuxが「将来組み込み用途で使えるのではないか」と着目し、準備を重ね、1999年にモンタビスタを設立しました。したがって、モンタビスタには当初から「組み込み技術がよくわかっている会社」という期待が集まっていました。

また、モンタビスタはHard Hat Linuxにクロス開発環境をいち早く整備しました。組み込みで用いられるCPUは主に、MIPS、StrongARM、SHですが、開発はパワーがあるインテルのx86ホスト上で行い、コードのコンパイル後のバイナリを、ターゲットとなるCPUのマシンにイーサネットやメモリを経由して送ります。このような開発形態はクロス開発と言い、Linuxに限らず組み込み製品開発では一般的な形態ですが、モンタビスタはこの環境を、多様なアーキテクチャに対していち早く整備したことで注目を受けました。

もちろん、Hard Hat Linuxが評価されたのはオープンソースで純粋なLinuxであるという点にもあります。ほかのOSに対するLinuxのアドバンテージは、ロイヤリティフリー、オープンソース、そして日進月歩で進化し続けるているという点です。加えて、Hard Hat Linuxの場合、モンタビスタが企業としてそのクオリティを保障する形となっているため、信頼のおける組み込み用OSとして評価されました。

Hard Hat Linuxは、現在ではアーキテクチャへの対応が進み、ボードへ対応した開発環境も提供しています。組み込み製品開発の場合、最初は市販のボードを用いて開発を行い、数量にもよりますが、その後カスタムのボードを作成します。Hard Hat Linuxには、現在、組み込み製品開発に使用されることの多い70～80種類のボードに対応した開発環境が含まれているため、開発者はHard Hat Linuxを導入すればすぐにアプリケーションの開発に取りかかることができるのです。組み込みシステム開発者にとっては、まさに至れり尽くせりといったところでしょう。

さらに、前述したようにLinux開発コミュニティとの緊密な連携により、モンタビスタの社員が業務としてLinuxのクオリティの向上に貢献し、そのことがモンタビスタのLinux製品のクオリティを向上させるというプラスの連環構造があります。

Hard Hat Linuxは、以上のような魅力によって組み込み製品の開発者に広く受け入れられたと言えそうです。

Linuxを利用した開発の効率の良さ

ビジネスはスピードが命です。組み込みシステム開発も例外ではありません。多くの組み込みシステム開発者・企業が、それまで使ってきたOSや自社OSにこだわるのは、慣れた環境・蓄積されたソフトウェア資産があれば速いスピードで開発を行うことができるからです。

しかし、『Embedded Linux Journal 2001年7月号』に、これまでのその「常識」を覆すような事例が掲載されました。それは、ある企業での組み込みネットワーキングボックスとターミナルサーバの開発事例です。その企業はそれまで自社OSを用いた組み込みシステム開発を行っていた会社で、Linuxを用いた開発は今回が初めてでした。この会社はいくつかの組み込み向けLinuxディストリビューションを比較・検討した結果、使用を考えているボードをサポートしている

という理由から、モンタビスタのHard Hat Linuxを採用しました。その結果、1人のエンジニアが6カ月で開発を完了させることができました。これは、これまで使用経験のないOSを採用した開発としては異例の速さです。通常ならば1.5人／年といった人的資源を必要とするところです。

この事例の報告者は、開発効率の面からも「Linuxへの移行を躊躇すべきではない」と読者に呼びかけています。掲載誌が『Embedded Linux Journal』であることを差し引いても、Linuxが組み込みシステム開発に非常に適していること、またディストリビューターによるボードまでのサポートが開発を非常に容易にすることを示している例だと言えるでしょう。

Hard Hat Linuxを利用した製品

モンタビスタによれば、この冬にかけてHard Hat Linuxを利用した製品が日本の市場に多数出現する見通しとのことです。社名や製品名はまだ発表できないということですが、中にはブロードバンドの一般家庭への普及を見越したユニークな製品もあるそうです。

たとえば近々発表されそうな製品には、ADSLやブロードバンドの受け口に設置するホームサーバがあり、この製品は、そこにPCを接続することももちろん可能なのですが、さらにコンテンツのセーブを行う機能もあるということです。また、ハードディスクにテレビ録画を行う装置も開発されているそうです。

もちろん、ビジネス用途の製品も開発されているとのことです。プリンタやFAXをPCから利用するのはごく当たり前のことですが、それらにLinuxを組み込むことで、PCとのデータ共有をより容易にするような製品が期待されます。このほか、ごく一般的なビジネス用のインターネット端末も開発されているそうです。

来年にかけては、産業用の計測器などに「深く静かに」採用されていく動きもあるとのことです。

組み込み製品でOSの名前が公開されることはめったにありませんが、Linuxを使用した製品に関しては「Linux Inside」と誇りを持って公開してほしい……と筆者は密かに思っています。

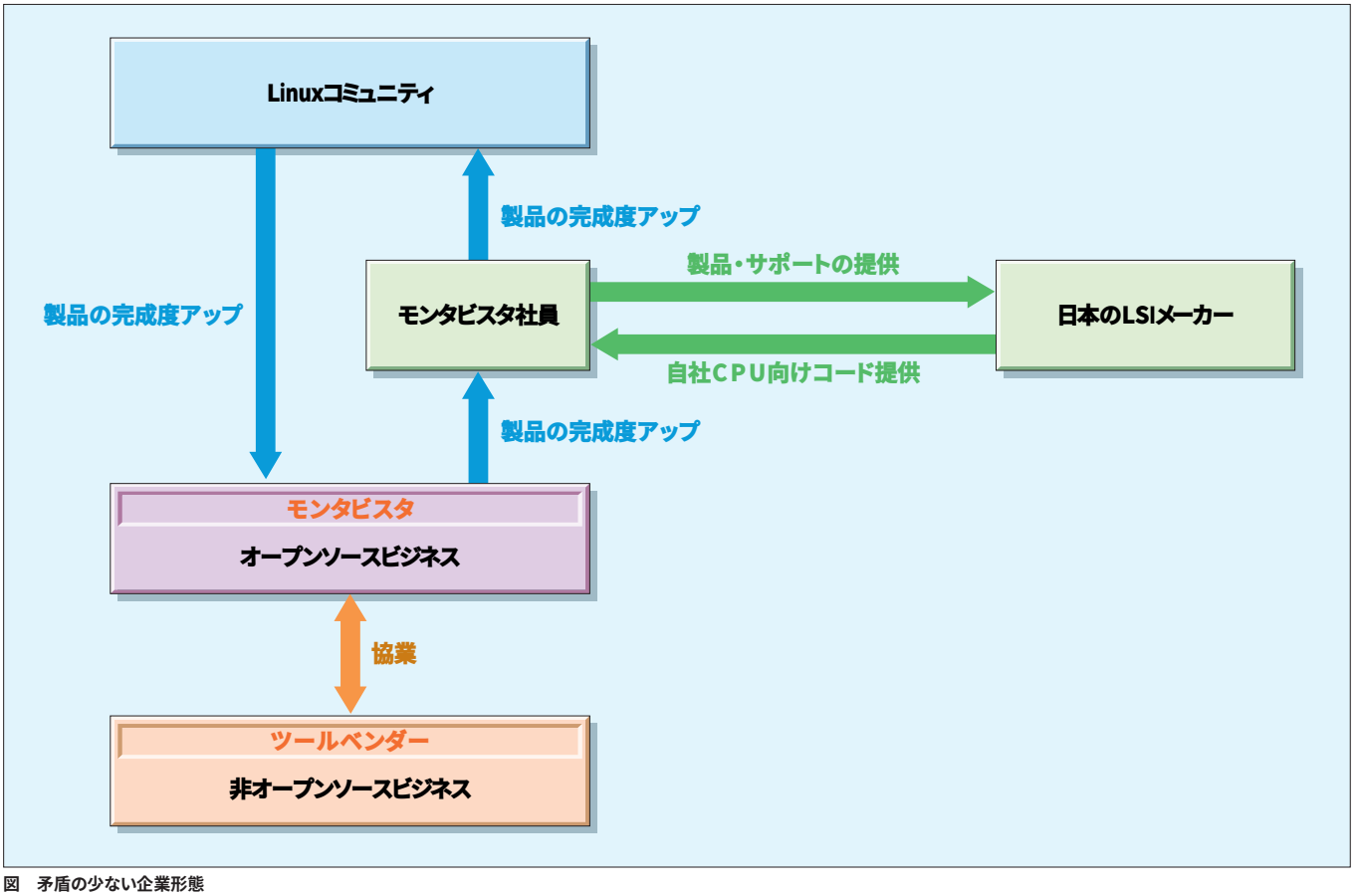


図　矛盾の少ない企業形態
内外に矛盾する要素をもたない企業形態が、モンタビスタの場合はそのままビジネスでの成功につながっている。

モンタビスタソフトウエアジャパン代表取締役社長 有馬仁志氏に聞く

Interview

大手組み込みLinuxディストリビューターとして、リネオと並んで「2大」というイメージのあるモンタビスタ。今回はその代表取締役社長である有馬仁志氏にインタビューをお願いしました。

――はじめまして。オフィスを移転されて広くなりましたね。

有馬：ええ、以前のオフィスでは手狭になってしまいまして……。現在、日本法人の従業員は15名なのですけれども、半導体メーカーさんから2名、それからパートナー会社から2名、大学から1名、テーマを持って勉強に来ている方がいるのです。

――いわゆる技術交流でしょうか？

有馬：いえ、半導体メーカーさんは、早くLinuxの品揃えをしないといけないという危機感を持っておられまして、自社の開発にもLinuxを使いたいということで勉強しにいらしゃっています。私どもが、そういう方々も受け入れて仕事をしましょうということでスペースも提供しております。ですから、手狭になってしまって移転を急いだのです。

写真　有馬仁志氏（ありまひとし）
モンタビスタソフトウエアジャパン株式会社代表取締役社長兼、モンタビスタソフトウェア日本セールス担当副社長。18年におよぶリアルタイムオペレーティングシステムの経験を持つ。Wind River Systems、Integrated Systemsの日本法人の管理職位、1999年にIntegrated Systemsに買収されたSoftware Development Systemsの日本法人の代表専務取締役を歴任。また、日本エンベデッドリナックスコンソーシアム（Emblix）では監査役として、日本における組み込みLinuxの普及活動に積極的に貢献。

――同じフロアにイーエルティ（本文参照）も入っているのですね。

有馬：ええ、イーエルティは兄弟会社で、「オープンなビジネスは私どもで、非オープンなビジネスはイーエルティで」という棲み分けをしているんですが、移転を機に同じ建物の同じフロアに入ってもらうことにしました。協力して効率よく仕事をしていく関係ですので……。

――それはいいですね。ところでモンタビスタさんはいつ設立されたのでしょう。

有馬：米国本社が1999年の3月、日本法人は2000年の7月です。日本法人ができてから1年と2カ月くらいですね。

――モンタビスタさんというと、組み込みLinuxの世界で非常にメジャーな存在というイメージがありますから、まだ設立から1年か2年しか経っていないというと少し違和感を感じます。

有馬：モンタビスタ自体の歴史は比較的新しいのですけれども、やっている人間の歴史は古いんです。

――とおっしゃいますと？

有馬：本社の社長はジム・レディなんです。「組み込みOSの父」と言われている人物で、VRTXという歴史の古い、もう20年以上も使われ続けているOSがあるのですけれども、それを開発した人です。彼が最初に作った会社はハンター・アンド・レディ、次いでレディシステム、マイクロテックリサーチ、それからメンター・グラフィックスというように、企業の吸収合併で会社は変わっていったのですけれども、その20年以上の長きにわたって、VRTXというリアルタイムOS、それからマイクロテックのツールの面倒をずっと見てきたのがジム・レディという人物なんです。

――凄い方ですね。

有馬：日本の社会でもVRTXというOSはずっと使い続けられているのです。そういうわけで、半導体ツールベンダー、

ミドルウェアベンダー、SI会社、もちろんお客さまも含めて、ジム・レディの知名度は非常に高いんです。そのジム・レディの会社ということで、モンタビスタは急速に受け入れられたところがあります。

――Linuxが組み込みに使えそうだと判断されたのは、ジム・レディさんご自身なんですか？

有馬：そうです。9年くらい前のことです。今、Linuxが誕生してちょうど10年と言われていますけど、ごく早い時期からジム・レディは注目していて、満を持してスタートしたという感じなんです。「Linuxは組み込みに使える」ということを信念にしてですね。

――前回取材したタクシアさんも、'92年ごろに世界最古の組み込みLinux製品を開発したとおっしゃっていました。それはごく普通のターミナルなんですが。

有馬：ええ同じ頃、'92年にLinuxの初期のバージョンをジム・レディが動かし、これは組み込みで使えるはずだとずっと温めてきたんですね。

――そういう経緯があってモンタビスタが設立されたわけですね。

有馬：モンタビスタを設立したのが99年3月なのですけれども、そのときに出資したベンチャーキャピタルは、ジム・レディが20年前にハンター・アンド・レディを設立したのと同じところなんです。この例を取って、いかにジム・レディという人物に信用があるかおわかりいただけると思います。

――結局は人なんですね。

有馬：そうですね。そして会社を設立するということで集まってきたのがLinuxのエンジニア、UNIXのエンジニア、それから従来の組み込みOSのエンジニアなのですが、その時にリアルタイムUNIXの世界で非常に有名なHPのリアルタイムUNIXを作っていた開発チームのトップだったケビン・モーガンが参加しまして……。結果として、Linuxコミュニティと、HP出身者と、従来の組み込みOSのエンジニア、これがバランスよく3分の1ずつ集まりました。

――Linuxコミュニティ出身者ばかりではビジネスはうまくいかないように思えますね。

有馬：そうですね。技術面はそういう感じでバランス良くなっています。それからビジネスサイドなんですが、やっぱり組み込みでVRTXの担当をずっとしていたようなセールスの人間など、組み込みシステムを熟知している者が集まってきまして、それでモンタビスタという会社が成り立っているという感じなのです。

――やはり企業は人なんですね。ところで、ワールドワイド展開も非常に速いというイメージを受けましたが？

有馬：ええ。昨年、日本をスタートする1カ月前にヨーロッパを立ち上げています。ヨーロッパは、まずオフィスをフランスに立ち上げ、その後すぐに日本オフィスが設立されました。続いて、ドイツ、イギリス、スウェーデンと4つ、ヨーロッパに作りました。このように、ワールドワイド展開は大変速かったですね。また、今年6月には、アジアにシンガポールオフィスも設立されています。

――Hard Hat Linuxのディストリビューションの出荷は確か昨年でしたね。

有馬：はい、2000年の3月です。本社が1999年3月にできてからちょうど丸1年間、ディストリビューションを一生懸命作ってきたんですよ。2000年3月に正式にリリースできたのが1.0という初期のバージョンですね。それが世界中のマーケットから非常に高い評価をいただけまして。みなさん、この組み込みのLinuxを待っていたという感じでした。

――それからワールドワイド展開されたわけですね。堅実な感じを受けます。

有馬：使いたいというお客さまが世界中にいらして……。日本の家電系のお客様が、アメリカの本社に「ぜひ使いたい」ということでかなり訪問をされていまして、もういくつも設計を始めていたということもありました。そういう理由でモンタビスタの拠点を日本でも早くスタートさせなくてはということになり、ご縁があって私が代表取締役社長ということになり、日本法人をスタートしました。

――ところで、以前からぜひ伺いたいと思っていたことがひとつあるんですが……。ディストリビューションの名前が「Hard Hat」ということは、レッドハットさんと何か関係があるんでしょうか？

有馬：直接の関係はないですね。名前の、まあ雰囲気をいただいちゃったという感じでしょうか（笑）。ただ、ゆるいパートナーシップと言ったらいいのでしょうか。私どもは、レッドハットのホストをサポートしていますから……。もちろん、Solarisも、SuSEも、Turbolinuxもサポートしているんですが、そのひとつということです。

――ほかの組み込みLinuxのディストリビューターさんとはどうでしょう？

有馬：Emblix（日本エンベデッドリナックスコンソーシアム）の中では「同じ業界」ということで仲良くさせていただいています。私もレッドハットの方とか、リネオの方と携帯の電話番号を共有して、会合があるときには「今、どこですか」なんてお話しながら会場に向かうという感じです。そこ

ではニュートラルに、組み込みLinux発展のために仲良くさせていただいています。

――競争するところは競争して、協調するところは協調してという感じですか。

有馬：Linuxを広めましょうというミッションとしては一緒なんですよね。やっぱり、多くの方にLinuxを使っていただくということが共通の課題ですので……。そのためにも、Emblixでは仲良くさせていただくということなんです。

――パイを大きくしてその中で各社、自分の持ち味で取り分を大きくしていきましょうという思想だと理解してよろしいんでしょうか。

有馬：はい、そのとおりです。もうパイが広がらないことにはどうにもなりませんから。ただ、最近の調査データを見てみますと、Linuxの今後の普及率の見込みは凄いんです。2005年までは組み込み市場の伸び率が150%という予測が出ています。

――毎年150%ということですか？

有馬：そうです。

――すごいですね。

有馬：150%ですからすごい伸び率ですよね。それはトータルで150%なんですけれども、その中で組み込みLinuxが占める比率は、今後2年間で36%まで伸びると言われています。組み込み市場自体が、年々150%ぐらい伸びていると。Linuxのシェアも相当広がっていくだろうというように言われておりますね。

――組み込みのすべてがLinuxというようにはならないでしょうけれども……。

有馬：これは私の感覚なんですけれども、日本ではITRONとLinuxがあったら十分ではないかと思います。Windows CEとか、VxWorksとか、OS-9とか、Nucleusとか、QNXとか、組み込みOSはいろいろあるのですけれども……。もちろん、用途によって「それでなくては」ということがありますので、それらのOSもやはり生き残っていくと思いますけれども……。

――Emblixでも、ITRONとLinuxとの共存ということを強く主張されていますね。

有馬：ITRONは今までの長い歴史がありますよね。それに対するLinuxのアドバンテージは、やっぱりネットワークだと思うのです。インターネットにつながる世界では、Linuxに大きなメリットがあります。

――Emblixの中島会長も、以前お会いしたときにそうおっしゃっていらっしゃいました。

有馬：ITRONのメリットは8ビットや16ビットの非力なCPUでもサクサク動くし、ソフトウェア資産もたくさんあるということなんですね。ですから、非力なCPUで十分で、なおかつインターネットにつなぐ必要のないところではITRON、インターネットにつながらなくてはならないところはLinuxという感じになるだろうと思います。

――現状を考えると妥当な棲み分けだと思います。

有馬：それに、ITRONはLinuxが日本の社会で受け入れられるためにも重要な役割を果たしていると思います。

――どういうことでしょう？

有馬：ITRONはLinuxと同様に、オープンなスタンダードだったのです。誰でもインプリメントできる。この文化の蓄積が日本にはあるのです。ですから、日本人の技術者は「Linuxはオープンソースで誰でも使えるよ」というのに非常に抵抗がないんです。

――そういう類似点が確かにありますね。

有馬：ITRONとLinuxはオープンスタンダードという共通点がありますから、Linuxは受け入れられやすかったと思います。

――私にとっては新鮮な見方です。

有馬：ですから、ITRONとLinuxが共存していれば、日本ではそれで十分ではないかと思います。8ビットとか16ビットとかの世界ではITRONのマーケットができあがってしまっていますし、そういう世界ではLinuxの良さが出せないですよね？

――おっしゃる通りです。

有馬：逆に、32ビットでMMUがあって、通信のためのインターネットのプロトコルが必要で、インテリジェントな機能が欲しいという場面では、Linuxはもう正しくうってつけだと思います。マーケットも半導体ベンダーもそう思っています。ということで、たとえばカーナビですとか、衛星放送の受信機、デジタルテレビといった、そういう家電向けマーケットでLinuxが今後加速度的に伸びていくのではないかと思います。

――海外では、8ビットや16ビットの世界はどういう状況なのでしょうか？

有馬：たくさんのOSがありますね。主なところではNucleusとか、レッドハットさんが買収されたeCosですとか。日本ではそれらがITRONに統合されているといったところです。

――状況はよくわかりました。ところで、今後の方向性について伺いたいのですが、現在の「Hard Hat Linux」はロイヤリティフリーでオープンにし、「Hard Hat Linux」のバイ

ナリパッケージと年間サポート契約をセットにして「サブスクリプション」で利益をあげるという形態でビジネスを行われてゆくのでしょうか？

有馬：そうですね。自社ですべてをまかなうという方向性には行かないと思います。他社ではよくありますが、買収を重ねて自社内で何もかもを持つという方向には私どもは行かないでしょう。それはオープンソースビジネスの姿ではないと思いますし。

――そうですね。

有馬：モンタビスタのミッションのひとつには、オープンソースビジネスのあるべき姿を追求するということもあると考えています。現在のところ、受け入れていただいているようでまことに喜ばしいのですが。

――これからの展開が楽しみですね。

有馬：そうですね。これから大阪にもオフィスを作る予定なんです。家電系のお客さまが大阪近辺に多数いらっしゃいますし。

――モンタビスタさんが今後注力していかれる分野は、どういったところでしょうか？

有馬：情報家電と通信インフラといったものですね。そのために、Hard Hat Linuxには両極端の機能が要求されているのですが……。

――と言いますと？

有馬：情報家電ではとにかく品揃えですね。もちろん、基本にはリアルタイム性やサイズの問題があるんですが、ブラウザとかグラフィックスですとか、Javaが動くことですとか、通信に関してはBluetoothですとか、無線LANに対応するとか、そういった品揃えが必要になっています。

――通信インフラではまたまったく異なるものが必要になりますね。

有馬：ええ、最も求められるのはハイアベイラビリティ（High Availability）です。信頼性が高く、稼働率が99.999%ですとか、非常に長い時間ノンストップでコンピューティングすることですとか……。

――地味ですが重要な技術ですね。

有馬：ハイアベイラビリティの対応も、私どもがオープンソースで対応しています。たとえば、ホットスワップですね。ボードを交換するときに電源を入れたまま交換できる仕組みを提供することも、信頼性を上げるための方策のひとつなんです。そのいろいろな方策を、私どもはオープンソースで作ったものを、オープンソースコミュニティで公開しながら提供しています。プログラムの作り方も信頼性には関係しますし、テストの作り方、それからホットスワップの例ではハードの作り方も重要になります。そういった「仕組み」も開発していますね。そういったことも含めて、組み込みLinuxの中では私どもがリーディングカンパニーだと思います。

――信頼性に関しては、ちょっとほかの追従を許さない感じになっているのですか。

有馬：そうですね。非常に信頼性の高いところで評価いただいています。私どもは、この高信頼性のためのいろいろな方策を公開情報として提供しているんです。従来は、有償の非常に高価なソフトを買ってきて作らなければならなかったのが、「オープンソースのLinuxで、オープンにされている情報を使って、私どものサービスを受けながら作ると、非常にコスト的にも折り合いますよ」というメッセージを発信しています。

――そうしますと将来的には、たとえばモンタビスタの開発した何かハイアベイラブルなものが、IEEEのスタンダードに取り入れられたりするというようなこともありえますか？

有馬：十分にありえると思いますね。私どもは、現在でもどんどんスタンダードの企画段階から参加していますし、オープンソースにしていっていますので、そういうスタンダードには相当協力しています。

――スタンダードへの協力というのは力がないとできないことですから、非常に楽しみだと思います。

有馬：そうそう、組み込みLinuxのリアルタイム性の追求ということで言えば、私どもで行ったカーネルのチューニング部分があるんですけれども、それも次のLinuxカーネル2.5で採用予定があるんです。

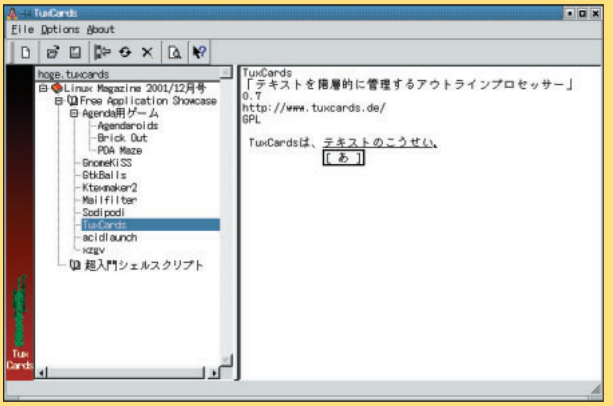
――コミュニティへの貢献とビジネスを同じ温度で行っていらっしゃる感じを受けます。ところで最後に伺いたいんですが、Hard Hat Linuxのシンボルマークのペンギンがかぶっているヘルメットは、何を意味しているんでしょう？

有馬：「安心して使えますよ」というメッセージです。Linuxそのものは、誰も保障してくれない。日進月歩だし、クオリティが保障されているのはカーネルしかないし、ツールキットで使おうと思ったりパッケージングしようとしたり、クオリティを評価しようとしたり、あるいはフォーム的に安心しようと思ったら、誰も保障してくれません。だからヘルメットかぶって、安心して我々のディストリビューションを使ってください（笑）。

――よくわかりました。今後、モンタビスタの技術がスタンダードになることも楽しみにしています。今日は大変ありがとうございました。

Free Application Showcase

文：出井 一
Text : Hajime Dei



TuxCards　P.146

Sodipodi　P.143

GtkBalls　P.154

※紹介したソフトは、すべて付録CD-ROMに収録されています。

SVGに対応した次世代ドローソフト

Sodipodi

バージョン：0.24.1　　ライセンス：GPL

http://sodipodi.sourceforge.net/

Sodipodiは、CorelDrawに似たインターフェイスを持つドローソフトだ。W3Cが標準化を進めている「スケーラブル・ベクタ・グラフィックス」（SVG）のサブセットに対応し、拡大してもジャギーのないベクタ画像を作成できる。gnome-printと組み合わせてプリンタで印刷したり、PS／PDFファイルに出力することも可能だ。実行にはgnome-printとGnome Applicationライブラリ（GAL）が必要となる。

●ライブラリのインストール

まずは、ライブラリのインストールを行う。印刷処理を行うgnome-print（0.21以降）は、最近のディストリビューションであれば最初からインストールされているはずだ。ただし、Sodipodiのビルドには、gnome-print-develパッケージも必要なので、ディストリビューションのCD-ROMからインストールしておこう。

画面1　メインウィンドウには、数多くのボタンがグループ別に並ぶ。

一方、Gnome Applicationライブラリ（GAL）は、バージョン0.8以降が要求される（最新版は0.14）。ほとんどのディストリビューションでは、新規インストールするか、バージョンアップする必要がある。

GNOMEのFTPサイトではtarボールのみ配布されているので、Red Hat系ディストリビューションでは、以下の手順でRPMパッケージを作成しよう。

「su -」としてrootになってから、「rpm -tb gal-0.14.tar.gz」とすると、/usr/src/redhat/RPMS/i386にgalとgal-develパッケージが作られる。これらを「rpm -Uvh gal-＊ 0.14-1.i386.rpm」（＊に注意）として、両方まとめてインストールしよう。なお、Kondara/MNU Linux 2.0を使っている場合はコラムを注意点を参照のこと。

●SodiPodiのインストール

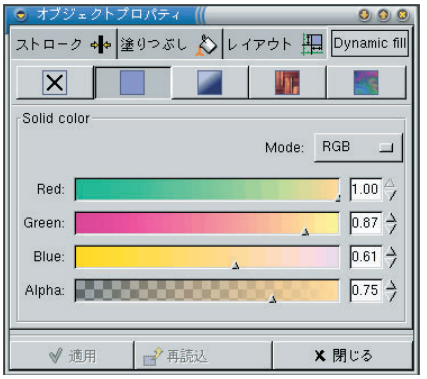
Sodipodiは、tarボールとRPMパッケージの両方で配布されている。tarボールの場合は、「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。

RPMを利用する場合、配布されているバイナリパッケージには日本語カタログが含まれていないため、以下の手順でソースパッケージからバイナリパッケージを作成しよう。

まず、「su -」でrootになった状態で、「rpm -i sodipodi-0.24.1-1.src.rpm」とする。/usr/src/redhat/SPECSにsodipodi.specが展開されるので、その末尾に「%{prefix}/share/locale/＊/LC_MESSAGES/sodipodi.mo」という行を追加しよう。

続いて、「rpm -bb sodipodi.spec」とすると、日本語カタログを含むバイナリパッケージが作成される。これを、「rpm -Uvh /usr/src/redhat/RPMS/i386/sodipodi-0.24.1-1.i386.rpm」としてインストールすればいい。

画面2　付属のサンプル画像（リスのOrav君）はSodipodiのマスコットだ。



画面3　オブジェクトの色などはプロパティダイアログで変更する。

Column

「Kondara 2.0にGALをインストールする際の注意点」

Kondara/MNU Linux 2.0では、Gnumericなど数個のソフトが以前のGAL（libgal.so.6）を参照するため、そのままではGALの最新版をインストールできない。依存性を無視したインストールを無理に行うと、Gnumericなどが使えなくなってしまう。

そこで、「su -」としてrootになった状態で、「cp -a /usr/lib/libgal.so.6.0.0 /tmp」として古いGALをいったん退避する。そのうえで、「rpm -Uvh --nodeps gal-＊ 0.14-1.i386.rpm」として、依存性を無視したインストールを行う。このとき、古いGALは自動的に削除されるので、「mv /tmp/libgal.so.6.0.0 /usr/lib」で書き戻し、「ldconfig」とすればOKだ。

●SVGとは

スケーラブル・ベクタ・グラフィックス（SVG）は、W3Cが勧告する2次元ベクタグラフィック記述言語だ（http://www.w3.org/Graphics/SVG/Overview.htm8）。XMLベースで可読性や可搬性が高いことから、次世代のWebページの一部として広く使われることが期待されている。

2001年の9月に最初の勧告（バージョン1.0）が出されたばかりの段階だが、すでにSVGに対応したドローソフトやコンバータ、WindowsのIE／Netscape用プラグイン（http://www.adobe.com/svg/）などが登場し、MozillaにSVGを実装するプロジェクト（http://www.croczilla.com/svg/）も進行中だ。

なお、Sodipodiでは、SVGの規格のうち、描画に関する部分にのみ対応しており、アニメーションやスクリプトといったインタラクティブな要素は実装されていない。

●Sodipodiの起動と画面構成

ktermなどで「sodipodi&」と入力するか、GNOMEメニューの[プログラム]－[グラフィック]－[Sodipodi]を選択すると、数多くのボタンが並んだメインウィンドウが開く（**画面1**）。付属の日本語カタログにより、メニューやダイアログ、チップヘルプのほとんどが日本語で表示される。

メインウィンドウは、ファイル、編集、オブジェクトなど機能別のグループに分類されており、境界部分のクリックで表示の有無を切り替え可能だ。また、境界部分の右端のアイコンをクリックすると、グループ単位で別ウィンドウに分離でき、柔軟なレイアウトで作業を行える。

ファイルグループの[新規描画]ボタンを押すと、新しいウィンドウが開いて白紙の画像が表示される。複数の画像のウィンドウを同時に開いたり、詳細図と全体図といった具合に、ひとつの画像に対して複数のウィンドウを開くこともできる。最初は、tarボールに含まれているサンプル画像を読み込んで、操作に慣れるといいだろう（**画面2**）。

このほか、画像上での右クリックメニューでもさまざまな操作を行える。たとえば、表示倍率の変更は、拡大グループのボタンを押す代わりに、右クリックメニューの[表示]－[拡大]以下から選択してもいい。

なお、拡大したために画像全体がウィンドウ内に収まらなくなった場合は、画像上でマウスの中ボタンをドラッグすることで、表示を自由にスクロールすることができる。

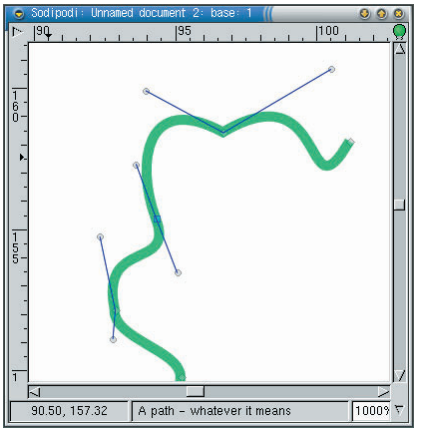
●オブジェクトの描画と選択

描画できる図形（オブジェクト）は、矩形／楕円／フリーハンド線／ダイナハンド線／テキストの5種類。Drawグループのボタンでオブジェクトの種類を選択し、画像上でドラッグして描画する。テキストの場合は、画像上でクリックしてから文字列をキー入力すればいい。同種のオブジェクトは連続していくつでも描画できる。

左ボタンは、このほかにもオブジェクトの選択やノードの操作などさまざまな場面で使われる。たとえば、Drawグループ左上の[Select and transform]ボタンを押してから、オブジェクトをクリックすると選択できる。ドラッグやShift－クリックによる複数オブジェクトの同時選択も可能だ。

オブジェクトを選択した状態では、

画面4　内部の色のAlphaパラメータを変更すれば半透明表示も可能だ。



画面5　フリーハンド線では、各ノードのハンドルで形状を調整できる。

内部のドラッグで移動したり、周囲の8つの矢印のドラッグでサイズを変更できる。さらに、再度クリックすると、オブジェクトの周囲の矢印が変化して、回転や変形が可能になる。

●オブジェクトのプロパティ

オブジェクトの太さや色などを変更するには、オブジェクトグループ上段の4つのボタンを押すか、右クリックメニューの[プロパティ]以下の項目を選択する。

ダイアログには、線の太さや色などを設定する「ストローク」、内部の色を設定する「塗りつぶし」と「Dynamic fill」（**画面3**）、位置などを設定する「レイアウト」などがある。

たとえば、Dynamic fillプロパティで、不透明度を表わすAlphaパラメータを減少させると、背後のオブジェクトが次第に透けて見えてくる。内部の色をグラデーションで表現することも可能だ（**画面4**）。

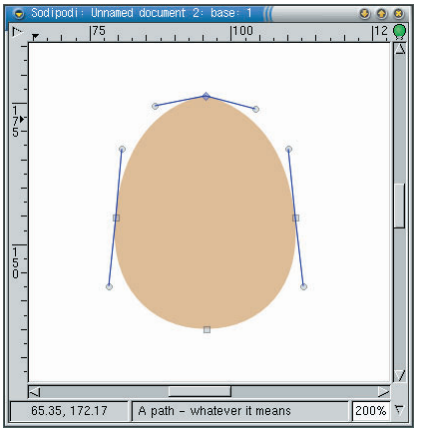
テキストのフォントや内容は、独立した「フォントスタイル」ダイアログで変更する。なお、現時点では日本語テキストには対応しておらず、入力した時点でSodipodiが異常終了してしまうので気をつけよう。

オブジェクトの前後関係は、選択グループのボタンで変更できる。また、複数のオブジェクトをまとめて扱うグループ化の機能も用意されている。

●ノードを編集して変形する

オブジェクトの種類が5つだけとは少なすぎると思われるかもしれない。実際には、どこでも太さが一定のフリーハンド線や、太さを場所により変えられるダイナハンド線を使って、ほとんどの図形を表現できる。

これらのオブジェクトには、適当な



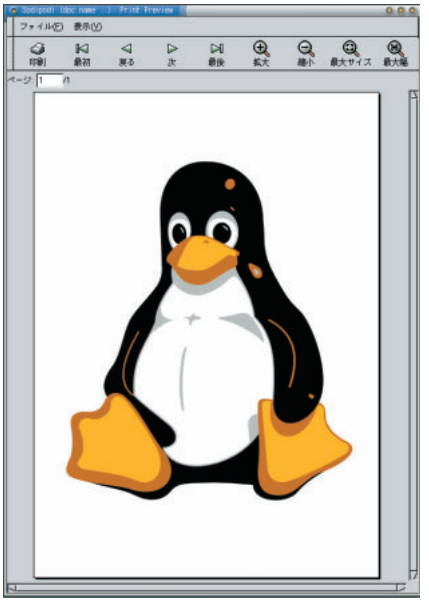
画面6　楕円をフリーハンド線に変換し、ノード編集で卵型に変形させた。

間隔で「ノード」が設定されており、描画後にノードを操作することで、形を滑らかに整えたり、大きく変形させることが可能だ。

対象となるオブジェクトを選択し、Drawグループの[ノード編集]ボタンを押すと、線上にノード（◇または□）が表示される。◇は尖った部分、□は滑らかな部分のノードを意味しており、相互に変換可能だ。これらを直接ドラッグして位置を変えたり、各ノードに2つ付属する「ハンドル」（○）をドラッグして、線の曲がり具合を調整してみよう（**画面5**）。

さらに、Nodesグループのボタンにより、ノードの追加や削除、分割などの操作も可能だ。なお、端点の結合や直線・曲線への変換といった操作を行うには、隣接する2つ以上のノードを、Shift－クリックにより選択しておく必要がある。

矩形／楕円／テキストオブジェクトからフリーハンド線への変換も可能だ。対象となるオブジェクトを選択してから、オブジェクトグループの[オブジェクトを曲線状に変形]ボタンを押せばいい。たとえば、楕円をフリーハンド線に変換した後、ノードを編集してゆがませると、卵型のオブジェクトを作成できる（**画面6**）。



画面7　印刷時のイメージはプレビューウィンドウで確認できる。

●印刷とファイルへの出力

Sodipodiで作成した画像は、ファイルグループの[描画の保存]ボタンにより、SVG形式で保存できる。このほか、プリンタに直接印刷したり、他のファイル形式にエクスポートする機能も用意されている。

ファイルグループの[描画印刷のプレビュー]ボタンを押すと、gnome-printのプレビューウィンドウが開いて、印刷イメージを確認できる（**画面7**）。表示の拡大・縮小や、印刷ダイアログの呼びだしも可能だ。

印刷ダイアログでは、PS形式に変換してlpr経由でプリンタにそのまま出力するほか、PS形式やPDF形式でファイルに保存することもできる。

このほか、ファイルグループの[出力]ボタンでは、画像全体や選択領域をビットマップに変換し、PNG形式でファイルに保存できる。Sodipodiで作成したロゴなどを、現行のWebページで利用するには、この方法を使うことをお勧めしたい。

テキストを階層的に管理するアウトラインプロセッサ

TuxCards

バージョン：0.7　　ライセンス：GPL

http://www.tuxcards.de/

TuxCardsは、テキストの構成を自由に組み替えられるアウトラインプロセッサだ。ひとまとまりのテキストを格納した「エントリー」を階層的なツリーで管理し、ドラッグ＆ドロップにより自由にツリー上を移動できる。作成したデータをHTMLファイルとして書き出して、MozillaなどのWebブラウザで閲覧することも可能だ。実行にはQt 2.2以降が必要となる。

●ビルドとインストール

TuxCardsは、tarボールとRPMパッケージが配布されている。ただし、そのままインストールしたのでは、フォント切り替えの際に日本語が文字化けしたり、エントリーのドラッグ＆ドロップで日本語の文字列が削除されてしまうという問題がある。

これらを解決するパッチ（tuxcards-0.7-i18n.patch）を筆者が作成したので、tarボールを利用して以下のように修正するとよいだろう。

まず、tarボールを展開したディレクトリに、本誌付録CD-ROMからパッチをコピーし、「gunzip -c tuxcards-0.7-i18n.patch.gz | patch -p1」としてソースを修正する。

あとは、「./configure」→「make」としてビルドし、「su」でrootになってから「make install」でインストールすればいい。

さらに、Cactus（サボテン）の画像データを「cp -P tuxcards/flowers/* /usr/local/share」として手動でコピーしておこう。

このほか、TuxCardsから利用するのに便利なアイコン集が別途tarボールの形で配布されている。こちらは、rootになった状態で、「tar zxvf icons.tar.gz -C /usr/local/share/tuxcards」として展開すればいい。

●起動と初期設定

ktermなどのコマンドラインで「tuxcards&」とすると、左右に2つのペイン（領域）を持つウィンドウが開く（**画面1**）。左側のペインには、階層的なエントリー（ノート）のツリー、右側のペインには選択中のエントリーの内容が表示される。

初めて起動した際には、以下の手順で初期設定を行おう。[Options]－[Edit Settings]で設定ダイアログを開いて、[Cactusbar Selection]の設定を「Enable Cactusbar」、ディレクトリを「/usr/local/share/tuxcards/flowers」に変更する（**画面2**）。

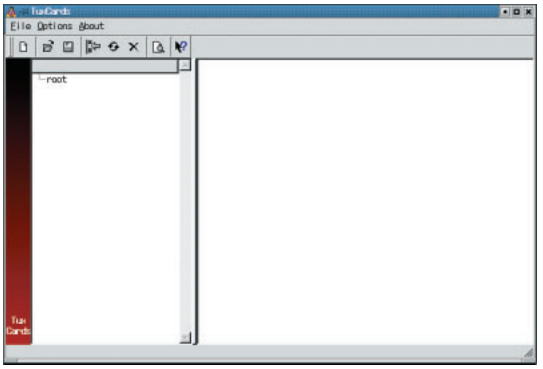
続いて、[Options]－[Change Font]でフォント選択ダイアログを開き、エントリー表示に使われるフォントを、「Gothic [aliastt]」など日本語を含むフォントに変更する。最後に、[Options]－[Save Settings]を選択して、現在の設定内容を設定ファイル（~/.tuxcards）に保存しておこう。

●エントリーの作成と編集

TuxCardsのツリーに新たなエントリーを追加するには、ツールバー中央の[Add Entry]ボタンを押すか、リスト中のエントリーを右クリックして[Add Entry]を選択し、エントリー追加ダイアログ（**画面3**）でエントリー名を入力する（日本語可）。

エントリー名の前にアイコンを表示したい場合は、[Icons]の設定を「use Icon」に変更し、右側のボタンを押してダイアログを開く。アイコン集を展開したディレクトリ（/usr/local/share/tuxcards/icons）などを指定すると、実際のアイコンを眺めながら選択可能だ（**画面4**）。

[Apply]ボタンを押すと、選択中のエントリーの下に、新たなエントリー



画面1　階層的なツリーで管理されたエントリーの内容が表示される。

画面2　まずは、設定ダイアログを開いて初期設定を行おう。

が追加される。なお、エントリー名やアイコンを後から変更するときには、対象となるエントリーを選択してから、ツールバーの[Change properties]ボタンを押せばいい。

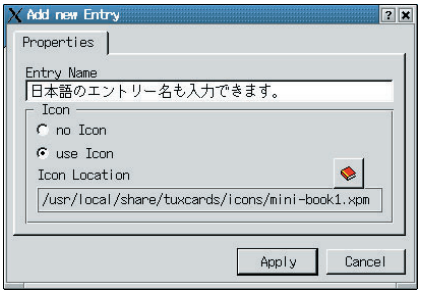
エントリーの内容は、そのエントリーを選択した状態で、右側のペインに入力する。もちろん、日本語も問題なく入力／編集可能だ（**画面5**）。

作成したデータは、ツールバーの[Save Data to File]ボタンでファイルに保存しておこう。なお、初期設定では自動保存機能が有効になっているため、15分ごとに自動保存が実行される（変更可能）。また、次回起動時には、最後に保存したファイルが自動的に読み込まれる。

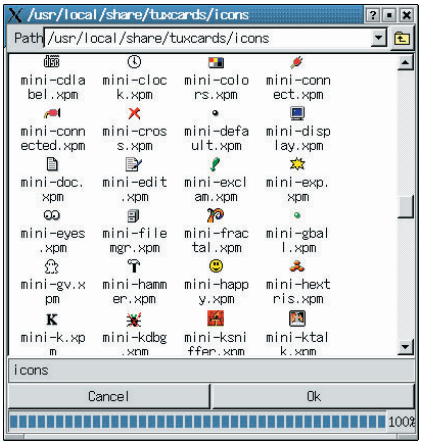
●エントリーの移動とコピー

各エントリーの左端の[−]ボタンをクリックすると、そのエントリーより下層のエントリーのツリー表示を隠すことができる。[＋]に変化したボタンをもう一度押せば再表示だ。

また、エントリーをドラッグ＆ドロップすると、その下層のエントリーも含めて、ツリーの別の位置に簡単に移動できる。こうして、バラバラに書いたエントリーをグループ分けしたり、上下関係を変更したりして、最終的な構成を考えていこう。



画面3　エントリーを追加するため、エントリー名とアイコンを設定する。



画面4　使用するアイコンをプレビュー機能付きのダイアログから選択。

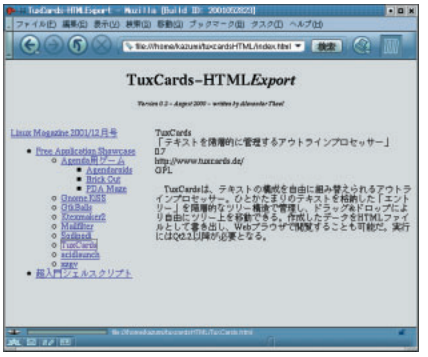
さらに、TuxCardsを複数起動しておけば、それらの間でエントリーをドラッグ＆ドロップできる（**画面6**）。この場合、エントリーの移動ではなくコピーが行われる。つまり、既存のデータの一部を簡単に流用できるわけだ。

●HTMLファイルにエクスポート

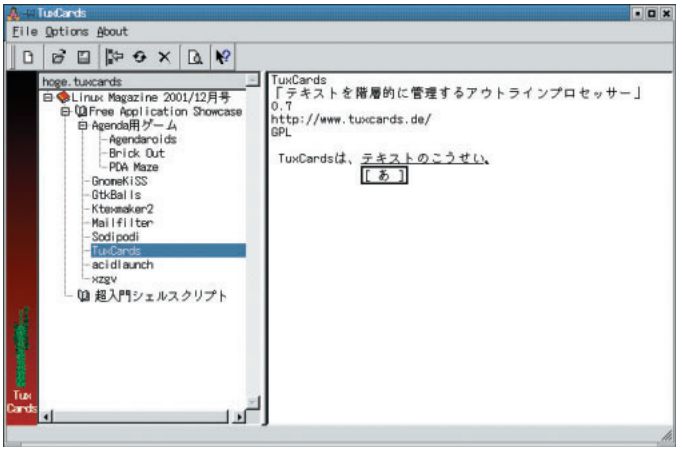
最終的な構成が確定したら、[File]−[Export to HTML]を選択しよう。ホームディレクトリにtuxcardsHTMLディレクトリが作成され、複数のHTMLファイルが格納される。

MozillaなどのWebブラウザを起動して、このディレクトリのindex.htmlを読み込むと、フレームを利用したページが開く（**画面7**）。左側のフレームに表示された各エントリーのリンクをクリックすると、その内容が右側のフレームに表示される。

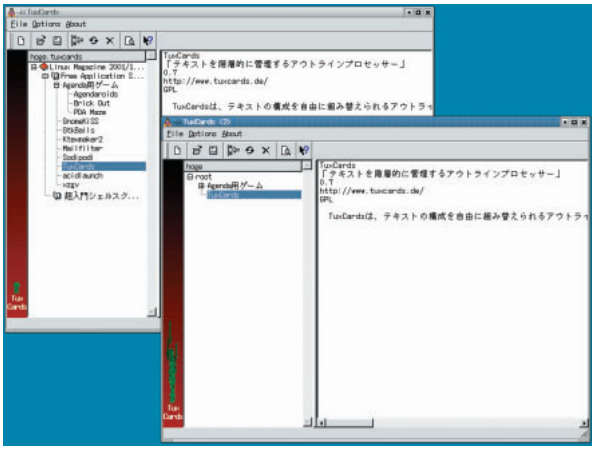
なお、現在の仕様ではエントリー名がそのままHTMLファイル名として使われる。このため、エントリー名が重複していると、リストの後方にあるエントリーの内容でHTMLファイルが上書きされてしまう点に注意されたい。



画面7　作成した文書をHTMLファイルにエクスポートしてMozillaで閲覧中。



画面5　エントリー名や内容には、日本語を問題なく入力できる。



画面6　複数のTuxCardsの間でエントリーをドラッグ＆ドロップできる。

KDE2上で動作するLaTeX文書処理の統合環境

Ktexmaker2

バージョン：1.6　　ライセンス：GPL

http://xm1.net.free.fr/linux/

Ktexmaker2は、LaTeX文書の作成やコンパイルをGUIで行える統合環境だ。カラー構文表示のソースエディタでLaTeX文書を作成し、コンパイルやプレビューをボタンひとつで実行できる。また、Gnuplotのフロントエンドとしても動作し、データファイルや関数に基づく2D／3Dグラフを簡単に作成可能だ。動作にはKDE 2.1／Qt 2.2.1以降に加え、LaTeX関係のツール一式が別途必要になる。

●ビルドとインストール

LaTeX関係のツールがインストールされていない場合は、ディストリビューションのCD-ROMなどからインストールしておこう。Red Hat系ディストリビューションでは、「tetex」で始まる複数のパッケージだ。

Ktexmaker2は、tarボールのほか、Red Hat／ Mandrake／ SuSE用 のRPMバイナリパッケージがそれぞれ配布されている。ただし、そのままインストールすると、日本語で書かれたセクションの切り替えに不具合が生ずる。

この問題を解決するパッチ（ktexmaker2-1.6-i18n.patch）を作成したので、tarボールを利用して以下のように修正するとよいだろう。

まず、本誌Webサイトのupdateページ（http://linux.ascii24.com/linux/linuxmag/update/）から、Ktexmaker2用のパッチをダウンロードし、「gunzip -c ktexmaker2-1.6-i18n.patch.gz | patch -p1」としてソースを修正する。あとは、「./configure」→「make」でビルドし、「su」でrootになった状態で「make install」としてインストールすればいい。

RPMを利用する場合は、別途配布されているSPECファイル（ktexmaker2-1.6-rh.spec）を修正して、tarボールからRPMパッケージを作成する。

まず、SPECファイルの5行目の「Source: …」の下に「Patch: ktexmaker2-1.6-i18n.patch.gz」、19行目の「%setup -q」の下に「%patch -p1」という行をそれぞれ挿入する。

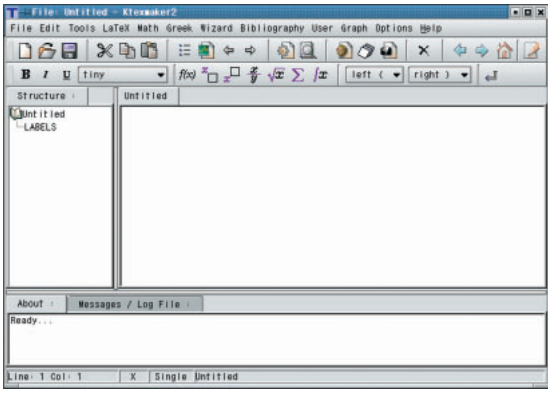
続いて、rootになった状態で、/usr/src/redhat/SOURCESにtarボールとパッチをコピーし、「rpm -bb ktexmaker2-1.6-rh.spec」とすると、/usr/src/redhat/SPECS/i386以下にバイナリパッケージが作成される。これを、通常の手順でインストールすればいい。

●起動から初期設定まで

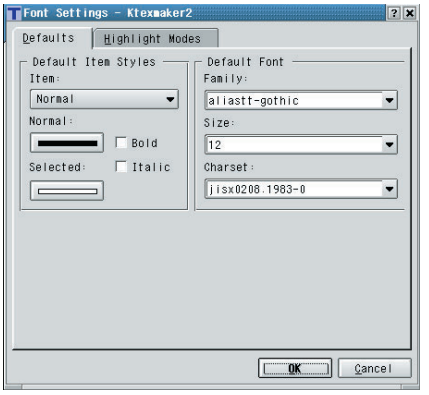
ktermなどのコマンドラインで「ktexmaker2&」とするか、KDEメニューの[アプリケーション]－[ktexmaker2]を選択すると、2段のツールバーと3つのペイン（領域）で構成されるウィンドウが開く（**画面1**）。

初めて起動した際には、フォントなどの初期設定を行おう。[Options]－[Editor Fonts and Hilighting]で設定ダイアログを開き、[Default Font]に「aliastt-gothic」など日本語フォントを選択する（**画面2**）。

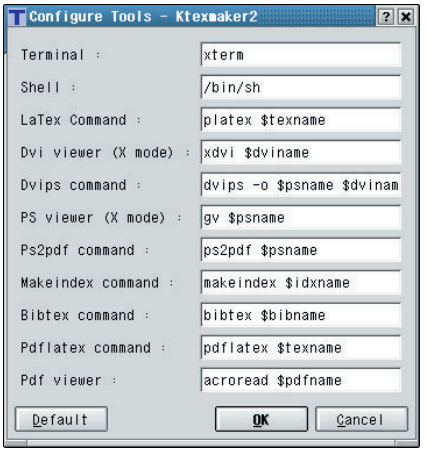
続いて、[Options]－[Configure Tools]を選択すると、Ktexmaker2から起動されるツール類が一覧表示される。たとえば、LaTeX文書のコンパイルにpLaTeXを使う場合は、[LaTex



画面1　ウィンドウはソースエディタなど3つのペインで構成される。



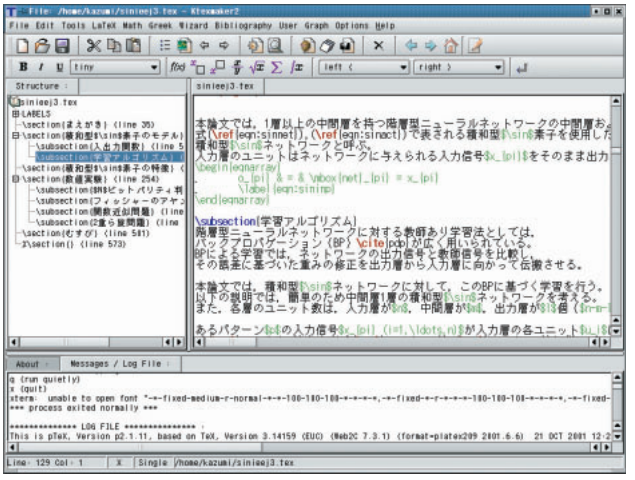
画面2　エディタで使用するフォントを設定。日本語も利用可能だ。



画面3　Ktexmaker2から起動される各種ツール類をカスタマイズする。

Command]の設定を「platex $texname」に変更しよう（**画面3**）。

なお、ツール類はKDE系とX系を切り替えられる。KDE系のツール（KDVIなど）は日本語表示に対応していないため、[Options]－[Toggle KDE/X mode]を選択してX系のツールを利用する設定にしておこう。



画面4　LaTeX文書の構造のツリー表示やカラー構文表示が行われる。

●カラー構文表示のエディタ

ウィンドウ中央の大きなペインは、ソースエディタで、LaTeX文書やその他のテキストを編集できる。複数のファイルを編集する際は、上部のタブで切り替えよう。

LaTeX文書の場合は、ファイルの内容が解析され、コマンドやコメント部分がカラー構文表示されるとともに、セクションやラベルのツリーが左のペインに表示される（**画面4**）。

ツリー中の項目をクリックすると、対応する行にソースエディタのカーソルが移動するので、大きなファイルでも素早くセクション間を移動可能だ。また、項目が別のファイルを指している場合（「input」や「include」など）は、ダブルクリックで対応するファイルが読み込まれる。

ツールバーには、LaTeX文書を書くのに便利なボタンが多数用意されている。タグ入力式のHTMLエディタを思い浮かべてもらえばいい。

たとえば、フォントの大きさを変えるには、ツールバーのリストから目的のサイズ（「tiny」など）選択すればいい。分数、平方根、合計などの数式記号もボタンで入力できる。

また、メニューの[LaTeX]以下には、セクション区分や箇条書きなどの環境を入力するための項目が並んでいるし、[Wizard]以下には配列や表組みの外形を作成するウィザードが用意されている。これらを使えば、さまざまなLaTeXのマークアップコマンドを素早く入力できるだろう。

●文書のコンパイルとプレビュー

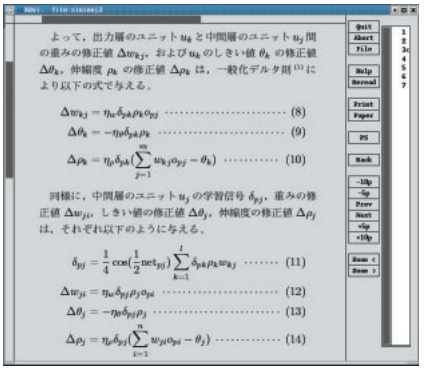
ツールバーの[Compile with LaTeX]ボタンを押すと、xtermのウィンドウが開いて、LaTeX文書のコンパイルが行われる。エラーが起きたら、プロンプト「?」に対して「x」を入力してウィンドウを閉じよう。

Ktexmaker2のウィンドウに戻り、ツールバーの[View 'Log File']ボタンを押すと、下のペインにコンパイル時のログファイルが表示される。ログ中の行番号をクリックすると、ソースエディタのカーソルが対応する行に自動的に移動するので、効率的にエラーを修正できる。

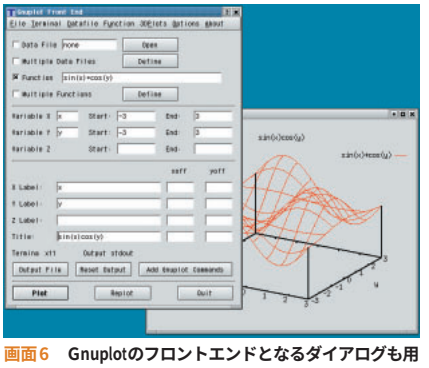
正常にコンパイルできたら、今度はツールバーの[DVI Viewer]ボタンを押そう。Xdviが起動されてdviファイルをプレビューできる（**画面5**）。PSファイルに変換してGVでプレビューすることも可能だ。なお、Ktexmaker2には、ウィンドウ内でプレビューする機能も用意されているが、日本語を含むLaTeX文書の場合は何も表示されないので注意されたい。

このほか、[Graph]－[Gnuplot Front End]で開くダイアログは、Gnuplotを起動して、データファイルや関数に基づく2D／3Dグラフを作成するためのフロントエンドだ（**画面6**）。

このダイアログには、テキストフィールドやボタンだけでなく、メニューバーも用意されており、グラフの描画形式や描画サイズの指定などをメニューから制御できる。



画面5　ツールバーのボタンを押すだけで、Xdviによるプレビューが可能。



画面6　Gnuplotのフロントエンドとなるダイアログも用意されている。

SPAMメールをPOP3サーバ上で削除する

MailFilter

バージョン：0.2.4　　ライセンス：GPL

http://mailfilter.sourceforge.net/

MailFilterは、SPAMメールの削除を目的としたフィルタリングソフトだ。単独で動作し、複数のPOP3サーバ上のメールを削除できるのが特徴だ。このため、使っているメールクライアントにかかわらず気軽に利用でき、ダイアルアップ環境でも威力を発揮する。サブジェクトや送信者アドレスなどのヘッダ情報に対して、正規表現を利用した柔軟で強力なSPAMチェックが可能だ（日本語には未対応）。

●ビルドとインストール

MailFilterは、tarボールとRPMパッケージが配布されているので、ディストリビューションに合わせて選択しよう。tarボールを利用する場合は、「./configure」→「make」でビルドし、「su」でrootになってから「make install」としてインストールする一般的な手順だ。

RPMを利用する場合は、バイナリパッケージをそのままインストールするか、またはソースパッケージを「rpm --rebuild mailfilter-0.2.4-1.src.rpm」としてリビルドし、/usr/src/redhat/RPMS/i386に作成されたバイナリパッケージをインストールすることもできる。

●設定ファイルをコピーする

MailFilterを起動する前に、接続するPOP3サーバや、接続に利用するユーザー名とパスワード、SPAMメールと判断するルールなどを、ホームディレクトリの設定ファイル（.mailfilterrc）に記述する必要がある。

設定ファイルのサンプルが、tarボール展開先のdocディレクトリ（RPMパッケージの場合は/usr/share/doc/mailfilter-0.2.4ディレクトリ）に存在する。これを、

```
$ cp doc/rcfile.example1 ~/.mail
filterrc
```

などとして、ホームディレクトリの.mailfilterrcにコピーしよう。

このファイルには、POP3サーバ用のユーザー名やパスワードが平文で記述されるので、

```
$ chmod 600 ~/.mailfilterrc
```

として、他のユーザーからアクセスできないようにしておく。また、

```
$ mkdir ~/logs
```

として、ログファイル保存用のディレクトリを作成しておこう。

●設定ファイルを変更する

設定ファイルはテキスト形式なので、好みのテキストエディタを起動して書き換えればいい（**画面1**）。以下では、サンプルの設定ファイルから変更すべき点を説明する。

なお、「＝」の両側に空白を入れたり、記述する順番を入れ替えたりすると、MailFilterが設定を読み込めなくなるので気をつけよう。

（1）POP3サーバの設定

まずは、31行付近から始まるPOP3サーバに関する設定だ。

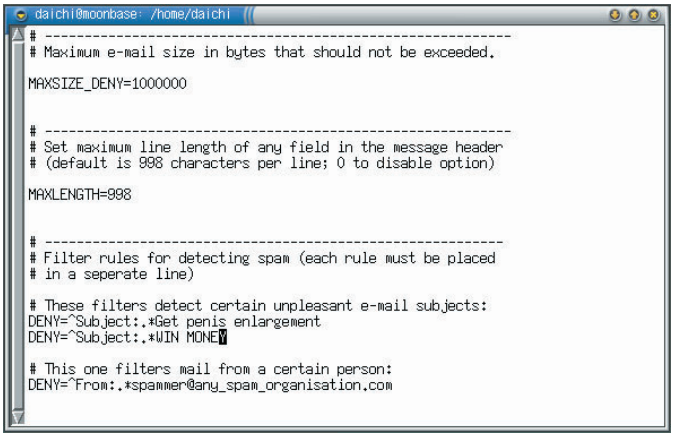
```
SERVER=POP3サーバ名
USER=ユーザー名
PASS=パスワード
```

という順番で、接続先のPOP3サーバの情報を記述する。「PROTOCOL」と「PORT」については、初期設定のまま変更しなくていい。

サンプルに、2つのPOP3サーバの例が記述されていることからもわかるように、MailFilterでは複数のPOP3サーバに対するSPAMチェックが可能だ。接続先がひとつだけなら、2つめの設定部分の行頭に「#」を付けてコメントにしておこう。

（2）サイズによる制限

64行目付近の「MAXSIZE_DENY」



画面1　まずは設定ファイルをテキストエディタで編集する。

は、添付ファイルなどが原因の巨大なメールを削除する設定だ。ここで指定されたサイズ（バイト単位）を超えるメールは、MailFilterにより自動的にPOP3サーバから削除される。

なお、友人らと巨大なメールをやりとりしている場合は、後述する「ALLOW」設定により、ここで設定したサイズ制限を回避できる。

(3) フィルタルールの設定

77行目付近から始まる「DENY」や「DENY_CASE」という数行が、メールのヘッダ情報に対するフィルタルールの設定だ。ここでは、行頭を表す「^」や、任意の文字列を表す「.*」など、正規表現の基本的な特殊記号（メタ文字）を利用できる。

たとえば、

```
DENY=^Subject:.*WIN MONEY
```

は、行頭が「Subject:」で始まり（件名）、任意の文字列が続いたあとで「WIN MONEY」という文字列が登場するヘッダを持つメールを自動的に削除するというルールだ。

これらのサンプルを参考にして、今までに受け取ったSPAMメールのサブジェクトなどからDENYの設定をいくつか追加してみよう。

なお、DENYは英字の大小文字を区別しないので、区別が必要な場合はDENY_CASEを利用すること。

(4)「友人」の設定

フィルタルールやサイズ制限の設定によっては、信頼できる友人や会社からのメールが自動的に削除されてしまう事態が起こりうる。MailfilterによってPOP3サーバから削除されたメールは復活できないので注意されたい。

こうした事態を避けるため、115行付近から始まる「ALLOW」に、フィルタルールやサイズ制限の対象外となるメールアドレスやサブジェクトなどを設定しよう。

たとえば、

```
ALLOW=^From:hogehoge@hoge.com
```

とすると、「hogehoge@hoge.com」から送られてきたメールは、ヘッダの内容やサイズにかかわらず、削除されないようになる。

また、107行付近からの「MAXSIZE_ALLOW」に0以外のサイズ（バイト単位）を指定すると、「ALLOW」で指定した友人に対し、SPAMよりゆるやかなサイズ制限を設けられる。

●MailFilterを実行する

ひとつ注意すべき点として、「設定ファイルに空白のみの行が含まれていてはいけない」という点が挙げられる（改行のみの行はOK）。こうした行を取り除くには、Perlを利用して、

```
$ perl -pi -e 's/^ +$//' ~/.mailfilterrc
```

とするのが簡単だ。

設定ファイルが正しく記述されていれば、MailFilterの実行自体はとても簡単だ。ktermなどのコマンドラインで「mailfilter」とすると、POP3サーバに順次アクセスしてヘッダを調べ、フィルタルールやサイズ制限にひっかかるメールを自動的に削除してくれる（**画面2**）。

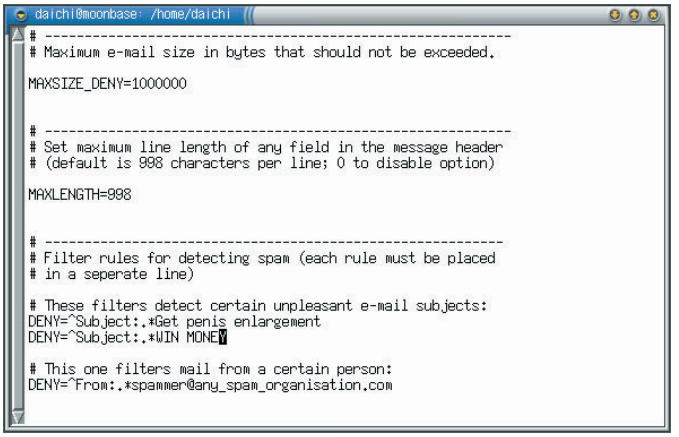
実行時刻や削除したメールのヘッダ情報などが、ログファイル（~/logs/mailfilter.log）に保存されるので、画面がスクロールしてしまった場合はこちらを確認するとよいだろう。

●さらに柔軟なチェックも可能

DENYやALLOWの設定で利用できる正規表現は、正規表現に慣れていない人に配慮した簡易版だ。たとえば、ドメイン名に含まれる「.」は、本来は「\.」と書かなくてはならない。

設定ファイル中の「REG_TYPE」の設定を、「basic」から「extended」に変更すると、選択を意味する「|」や、範囲を指定するカッコなど、本格的な正規表現のメタ文字を利用できるようになる。正規表現に慣れたユーザーなら、これらを利用してさらに柔軟なSPAMチェックが可能だ。

ただし、ドメイン名の「.」は「\.」と記述するなど、設定内容が複雑になってしまう。詳細は、もうひとつの設定ファイルのサンプル（rcfile.example2）を参照されたい。



画面2　POP3サーバにアクセスしてSPAMメールを削除する。

GNOME上で動作する着せ替えシステムKISSローダ

GnomeKiSS

バージョン：1.0　　ライセンス：GPL

http://www.ecs.soton.ac.uk/~njl98r/code/kiss/

GnomeKiSSは、日本で作られた着せ替えシステム規格KISS/GSや、その拡張仕様であるFrench-KISS!、CherryKissに対応したアプリケーションだ。世界中で作られたさまざまなKISSデータを読み込んで、人物（たいがいは女の子）の着せ替えや、インタラクティブなパズルなどのコンテンツを楽しむことができる。LHAでアーカイブされたKISSデータを直接扱うことも可能だ。動作にはGNOMEとlhaが必要となる。

●ビルドとインストール

GnomeKiSSは、ソース一式をtar＋gzipしたtarボールのみ配布されている。「./configure」→「make」でビルドし、「su」でrootになってから「make install」としてインストールする一般的な手順だ。

MIDIデータの再生にはPlaymidiが使われる。Timidityを利用したい場合は、tarボール展開先のsrc/event.cの132行目を「execlp（"timidity", "timidity", "-idq", song, NULL）;」に変更してからビルドすればいい。

実行時には、Linux用のlhaも必要になる。環境変数PATHに設定されたディレクトリにlhaが見当たらない場合は、ディストリビューションのCD-ROMなどからインストールしておこう。

●収録したKISSデータについて

KISSのデータは、グラフィック部品となる「セル」をはじめ、カラーパレットやコンフィグファイルなど多数のファイルで構成され、LHAでアーカイブされた状態で配布される。

本誌付録CD-ROMには、以下の3データを収録している。

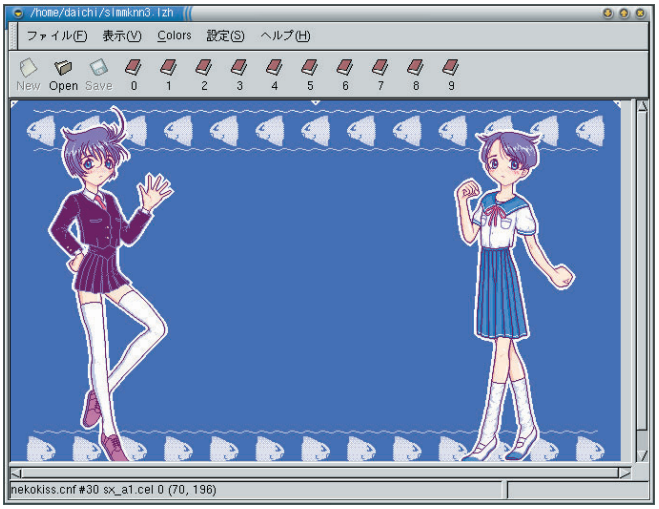
- **「○○ぢなるネコっぽい（＊>_<＊）むしゅめ」（slmmknn3.lzh、SiLent's MerMaid氏作）**
- **「太陽がいっぱい!」（fk_sspzlv10_midi.lzh、kazMaster氏作）**
- **「分割パズル壊れたチェス盤」（fk_cpzl2.lzh、kazMaster氏作）**

GnomeKiSSではLHAでアーカイブされたKISSデータをそのまま読み込めるので、ホームディレクトリなどにコピーしておけばいい。

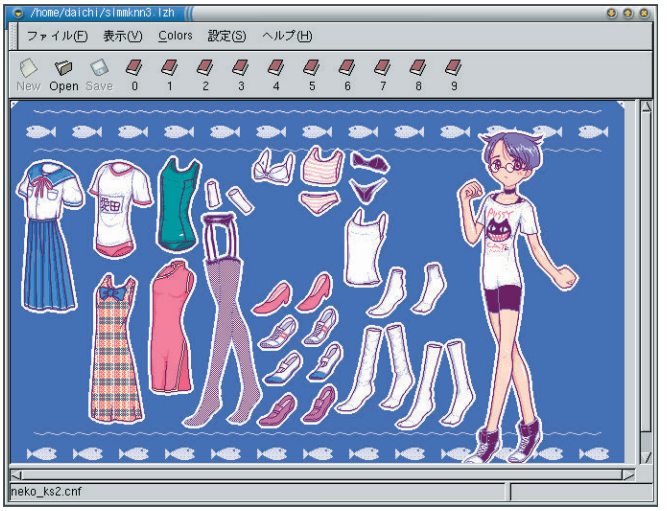
●KISSの歴史と規格

「KISS」（KIsekae Set System）の歴史は、PC-9801シリーズのMS-DOS用フリーソフトとして、MIO.H氏が作成した着せ替えソフト（KISS.EXE）から始まる。画面上でマウスを使ってリアルタイムに着せ替えを楽しめるこのソフトは多大な支持を集め、当時使われていたさまざまなパソコン用にローダ（ビューア）が作られた。

画面1　GnomeKiSSのウィンドウ。まずは[Open]ボタンを押そう



画面2　雰囲気がなんとも言えない「ネコっぽいむしゅめ」（ネコぽむ）



画面3　コンフィグファイルを切り替えると画面の構成が変化する。

そこで、KISSデータの標準規格として策定されたのが「KISS/GS」だ。複数のカラーパレットや最大256色の多色化を実現し、現在のKISSローダのほとんどがこのKISS/GS規格に対応している。

このほか、拡張規格としてアニメーションやサウンド再生などを実現する「French-KISS!」（fkiss）、TrueColor化や半透明表示などを実現する「CherryKISS」（ckiss）が存在する。また、海外ではfkissを独自に拡張したFKiSS2、FKiSS3も使われている。

GnomeKiSSでは、これらの規格のすべてに対応しており、国内外で提供されるさまざまなKISSデータを楽しむことができる。

●着せ替えを行う

ktermなどのコマンドラインで「gnomekiss&」として起動すると、多数のツールバーを持つウィンドウが開く（**画面1**）。

以下では、KISS/GS規格に基づいた「○○ぢなるネコっぽい（＊>_<＊）むしゅめ」（以下「ネコぽむ」と表記）を例として、GnomeKiSSの基本的な操作方法を説明しよう。

まず、ツールバー左の[Open]ボタンを押してダイアログを開き、「ネコぽむ」のアーカイブファイル（slmmknn3.lzh）を選択する。続いて、ダイアログにコンフィグファイル一覧が表示されるので、どれかひとつ（たとえば「nekokiss.cnf」）を選択する。

すると、コンフィグファイルの設定に基づいてKISS画像が表示される。複数の「ビュー」を、ツールバーの[0]～[9]ボタンで切り替え可能だ。「ネコぽむ」の場合、ビュー0は表紙にあたるので、実際の着せ替えはビュー1以降で行う（**画面2**）。

あとは、マウスのドラッグを利用して、服や靴、アクセサリなどを好きなように着せ替えればいい。また、[Colors]以下のメニューでカラーパレットを切り替えると、肌や服の色合いが微妙に変化する。

このほか、[ファイル]－[Open Configration]でダイアログを開き、別のコンフィグファイル（「neko_ks1.cnf」など）に切り替えると、同じセルを使っていても、画面構成の異なるKISS画像が表示される（**画面3**）。

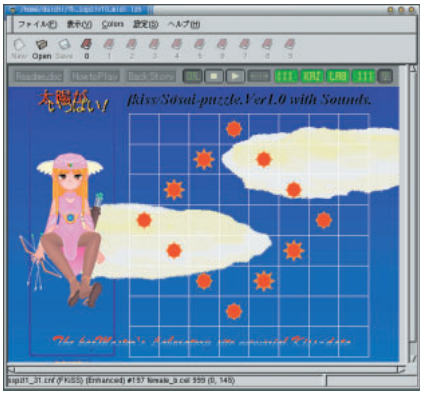
●fkiss対応ゲームで遊ぶ

French-KISS!で拡張されたイベント駆動型のスクリプトを使うと、単なる着せ替えにとどまらないインタラクティブなKISSデータを作成できる。代表的な例として、パズルゲームを実現したKISSデータ「太陽がいっぱい！」（**画面4**）を取り上げよう。

[Open]ボタンで「fk_sspzlv10_midi.lzh」を読み込み、ひとつしかないコンフィグファイルを選択すると、ゲームの盤面が表示される。左側の娘（実は神様見習い）をクリックするとプレイ開始だ。

ルールは簡単で、盤面上に表示された太陽をクリックして、太陽を盤面から消していけばいい。左上の[HowtoPlay]ボタンを押すと、ポップアップウィンドウが開いて、詳しいルールの説明が表示される（**画面5**）。

画面5　ゲームのルールなどがポップアップウィンドウに表示される。

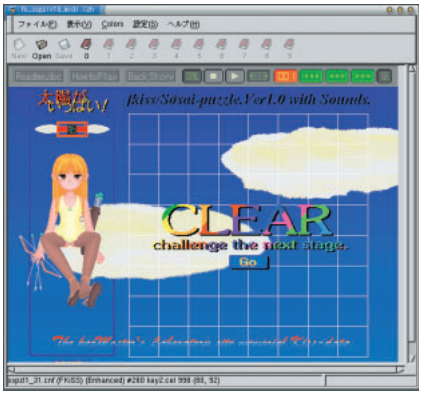


画面4　「太陽がいっぱい!」はfkissの拡張機能を利用したパズルゲーム。

盤面からすべての太陽を消せればクリアで、次の面に移る（**画面6**）。面の始めからやり直したい場合は、左下の[RESET!]ボタンを押せばいい。

また、右上にはBGM演奏用のボタンが並んでおり、[OF.]ボタン→[001]ボタン→[再生]ボタンの順でクリックすると、Playmidi（またはTimidity）によるBGMが繰り返し再生される。ただし、[停止]ボタンが機能しないなどGnomeKiSSでは実験的な機能だ。

French-KISS!の拡張機能を利用したポップアップウィンドウやボタンは、同じ作者の「分割パズル壊れたチェス盤」でも使われている。こちらは、チェス盤に複数のピースをはめ込むパズルゲームだ。



画面6　すべての太陽を消すとクリアで、次の面に進むことができる。

思考力と運の強さが試されるパズルゲーム

GtkBalls

バージョン：2.0　　ライセンス：GPL

http://gtkballs.antex.ru/

GtkBallsは、「Lines」と呼ばれるポピュラーなパズルゲームのクローンだ。9×9マスの盤面に配置された7色の駒を動かし、同色の駒を5つ以上揃えて消していく。駒を動かすたびに新たな駒が3個ずつランダムな位置に補充されるので、手詰まりを防ぐ先読みの力と運の強さを試されるゲームだ。盤面や駒のデザインは、テーマ機能により簡単に切り替えられる。実行にはGTK＋が必要だ。

●ビルドとインストール

GtkBallsは、tarボールとRPMパッケージの両方が用意されている。tarボールからのビルドとインストールは、configureスクリプトを利用する一般的な手順だ。

RPMパッケージはi686用のバイナリパッケージのみ配布されている。i386用などのパッケージが必要な場合は、以下の手順でtarボールからバイナリパッケージを作成しよう。

まず、tarボール展開先のディレクトリにあるgtkballs.specをエディタで編集し、91行目の末尾の「.gz」を「.*」に修正する。

続いて、「su」としてrootになった状態で、tarボールを/usr/src/redhat/SOURCESディレクトリにコピーし、「rpm -bb gtkballs.spec」とすればいい。/usr/src/redhat/RPMS/i386以下にバイナリパッケージが作成されるので、通常の手順でこれをインストールしよう。

●同色の駒を5個以上並べて消す

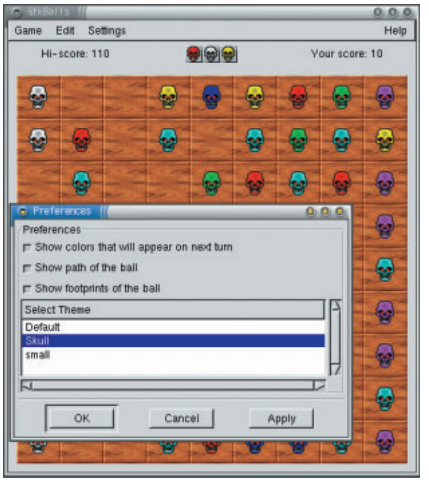
ktermなどのコマンドラインで「gtkballs&」として起動すると、9×9マスの盤面を持つウィンドウが開いてプレイが始まる（**画面1**）。

盤面には5つの駒がランダムに配置されている。そのひとつをクリックで選択し、動かしたい先のマスをクリックして移動させよう。駒は、他の駒にブロックされていない範囲なら、何マスでも移動できるので、なるべく他の駒を邪魔にならないように動かそう。移動が終わると、ウィンドウ上部に表示された3個の駒が、盤面のランダムな位置のマスに追加配置される。

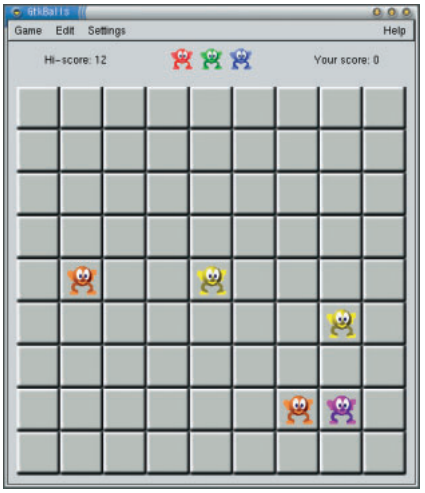
駒は全部で7色用意されており、同種の駒を縦・横・斜め方向に5個以上揃えると、それらの駒が消えて点数が入る（**画面2**）。一方、すべてのマスが駒で埋まってしまうと手詰まりとなってゲームオーバーだ。

当然のことながら、多くの駒を揃えたほうが点数が高い。しかし、揃える駒の数を増やすには多くの手順が必要なので、その分手詰まりになる危険も増すわけだ。

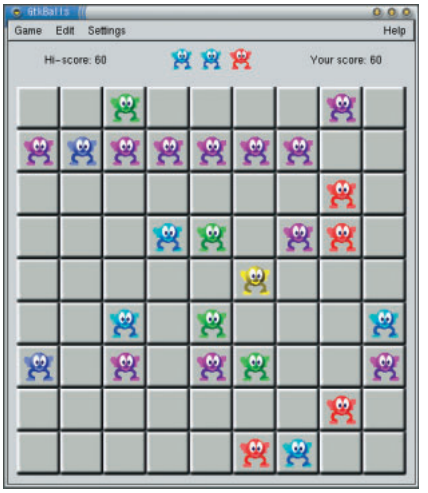
盤面や駒の外見は、[Settings]－[Preferences]で開く設定ダイアログで切り替えられる（**画面3**）。デフォルトテーマのほか、駒が骸骨の「Skull」、コンパクトな盤面の「small」が用意されている。



画面3　設定ダイアログのテーマで盤面や駒の外見を切り替えられる。



画面1　9×9マスの盤面に5つの駒がランダムに初期配置されている。



画面2　同色の駒を5つ以上、縦・横・斜めに揃えれば消すことができる。

PCのX上でも動作するAgenda VR3用ゲーム3種

Agendaroids／Brick Out／PDA Maze

バージョン：2001.10.09/2001.10.05/2001.10.05　ライセンス：GPL

http://www.newbreedsoftware.com/

Agendaroids／Brick Out／PDA Mazeは、いずれもLinuxベースのPDA「Agenda VR3」用のミニゲームだ。それぞれ、アステロイド風シューティング、ブロック崩し、3D迷路ゲームだ。Agenda VR3用にビルドされたバイナリが付属するので、クロス開発環境がなくてもインストールできる。また表示はコンパクトだが、PCのX Window System上でプレイすることも可能だ。

●ビルドとインストール

3つのゲームともtarボールのみ配布されている。アーカイブにはSNOW化バイナリ（＊.snow）が含まれるので、Agenda VR3を持っている人はそのままインストールすればいい。

PCのX Windows上で動作させるには、tarボールの展開先ディレクトリで「make host」としてソースからビルドする（PC用のバイナリは「.host」という拡張子）。インストールは、「su」としてrootになった状態で、「cp ＊.host /usr/local/bin」などとして手動でコピーすればいい。

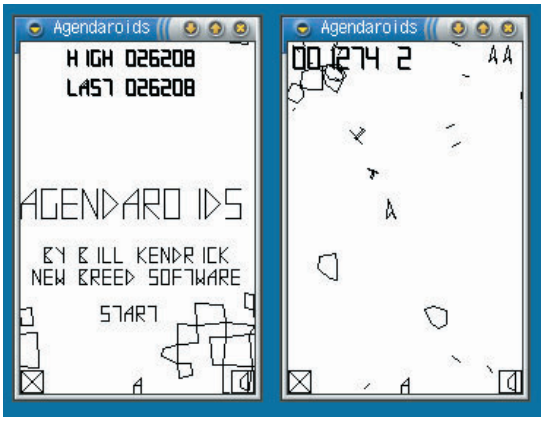
●ゲームをプレイする

以下では、PC上での各ゲームのプレイ方法を簡単に説明しよう。

• Agendaroids（画面1）

「agendaroids.host&」として起動すると、ベクターグラフィックス風のウィンドウが開く。「START」をクリックするとプレイ開始だ。

画面中央の自機をカーソルキーで移動／回転させ、スペースキーで弾を発射する。周囲を漂う隕石を細かく砕いて消してしまえば、その面はクリアだ。なお、画面の左右端と上下端はそれぞれ繋がっている。

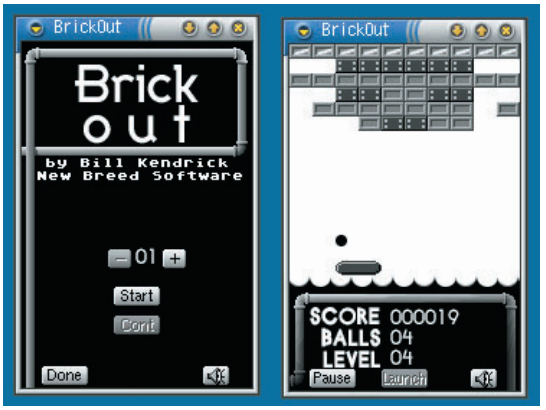


画面1　アステロイド風のシューティングゲーム「Agendaroids」

• Brick Out（画面2）

「brickout.host&」として起動すると、小型のウィンドウが開いてタイトル画面が表示される。開始面を選択してから[Start]ボタンをクリックするとプレイ開始だ。

画面下の[Launch]ボタンで玉が発射されるので、マウスまたは←／→キーでパドルを操作しよう。面が進むに連れて、玉を1回当てただけでは消せない堅いブロックなどのギミックが登場する。



画面2　正統的なブロック崩しゲームの「Brick Out」

• PDA Maze（画面3）

「pdamaze.host&」として起動すると、設定画面を兼ねたタイトルが表示される。右下の[Start]ボタンをクリックするとプレイ開始だ。

3D迷路上をマウスクリックするか、カーソルキーにより前後左右に移動すると、画面下部にマップが自動的に描かれる。迷路の出口まで到達することがゲームの目的だ。

画面3　3D迷路を歩いて唯一の出口を探索する「PDA Maze」

ドラッグ＆ドロップ対応のボタン型ランチャ

acidlaunch

バージョン：0.4　　ライセンス：GPL

http://linuxgamers.net/infoPage.php?page=acidlaunch

acidlaunchは、アプリ起動用のボタンが並んだランチャソフトだ。XMLベースの設定ファイルを編集して、ソフトの登録やボタン数の変更などを行う。複数のページをキー操作で切り替えて使うことも可能だ。また、ファイルマネージャやWebブラウザなどからのドラッグ＆ドロップに対応しており、GIMPやwgetなどと組み合わせて使うと威力を発揮する。実行にはGTK＋とlibxml2、gdk_pixbufが必要だ。

●ビルドとインストール

実行に必要なライブラリは、ほとんどのディストリビューションでは最初からインストールされている。ビルドにはそれぞれのdevelパッケージ（libxml2-develなど）も必要なので、忘れずにインストールしておこう。

acidlaunchは、ソース一式をtar＋gzipしたtarボールのみ配布されている。「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順でOKだ。

●ドラッグ＆ドロップにも対応

ktermなどのコマンドラインで「acidlaunch&」として起動すると、8個のボタンが並んだ小型のウィンドウが開く（**画面1左上**）。まずは、ウィンドウ上でAlt－ドラッグして、適当な位置まで移動させよう。

それぞれのボタンには、MozillaやGIMPなどのアプリが登録されている。たとえば、左列の上から3番目のボタンをクリックするとGIMPが起動する。また、NautilusやKonquererといったファイルマネージャから画像ファイルをこのボタンにドラッグ＆ドロップすると、自動的にそのファイルを読み込んで起動する。

なお、リンゴの形のアイコンは、設定ファイルで指定したアプリのアイコンが存在しない（あるいはパスが間違っている）ことを意味する。この場合は、ボタン上にカーソルを移動させ、チップヘルプの表示でアプリの名前を確認できる。

複数ページを持つ場合には、aキーとzキーで前後のページに切り替え可能だ。acidlaunchを終了するには、qキーを押せばいい。

●設定ファイルはXMLベース

一度acidlaunchを実行すると、ホームディレクトリにXMLベースの設定ファイル（.acidlaunch/config.xml）が作成される。登録されているアプリや表示されるボタンの数、アイコン用の画像ファイルなどを変更するには、この設定ファイルをテキストエディタで編集すればいい。

設定ファイルの冒頭には、ボタンを並べる方向（orientation）や、ボタンのサイズ（xsize、ysize）、ボタンの列（または行）数（width）などを設定する項目が記述されており、これらを変更することでウィンドウの形状を変えられる（**画面1右、下**）。

その後に、以下の形式でアプリ用のボタン設定が並んでいる。

```
<appicon>
  <name>アプリの名前</name>
  <icon>アイコンファイル</icon>
  <command>実際に実行されるコマンドライン</command>
</appicon>
```

たとえば、Mozillaのアイコンを正しく表示するには、14行目の記述を「<icon>/usr/lib/mozilla/icons/mozicon50.xpm</icon>」に修正する。また、日本語化されたwgetが文字化けしないように、51行目を「<command>rxvt -e wget</command>」と書き換えて、rxvt上でwgetが実行されるように修正したほうがいいだろう。

画面1　小型のボタンが並んだacidlaunchのコンパクトなウィンドウ。

サムネイル付きの軽量画像ビューア

xzgv

バージョン：0.7　　ライセンス：GPL

http://xzgv.browser.org/

xzgvは、サムネイル（縮小イメージ）付きの画像ビューアだ。JPEG／GIF／TIFF／PNGなど主要な画像形式に対応しており、サムネイルをxvやGIMPと共有できる。元の画像の縦横比を変えずに、ウィンドウに合わて画像を拡大／縮小表示できる。マウス操作はもちろんのこと、キーボードからでも、画像の切り替えや拡大／縮小、ファイル操作などを行える。動作にはGTK＋とImlibが必要だ。

●ビルドから起動まで

xzgvは、ファイル一式をtar＋gzipしたtarボールのみ配布されている。configureスクリプトは付属しておらず、設定ファイル（config.mk）を手動で書き換える方式だ。通常は設定ファイルを書き換える必要はなく、「make」でビルドし、「su」としてrootになってから「make install」としてインストールすればいい。

「xzgv&」として起動すると、ウィンドウが開いて、xzgvのロゴが表示される（**画面1**）。また、表示したい画像が置かれたディレクトリや、画像ファイル自体をコマンドラインで指定して起動することも可能だ。

●サムネイル作成と画像の表示

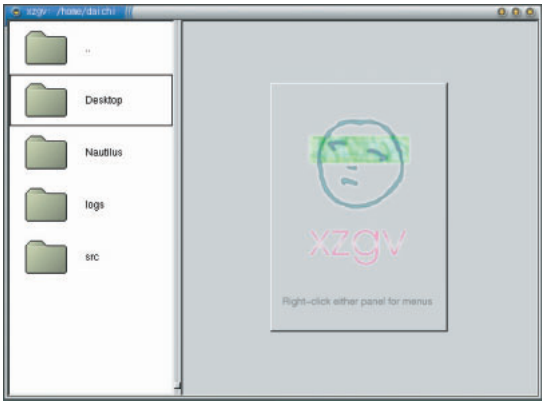
ウィンドウ左側の「セレクタ」には、カレントディレクトリのサブディレクトリと、表示可能な画像ファイルの一覧が表示される。

最初はどの画像も同じアイコンで表示されているが、uキーを押すか、セレクタ上での右クリックメニューから[Update Thumbnails]を選択すると、サムネイルが新規作成（または更新）され、各画像の概要を確認できるようになる。サブディレクトリ以下の画像ファイルのサムネイルを再帰的に作成することも可能だ。

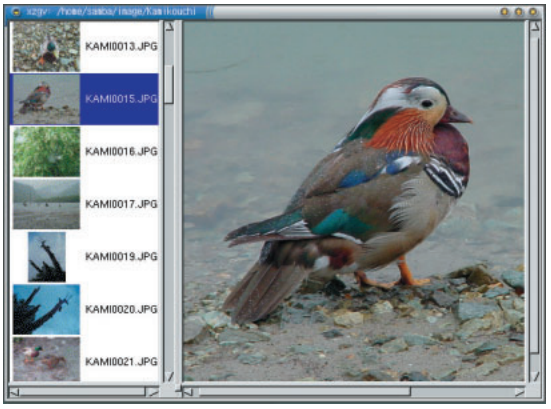
なお、サムネイル用の画像ファイルは、各ディレクトリの.xvpicsサブディレクトリに、画像ファイルと同名で作成される。これはxvやGIMPと同じ形式なので、いずれかのソフトでサムネイルを作成済みの場合は、最初からサムネイルが表示される。

セレクタに表示されたサムネイルをクリックすると、その画像が右側の「ビューア」に表示される（**画面2**）。あとは、SPACEキーを押すか、ビューア上で左ボタンをクリックするだけで次の画像に切り替わる。前の画像に戻るにはbキーを押せばいい。

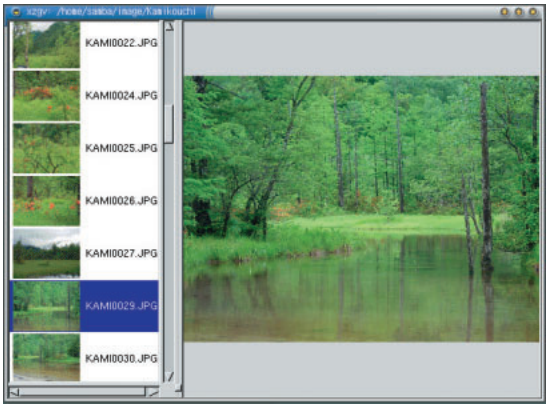
zキーを押すと、現在のビューアのサイズに合わせて、自動的に画像の拡大率が調整されるようになる（**画面3**）。画像の縦横比は変わらない。再度zキーを押すと元の状態に戻る。このほか、コピー／移動／削除などのファイル操作や、表示画像のガンマ値の変更、回転や反転なども可能だ。



画面1　起動時にはxzgvのロゴが表示される。サムネイルはまだない。



画面2　サムネイルをクリックすると、その画像がビューアに表示される。



画面3　ビューアのサイズに合わせて画像を拡大・縮小して表示できる。

隠喩としてのコンピュータ

かくして列強は結束した

文：豊福 剛
Text : Tsuyoshi Toyofuku
illustration : hmm

9月11日を境に、世界は変わり、新しい時代に突入したのだと言われている。従来の戦争が国家対国家の戦争であったのに対して、この新しい戦争は国家対テロ組織の戦争である。しかしながら、国家と非国家の戦争を、はたして戦争と呼ぶことができるのだろうか。

それをあえて新しい戦争と呼ぶのであれば、戦争の概念が変わるときには、その主体である国家の概念も同時に変わるのではないだろうか。また、平和という概念を戦争がない状態と考えるのであれば、平和の概念も変わるのではないだろうか。

20世紀の負の遺産

20世紀は戦争の世紀であった。アメリカとソビエトによる軍拡競争の果てに、いつか全面核戦争というカタストロフィに陥るのではないかという終末論的恐怖が世界を支配した。冷戦の時代は、かつて欧米列強の植民地であった第3世界が、国家として独立する歴史でもあった。そして、第3世界におけるアメリカとソビエトの代理戦争として展開された。

冷戦は、ソビエトの崩壊という、誰も予想しなかった幕切れとなった。イデオロギーによって東西に二分されていた時代は終り、政治における民主主義と経済における自由主義が普遍的な原理となったことが、歴史によって証明されたのだと考えられた。つまり、グローバリゼーションのはじまり、というわけである。

しかし、ソビエトの崩壊は、グローバリゼーションが勝利したという側面よりも、ソビエトそのものが破産しただけなのだと考えたほうが、実態に近いのではないだろうか。そして、ソビエトの破綻にともなう負の遺産として、たとえば核兵器や生物・化学兵器の拡散の危険性があるといえるだろう。

兵器というのは、それ自体で存在するものではなく、兵器の管理体制を含めたトータルな戦争技術体系の中に位置づけられるべきものである。その管理体制である国家が破綻したことによって、兵器に対する管理が部分的に不能になり、無政府状態におかれた兵器が存在すること、そのことが潜在的な危機なのである。

兵器自体のポータビリティが、さらに潜在的な危機を増大させている。スーツケースで持ち歩ける小型核爆弾。そ

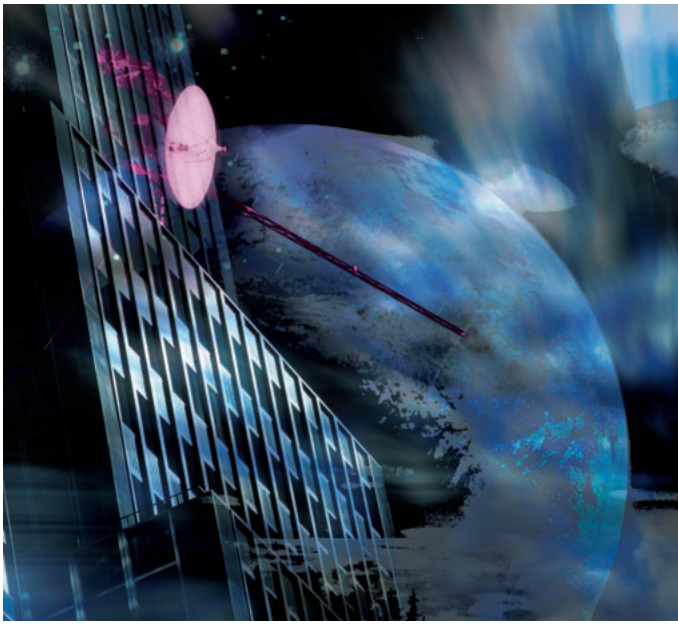
して、生物兵器による攻撃は、炭疽菌による郵便的テロとして、現実のものとなった。グローバリゼーションを支える交通、輸送、通信のインフラに忍び込んだ恐怖がウイルスのように伝播していく。見えないテロ、ミクロのテロが、戦争状態と平和な状態の境界を曖昧にする。ポータブルな兵器が、戦争における前線をなくしてしまい、いまここの日常を潜在的な戦場に変えてしまった。メール爆弾が、サイバースペースではなく、リアルワールドにおいて現実のものとなってしまったのだ。

冷戦後の危機のもうひとつの側面は、それまでの強権的な支配体制によって表面化しなかった民族対立の問題が、一気に噴出していることにある。それまで秩序を維持していた上からの権力がなくなったとき、その権力の空白地帯に人々は投げ出され、ホッブスのいう「万人の万人に対する闘争」という自然状態に置かれてしまう。そのとき、それまでの上からの権力に代わって、求心力をもつ装置として、民族の記憶や宗教が呼び戻されるのだ。そして、民族の違い、宗教の違いが、終わりのない憎しみと暴力の連鎖をもたらしてしまう。

そのような不寛容や排他性は、貧困、富の格差、社会における公平さの欠如と深く関係しているのだろう。民族や宗教の違いによって支配階級であるか被支配階級であるかが決まる社会においては、闘争がなくなることはない。そして、グローバリゼーション自体は、そうした富の格差を直接的には解決しないのではないか。むしろ、社会が民主化されていない状態のまま、グローバリゼーションに組み込まれてしまうと、利権と腐敗の温床になってしまうのではないか。新興のマフィア系資本家が実権を握っていたエリツィン政権末期のロシアが、まさにそうだったのではないか。

中東はグローバリゼーションの外部なのか？

20世紀は石油の世紀でもあった。そして、先進国にとって中東とは、東西冷戦における地政学上の重要拠点であるとともに、良質で安価な石油の供給地であり、そこでの覇権を確立することが世界戦略の大きな柱であった。しかし、中東の国際関係が安定的であったためしはなかった。軍事と石油の2つの点で、中東は常に戦火の絶えない、クリティカルゾーンであった。



中東の産油国の多くは、国内の政治体制においては、首長が支配する前近代的な国である。たとえばクウェート、サウジアラビア、アラブ首長国連邦がそうだ。それに対して、イラクやリビアのようなイスラムの中では世俗的な国のほうが、他のイスラム国家よりも民主的なのである。

1979年のイラン革命は、それ以前の王制を倒し、イスラム原理主義による政治体制を樹立した。それは政治体制の点からみれば、それ以前よりも民主的で公正な社会なのである。そして、1980年、イラン革命が飛び火するのを恐れたイラクは、イランへの攻撃を開始した。イランはイラクのような世俗化した国家はイスラム国家とは認めず、イラク打倒を呼びかけた。そして、戦況がイラン優勢になると、アラブ、アメリカ、ソビエトはイラクのフセイン政権を支援した。孤立化したイランは、1988年に国連決議である停戦を受け入れた。

そのイラクは、1990年にクウェートに侵攻した。その原因は、イラン・イラク戦争に費やした軍事費による財政圧迫や、クウェートの石油の生産過剰によるイラン経済の圧迫などがあるといわれている。国連安全保障委員会はイラクに対してクウェートからの無条件撤退を求めたが、イラクはこれを拒否したため、1991年アメリカを中心とする多国籍軍との湾岸戦争に突入した。

湾岸戦争のとき、これは聖戦（ジハード）であるとフセインが呼びかける声明をテレビ放送したような記憶がある。聖戦を呼びかけ、アメリカとイスラエルを敵と位置づけている点では、ビンラディンとフセインは共通点が多いように思えるが、イラクに侵攻されたクウェートに対して、ビンラディンは軍事的支援を申し出たという話があるので、実際のところ、この二人を結ぶ線があるのかどうか、よくわからない。

ひと括りにイスラムといっても、宗派と民族が複雑に入り組んでいるわけで、その点ではキリスト教文化圏であるヨーロッパ内部において戦争が繰り返されてきたことと似ているように思える。それはともかく、イスラムが独自の近代化を達成できず、石油だけに依存した経済構造のままであれば、イスラムに未来はないだろう。

文明の衝突の議論はともかくとして、先進国（日本を含む）にとって中東の産油国は、自分たちに都合のいい政権であってほしいというのが本音であるのだ。そして、中東情勢が不安定であるからこそ、カスピ海周辺の石油および

天然ガスの確保が、戦略的な重要性をもつのである。

ちなみに、地球温暖化防止の京都議定書に加わらないアメリカの利害というのは、ブッシュ政権と石油資本の深い関係を補助線にすると、わかりやすいように思う。しかし、20世紀的な資源大量消費型の経済成長を続けることにも、やはり未来はないだろう。

世界新秩序とは列強支配なのか？

アフガニスタンに対するアメリカの攻撃は、10月下旬の現在、空爆から地上戦へと新たな段階に移行しつつあるのだが、アメリカの戦略は、タリバン政権の打倒と、その後のアフガニスタンにおける新しい政権の模索へと展開しつつある。その意味では、タリバン政権という国家との古い戦争に強引に持ち込んだといえるだろう。テロ集団との戦争といいつつ、実質的にはタリバン政権に対する戦争であるのだから、アフガニスタンの民間人を敵としているのではない、というのは詭弁でしかないだろう。

アメリカのアフガニスタンに対する軍事行動は、パキスタンやインドネシア、マレーシアの国民の間で反米感情を増幅させる結果を招いている。仮にアフガニスタン新政権が樹立できたとしても、もともと内戦状態にあった民族が、そう簡単に復興をめざして協力できるとは思えない。イランやパキスタンの思惑も複雑にからみあっている以上、戦争そのものよりも、その後の安定した秩序を維持することのほうが、より複雑で困難な課題であるだろう。

そうした不安定要因を抱えながらも、アメリカとイギリスの外交は、ロシアと中国とインドの協力を取り付けた。報復も、大国が支持していれば文句ないだろうという強者の論理というわけだ。この強者の論理が、アメリカだけでなくロシアと中国のイスラム系少数民族に対する歯止めのない抑圧につながる可能性は否定できない。テロとの戦いという口実さえあれば、アメリカと同じように、ロシアや中国も、ある国に対して武力を行使し、自らに都合のいい政権を樹立してもいいという前例になってしまうのではないだろうか。

新しい戦争とは、つまりは、そのようなことではないのか。そして、大国の行き過ぎた暴力に対して、冷静に歯止めをかけることができなければ、怒りと憎悪の連鎖が根絶されることはないだろう。

Profile

とよふく つよし

1962年東京生まれ。メディアデザイン研究所技術顧問。訳書に『Javaプログラムクイックリファレンス』『Java分散コンピューティング』（オライリージャパン）『GIMPパーフェクトガイド』（エムディーエヌコーポレーション）などがある。

テロリストと知的所有権

文：安田幸弘
Text : Yukihiro Yasuda

アメリカの独善

アメリカは何となく嫌いだという人は少くない。アメリカ的独善というのか、アメリカはいつもいかにも正義の味方のような顔をしてズケズケと他国に干渉し、アメリカに対する批判には知らんぷりをし続ける。

このあいだ、ある団体のサーバの設定をするためにやってきたあるアメリカ人が、「それはねえ、アメリカ人って、外の世界のことを知らないからなんだよ」と言っていた。彼に言わせれば、アメリカ人は「アラブのこともアジアのこともアフリカのことも知らない。アメリカ人は、アメリカが第三世界で何をやっているのかを知ろうとせず、貧しい第三世界の人々だって頑張ればチャンスはあるのにね、なんてぐらいにしか考えていないんだ」という。そうなのかもしれない。ソフトウェアをいじる人ならわかるだろうが、アメリカ主導でゴリゴリ決められたUNICODEだとか、国際化のことなんて全然考えていないソースコードを見ながら「ブッ殺してやる……」なんてつぶやいた経験を持っている日本のハッカーは少くないんじゃないだろうか。

それはともかく、特にソ連が崩壊して世界唯一のスーパーパワーとなって以来、アメリカは完全に独善に陥ってしまった。たとえば国際的な合意だった温暖化条約の京都議定書を「国益に反する」と蹴とばし、国連人権委員会でのイスラエル批判を不満として席を立ち、国際的な批判ももものともせずに核実験やミサイル防衛構想を強引に押し進め、第三世界の国々には、あまりにも過酷なアメリカン・スタンダードを「正義」の名前で強要し続けてきた。

テロリストたちが例の悲惨な暴挙に出た裏には、そんなアメリカに対する絶望的な反感があることをアメリカ人は知らないのだ。

もちろん、それでもあのニューヨークでのテロは弁護の余地がないほど許しがたい暴力だし、ビンラディンがいくら「アメリカがアラブから出ていかなければテロは続くだろう」なんて言ったって、アメリカが「はいそうですか」とアラブから手を引くわけもない。

アメリカのジレンマ

そんなわけで保安官気取りの大統領は、いつもの調子で独善的にテロとの闘いだの、テロを容認する国家への攻撃だのと、勇ましいことを言い始めたのだけれど、世界中に散らばるテロリストを根絶やしにするにはアメリカが一人で頑張ってもどうなるものでもない。テロリストの息の根を止めるには、それこそイスラムや第三世界諸国を含めた国際的な協力が必要になるわけだ。これまでのように、強引なパレスチナ政策や、傲慢な第三世界への対応を続けていれば、国際的な協力なんてできるわけがない。アメリカは、アラブや第三世界の人々の声に、これまで以上に真摯に耳を傾けなければならなくなったわけだ。

つまり、アメリカはテロリストを撲滅するために、テロリストを生んだアメリカの独善を反省せざるを得なくなってしまったわけだが、これはアメリカにとっては何とも皮肉な結果といえるかもしれない。アメリカ人に第三世界の怒りをつきつけたあの破滅的な暴力がアメリカの独善的な外交政策を変えさせたということになれば、テロにも一定の効果があったこと、テロリストたちの無言のメッセージに含まれる抗議を自ら認めたことになりはしないだろうか。

もちろん、われわれの身の回りには、似たようなことはいくらでもある。世界化に反対する連中は世界化のシンボルのようなインターネットを使って連絡し合い、アメリカ帝国主義に反対する第三世界のナショナリストたちが英語で話をしていたりする。世の中って、そんなもんだ。

しかし、やっぱりジレンマはジレンマ、今回の事件をきっかけとして、これまでアメリカの市場原理主義者が国益をかけて第三世界に押し付けてきたいろんな不条理をアメリカは自覚せざるを得なくなっ

たケースが、炭疽病治療薬をめぐる特許権問題だ。

知的所有権と命

特許権などを含む、いわゆる知的所有権というやつがクローズアップされるようになったのは、世界貿易協定の枠組みに組み込まれたことによる。つまり、従来のように形のある物だけでなく、ソフトウェアのような形のない知的生産物もまた、モノと同じように貿易協定の中で取り扱うようにしようということになったわけだ。この背後にはモノ作りから知識産業へのアメリカの産業構造の転換があり、知識産業の収益を確保する必要があった。だからこそ、アメリカをはじめとする先進国の国益を守るために、多くの第三世界の異論を押しきって、知的所有権に関する条文が協定の内容につっこまれたわけだ。

しかしそれによって、第三世界の国々は何をするにも先進国に何らかのライセンス料を支払わなければならなくなってしまった。工業製品の生産はいうまでもなく、農業の分野でさえ種苗に対する特許権や農薬に対する特許権への支払いが必要になる。日常生活でさえ、たとえばその土地の住民が古くから使ってきた薬草の薬理効果に特許権が設定されたおかげで、先祖代々利用してきた薬草にまでライセンスが発生してしまう。新しい知的所有権を開発しようとしても、さまざまな基本特許はすでに先進国に押さえられてしまっているのだから、将来の巻き返しも難かしい。日本や東南アジアの新興工業国のように、先進国の工業製品のコピーで力をつけて欧米に競争を挑むということもできなくなってしまう。

もちろんすべての知的所有権が問題だと言うわけではないだろうし、現実的な意味のある知識の開発や生産だってあるかもしれない。新しい品種を使うことで農作物の収量が上がれば、新品種のライセンス料金を支払っても帳尻が合うのかもしれない。生きるか死ぬかの瀬戸際にいる人は、高価な新薬に喜んでお金を払うだろう。ただしそのお金があれば、の話。特に第三世界で深刻な問題になっているのが、あまりにも高額な新薬の価格である。

代表的なケースがエイズの治療薬で、国によっては成人の8割がエイズに感染しているアフリカでは、特許により販売価格が維持されている高価な抗HIV薬を買えない貧しいAIDS患者が毎日1万人近く死んでいるという。人道上の問題であるとともに、国家存亡にかかわる非常事態だとして、発展途上国が治療薬の特許に関する例外規定を適用するように訴えているのだが、欧米の製薬会社は簡単にはウンと言わない。そしてアメリカや日本は、そんな製薬会社側の言い分をバックアップし続けてきた。

ところが、アメリカで炭疽菌によるテロの疑いが強くなると、アメリカは緊急事態だとして炭疽菌治療薬のコピーを認めろと言い始めたのである。死の恐怖や国家崩壊の恐れに直面した国にとって、知的所有権よりも大切なものがあるということを、ようやくアメリカ人は学んだわけだ。

もちろん、国際社会はイジワルなんかせずにアメリカの要求を認めてやるべきだろうし、これまで知的所有権についてアメリカと同調し続けてきた日本も、総理大臣が約束した「最大限の貢献」を根拠に、国家的な危機下でのコピー薬の承認を認めるように国際社会に働きかけるべきだろう。何も旗を見せるだけが対米協力じゃない。そして言うまでもないが、アメリカを初めとする市場原理主義者たちは、知的所有権の無理な強制や知的所有権そのものが抱える多くの問題点を直視してほしい。そうすれば、テロの温床となっている世界中の貧困や病、政治的な不正義も見えてくる。

リチャード・ストールマンが言うように、知識は人類の共有財産だ。これは何もソースコードだけの話じゃない。

Profile

やすだ ゆきひろ

生業はテクニカルライター。原稿書きのかたわら、(株)市民電子情報網のボランティア社長兼技術スタッフとして、NGO向けプロバイダのネットワーク運営に携わる。

正しきタコへの道

初級Linuxer養成講座

Linuxを時計代わりに使う～(1)

現代人にとって「時計」はなくてはならない道具だ。現在の時刻を知らせたり、指定した時刻にアラームを鳴らして、約束の時間に遅れないようにしたりする。今回はLinuxのコンソールを「アラーム時計」として使う方法をいくつか紹介しよう。

文：竹田善太郎
Text : Zentaro Takeda

生来、**整理整頓が苦手**なので、自宅の作業用机の上はいつも雑然としている。17インチのディスプレイを置いても十分な作業スペースがとれるだけの大きめのテーブルを使っているのだが、キーボードにトラックボール、ノートPC 2台を置いたうえに、さまざまな資料や小物が散らかっているので、ちょっとした書き物をするスペースに困ることもある。余計なものを片づけてしまえばよいのは重々承知しているのだが、こればかりは**性分**で、なかなか実行できない。

同じような悩みを持つ人は少なくないようで、電器店のパソコン売り場にいくと、ノートPCやキーボードをディスプレイの下にしまったり、テーブルに取り付けるキーボードやマウス用の**拡張テーブル**のようなものが数多く売られている。しかし、これらの製品はプラスチック製で**頼りなげ**なものが多く、木製や金属製のしっかりしたものは価格が高い。なにより、ただでさえ狭い部屋なのに、テーブルの横や裏側にものを取り付けると、動き回るたびに**膝をぶつけて**しまいそうだ。

ところがある日、便利そうなものを見つけた。フライトシミュレータ関連のWebページからたどり着いた航空関連用品専門のオンラインショップに、パイロット用の**ニーボード**と呼ばれるものがあったのだ（**画面1**）。ニーボードとは、膝（knee）にゴムバンドで取り付けることのできる、小さめのクリップボード（書類ばさみ）のようなものである。

大型の旅客機であっても、飛行機の操縦席はとても狭い。そんな中でも、パイロットは操縦をしつつ、航空地図を広げたり、管制官との無線交信の内容をメモしたりする必要があるのだが、このようなときに、膝に取り付けたニーボードを**テーブル**のように使うらしい。

件のショップでは、安いものから高いものまで、何種類ものニーボードを扱っていたのだが、ナイロン製のカバーにアルミ製のクリップボードが内蔵されている製品がかなり手頃な価格（27ドル弱）で売られていたので、試しに購入してみた。航空用地図の出版元として有名な**ジェプセン社**の「VFR/IFR Kneeboard」という製品である。ちなみに、「VFR」は「有視界飛行」、「IFR」は「計器飛行」を意味する航空用語だ。

連絡のちょっとした行き違いなどがあって、商品が届くまでに1カ月以上もかかってしまったのだが、届いたものは予想通りの品物で、さっそく使ってみた。クリップボードのサイズは、ちょうど「リーガルサイズ」のメモ用紙が収まる大きさで、ナイロン製カバーにメモ帳を直接挟んで使えるようになっている。ペンホルダーや電卓（あるいはPalmデバイス）を入れられるポケットもついている（**写真1**）。膝に固定するゴムバンド（面ファスナーつき）は、幅広のしっかりしたもので、ニーボードを装着したままで部屋の中を歩き回るのにも支障はない。また、ナイロンカバーを閉じれば、そのまま持ち歩くのにも都合がよく、鞄の中の小物を整理するのにも役立ちそうだ。

自宅の机で作業するときはもちろん、**喫茶店の狭いテーブル**にノートPCを広げて仕事をしたり、講演を取材したりするときなど重宝している。人目が気になるのでやったことはないが、**電車の中で書き物**をしたいときにも使えそうだ。**パソコン用品とカー用品**は、それぞれの専門店以外で探したほうが、安くて良いものが見つ

かる可能性が高い気がする。

時刻を表示するコマンド

PDAからPC、果てはスーパーコンピュータに至るまで、ありとあらゆるコンピュータは、当然のように時計としての機能を持っている。スケジュール管理に使ったり、あるいはシステムの状態を監視して記録したりする際に、現在時刻を知る必要があるからだ。PCの画面上に時刻を表示するアプリケーションは、数えきれないほど多く存在する。Microsoft WindowsやX Window SystemなどのGUIシステムでは、それ自体に時計を表示する機能やアプリケーションが付属している（**画面2**）。

もっとも、高価なサーバ用コンピュータは別として、一般的なPCが内蔵している時計機能（「リアルタイムクロック」と呼ばれる機能）は、あまり正確ではない。筆者宅にあるPCの時計機能は、1日に十数秒近くも進んだり遅れたりする。「月差」でいえば数分も狂うことになり、一昔前の機械式腕時計よりも精度は低い。それでも、仕事をしているコンピュータ画面上に現在時刻が表示されるのは何かと便利である。

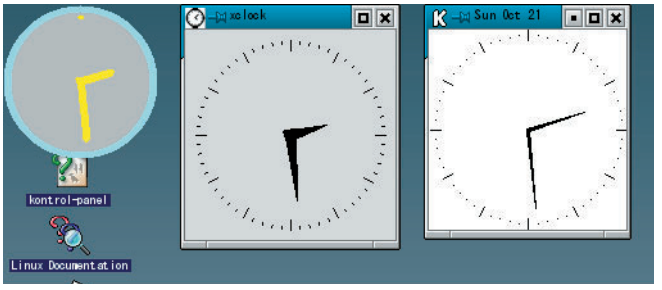
時刻を知るのなら、腕時計を1つ持っていれば十分なはずなのだが、現代社会では、ありとあらゆるところに時計を置いて、つねに時間を気にしながら生活するのが当たり前のようになっている。これが良いことなのか悪いことなのかは別として、現在時刻がすぐにわかるようになっていたほうが安心できるという人は多いようなので、Linuxマシンで時刻を表示する方法をいくつか紹介しよう。

基本の「date」コマンド

コマンドラインで時刻を表示するもっとも基本的な方法はdateコマン

写真1　「Jeppesen」のロゴが入ったニーボード「VFR/IFR Kneeboard」
左側のポケットには電卓やPDA、真ん中のクリップボード部分にメモ用紙や地図、右側のポケットには地図や書類を挟んでおくことができる。アルミ製のクリップボードには、航空機を運行する際に必要となるさまざまな規則や計算方法などが印刷されており、これだけを取り出して使うこともできる。

画面1　「本物の」パイロット用品を販売するWebサイト「AVShop」(http://www.avshop.com)
画面のニーボード以外にも、GPS受信機、航空用計算機、ヘッドセット、フライトプラン作成用ソフトウェアなど、さまざまな用品を扱っていて、興味のある者には飽きさせない品揃えである。価格も想像以上に手頃だ。



画面2　X Window Systemの時計のいろいろ
Linuxをインストールしただけでも数種類の「時計」が使えるようになっている。左は、丸いウィンドウで表示できる「oclock」、中央はX Window Systemの標準アプリ「xclock」、右側はxclockを拡張して、アラーム機能などを備えた「rclock」。このほかにも、星の数ほどの「時計」がフリーソフトウェアとして利用できる。

ドである。とりあえず、dateコマンドになんのオプションもつけずに実行してみよう（**画面3**）。

```
$ date
Sun Oct 14 09:55:12 JST 2001
```

このように、現在の曜日、日付、時刻、西暦などが表示される。時刻のうしろにあるJSTという文字は、この時刻が日本標準時間（Japan Standard Time）であることを意味している。Linuxをインストールしたときの設定によっては、「GMT」（Greenwitch Mean Time：世界標準時のこと）などと表示されることもある。この設定を変更する方法については、次回に説明することにする。

dateコマンドで表示されるのは、コマンドを実行した時点での時刻である。画面上に現在時刻を継続的に表示するものではないが、コマンドライン上で作業をしている最中に時刻や日付を知りたくなったら、**気軽に**使えるコマンドなので、覚えておくと便利だ。

■timeコマンドはないのか？

ところで、「date」というのは「日付」という意味なので、**時刻だけを表示する「time」というコマンドがあるのでは？**と思うかもしれない。MS-DOSには「dateコマンド」と「timeコマンド」の両方が存在して、それぞれ日付と時刻を表示したり設定したりできるようになっている。試しに、Linuxのコマンドラインで「time」を実行すると、次のようなエラーメッセージが表示される。

```
$ time
bash: syntax error near unexpected
token `time'
```

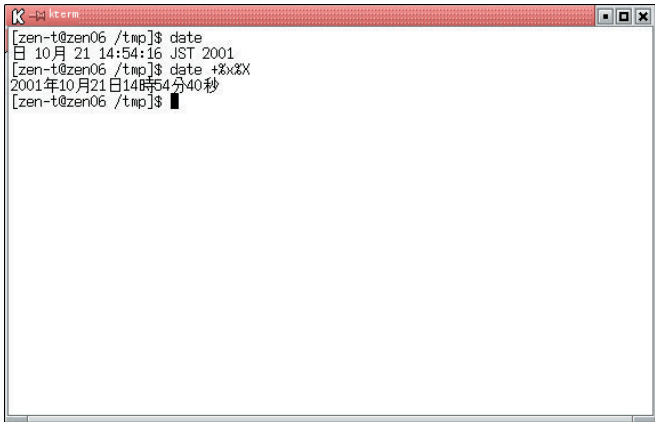
実は、timeというのはbashの内部コマンドになっていて、**プログラムの実行にかかった時間**などを測定するときに使うものである。一般ユーザーにはあまり使い道のないコマンドだが、**PCの性能を測定**するときに利用できるので、覚えておいても損はないかもしれない。timeコマンドは次のように使う。

```
$ time cp test00.avi /tmp
```

上のコマンドラインを実行すると、「cp test00.avi /tmp」（test00.aviというファイルを/tmpディレクトリにコピーする）というコマンドが実行され、終了までにかかった実時間、およびCPUがこのコマンドを処理するのに費やした内部時間などが表示される。

```
real  0m12.518s
user  0m0.020s
sys   0m0.150s
```

上の例では、コマンドが終了するまでの実時間が約12.5秒であることを示している。これに続く「user」と「sys」がCPUが費やした時間を示しているが、この数値の意味についてはこ



画面3　コマンドラインでdateコマンドを実行してみる
この画面はTurbolinux 7.0での例。日本語対応のコンソールでは、曜日や日付が日本語で表示されるが、「10月21日　日曜日」としてほしいところが「日　10月21」となってしまうのが残念。ちなみに、「date +%x%X」としてやれば、画面のようにもう少しまともな形式で表示される。

時刻	
%H	時（24時間制、2桁固定）
%I	時（12時間制、2桁固定）
%k	時（24時間制、可変桁）
%l	時（12時間制、可変桁）
%M	分（2桁固定）
%p	午前・午後（AM/PMなど）
%r	12時間制の時分秒（hh:mm:ss AM/PM）
%S	秒（2桁固定）
%T	24時間制の時分秒（hh:mm:ss）
%X	現在の言語設定に合わせた時刻
%Z	タイムゾーン名（GMT、JSTなど）
日付	
%a	曜日名（省略形）
%A	曜日名（完全表記）
%b	月名（省略形）
%B	月名（完全表記）
%c	日付と時刻
%d	月内通算日数
%D	日付（mm/dd/yy）
%h	（%bと同じ）
%j	年内通算日数
%m	月（2桁数値）
%U	日曜日を週の最初の日とみなした場合の年内通算週
%w	曜日（0を日曜日とした1桁の数値）
%x	現在の言語設定に合わせた日付
%y	西暦の下2桁
%Y	西暦（4桁）

表1　dateコマンドで設定できる表示項目（抜粋）

こでは説明しない。

■dateコマンドのオプション

本題に戻るが、dateコマンドでは、オプションをつけることで、日付や時刻をさまざまな形式で表示することができる。日付や時刻の形式は、「＋」（プラス記号）に続けて、**表1**にあげたような表示項目を並べて指定する。たとえば、何時何分かだけを表示させたい場合には、次のようにオプションを指定する。

```
$ date +%I:%M
11:05
```

「%I」や「%M」が表示する項目を指定する部分で、項目を指定する以外の文字（上の例では「：」）はそのまま表示されるようになる。この機能を応用すると、たとえば現在が今年の第何週なのか、あるいは1月1日から何日経過しているのか、などを簡単に知ることができる。

```
$ date +%W
41

$ date +%j
287
```

「%W」は、月曜日を週の最初の日とした場合の年内の通算の週、「%j」は年内の通算日数を指定するオプションである。出力結果は、現在は今年の41週目で、1月1日から数えて287日目にあたるということを示している。

また、現在日時だけでなく、指定した日時についての情報を表示させることもできる。日時の指定には-dオプションを使う。たとえば2001年1月1日について知るには次のようにする。

```
$ date -d 2001/1/1 +%W
01
$ date -d 2001/1/1 +%j
001
```

シェルプロンプトに時刻を表示する

X Window SystemなどのGUI環境でLinuxを使っている場合なら、デスクトップ上に時刻を表示する方法はいろいろあるが、別のマシンからtelnetなどでログインしているような場合には、前述のdateコマンドを使わないと現在時刻は表示できない。**dateコマンドをいちいち起動するのは面倒**という場合には、シェルプロンプトに時刻を表示させてしまう方法もある。使っているシェルの種類によって、この方法は多少変わるのだが、bashやtcshなど、現在使われている主なシェルには、プロンプト部分に時刻を表示する機能が含まれている。ここではLinuxの標準シェルである、bashの場合について説明しよう。

bashでは、PS1という環境変数の値を変更することで、プロンプト部分に表示される内容を自由に変更できるようになっている。Linuxのディストリビューションによって多少変わるが、インストールしたままの状態では、**画面4**のようなプロンプトが表示されるような設定になっている。試しに、これを別の文字列に変えてみよう。

```
$ export PS1=hello

hello
```

画面上には「hello」という文字が表示されるだけで、一見すると何も起こらないように見える。しかし、実は**この「hello」という部分がコマンドラインのプロンプト**になっていて、「hello」に続けてコマンドを入力できるようになっている（**画面5**）。

```
hellols
test.txt test00.avi
hello
```

画面4　通常のコマンドプロンプト（Turbolinux 7.0の場合）
左から、ユーザー名、ホスト名、カレントディレクトリ名が表示されるようになっている。

画面5　プロンプトを変えてみる
プロンプトをデフォルトのものから「hello」に変えてみる。プロンプトとコマンドの間にスペースが入っていないので、とても見づらいが、プロンプトが変わっているのがわかるだろうか。

先ほど設定した内容では、プロンプト部分（hello）の後ろに空白部分がないので、プロンプトとコマンドがくっついてしまって見づらくなっている。たとえば、次のように設定すれば、多少は見やすくなる。

```
$ export PS1="hello "
```

このPS1という環境変数には、単なる文字列だけでなく、ホスト名、ディレクトリ名、ユーザー名、時刻などのさまざまな情報を表示するための指示を追加できるようになっている。設定できる指示の一覧を**表2**にまとめておこう。

これを利用して、プロンプト部分に時刻を表示するには、たとえば次のように設定すればよい。

```
$ export PS1="[\t] "
```

すると、プロンプトは次のようになる。

```
[11:34:26]
```

コマンドを実行したり、リターンキーを押してプロンプトを再表示させたりするたびに、現在時刻が表示されるようになる（**画面6**）。

余談になるが、このプロンプトの設定を駆使すると、プロンプトをカラーで表示させたり、複数の行にまたがるような複雑なプロンプトを作ることもできる。

またしてもLinuxの話題からは外れてしまうが、MS-DOSにはプロンプトを自由に変更できるようにするツールがいくつかあって、プロンプト部分に**キャラクタ文字で描いた絵**（今風にいえば「アスキーアート」）を表示している人もいた。中には、画面の半分以上を埋めてしまうような**巨大なプロンプト**を作ったりするなど、かなりエスカレートしたこともあったが、肝心のコマンドラインや実行結果を表示する部分が狭くなってしまうので、プロンプトに凝るのは**ほどほど**にしておいたほうがよいだろう。

\a	端末のビープ音を鳴らす
\d	日付
\e	エスケープ文字
\H	ホスト名（ドメイン名を含む）
\h	ホスト名（ドメイン名を含まない）
\n	改行文字
\s	シェルの名前（sh、bashなど）
\T	12時間制での現在時刻
\t	24時間制での現在時刻
\@	12時間制での現在時刻（am/pm表記付き）
\u	ユーザー名
\w	カレントディレクトリ（パス付き）
\W	カレントディレクトリ（パス省略）

表2　プロンプトに指定できる項目一覧

指定時刻になったら知らせる

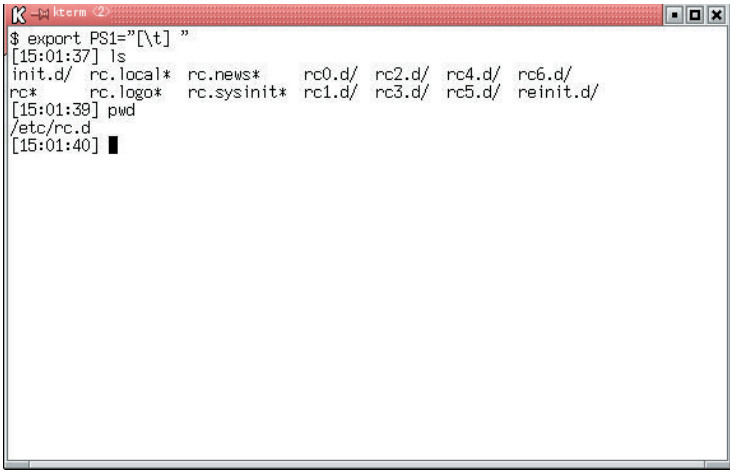
現在時刻を表示できるようになったら、**アラーム機能**も欲しくなるのが人情というものだろう。**カップ麺を調理（？）**するときなど、指定時間が経過したら知らせるタイマー機能や、会議や外出などの予定時刻がきたら知らせるアラーム機能が使えると便利だろう。

もちろん、これらの機能を備えている、スケジュール管理アプリケーションを使えば、わざわざコマンドラインでアラーム機能を使う必要もないのだが、ちょっとした用事だけでわざわざアプリケーションを起動するのは面倒なこともある。また、アラーム機能はともかく、タイマー機能を持っていないスケジュール管理ソフトは、意外と多い。Linuxのコマンドラインを使う時間が長いユーザーなら、コマンドラインからアラームやタイマーを使えれば便利に思うはずだ。

■sleepコマンド

まず、指定時間が経過したら知らせてくれる、**タイマー機能**を実現する方法を説明しよう。のっけから失望させてしまうようだが、Linuxでは標準的に使える**タイマーコマンドのようなものは存在しない**。オンラインマニュアルで「timer」という単語を検索してみても、初心者にはわけのわからない、プログラミング用の情報しか表示されない。その代わり、Linuxでは、**指定した時間だけなにもしないで待つ**というコマンドがある。その名もずばり、**sleep**（休眠）という名前のコマンドで、このコマンドを実行すると、指定された時間が経過するまで、なにもしないで休眠状態になる。指定時間が経過するとコマンドは終了する。

画面6　プロンプトに現在時刻を表示するようにしてみる　周りをかっこ（[]）で囲んで、見やすくする。最後に空白文字を入れておけば、プロンプトとコマンドがくっついてしまうこともなくなる。



このコマンドを利用すれば、コマンドラインでタイマー機能を実現できる。

```
$ sleep 180; echo "Time to eat."
```

上のコマンドラインを実行すると、180秒（3分）後に「Time to eat.」という文字列が表示される。

ただし、これでは3分間、コマンドラインが使えなくなってしまうので、**バックグラウンドで実行**するようにしてみよう。

```
$ (sleep 180; echo "Time to eat") &
```

ここで、コマンドライン全体を()で囲んでから、最後に「&」（バックグラウンド実行を指定する文字）を追加している点に注意すること。

echoコマンドでは、画面上に文字列が表示されるだけなので、忙しいときなどはメッセージに気がつかないかもしれない。そのようなときには、前回説明した**writeコマンド**を使えば、ビープ音も鳴らしてくれるので、少しは気がつきやすくなるだろう。writeコマンドを使う場合は次のように入力する。

```
$ (sleep 180; echo "Time to Lunch"
| write user1) &
```

「user1」の部分には、自分自身のユーザーIDを入力すればよい。指定時間が経過すると、**画面7**のように、メッセージが表示されると同時に、ビープ音が鳴る。

■atコマンド

目覚まし時計のように、指定時刻になったら知らせてくれる**アラーム機能**は、atというコマンドを使えば実現できる。

```
$ echo "echo time to lunch" | at
13:00
```

atコマンドの使い方はちょっと癖があって、指定時刻に実行させたいコマンドラインは、atコマンドのオプションとして与えるのではなくて、**標準入力に与える**ようになっている。上の例では、"echo time to lunch"という文字列をatコマンドの標準入力に与えている。つまり、指定時刻（13:00）になったら、「echo time to lunch」というコマンドが実行されるようになるわけだ。

さきほどのsleepコマンドと同様に、echoコマンドではなくwriteコマンドでアラームメッセージを表示させるようにすることもできる。

```
$ at 13:00 <<EOF
> echo "Time to Lunch" | write
user1
> EOF
```

この例では、「ヒアドキュメント」を使っているので、ますます複雑なように見えるが、要するに、「at 13:00」というコマンドの標準入力に、「echo "Time to Lunch" | write user1」という文字列を与えている。つまり、13:00になったら「echo "Time to Lunch" | write user1」というコマンドラインが実行され、user1に「Time to Lunch」というメッセージが送られるようになるわけだ。

atコマンドで設定した実行予定の一覧は、**atqコマンド**を実行すると次のように表示される。

```
$ atq
1       2001-10-21 13:00 a
3       2001-10-22 12:48 a
```

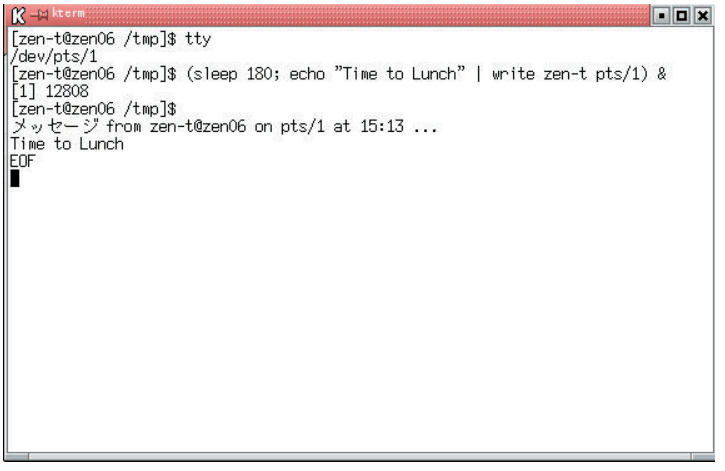
設定を取り消すには、atrmコマンドを使えばよい。そのとき、取り消す予定はatqコマンドで表示される一覧の一番左側の番号を使う。

```
$ atrm 3
```

上のコマンドを実行すると、3番のジョブ（2001-10-22 12:48のジョブ）が取り消される。

●

次回は、atコマンドのもう少し詳しい使い方と、Linuxの内部時計の設定方法（時刻の合わせ方）などについて説明したい。



画面7　sleepコマンドを使って、指定時間後にwriteコマンドでメッセージを送る
必要に応じて「ttyコマンド」を使い、メッセージを表示するウィンドウ（コンソール）を指定する。writeコマンドの詳しい使い方については、前号を参照してほしい。

初等 Ruby 講座

プログラムはコンピュータに対する手順書だと考えることができます。今回はその手順書の「読み進め方」に当たる手順の制御方法について学びます。かなり基本的な部分ですので、プログラムの知識・経験のある方は雑学の部分だけを楽しんでください。

第2回　条件判断とループ

文: まつもと ゆきひろ
Text : Yukihito "Matz" Matsumoto

最初に白状しておきますが、私は数学がダメです。昔から全然ダメなんです。よくコンピュータは理系的と言われます。私は学校は理系にいましたが（コンピュータがどうしてもやりたかったので）、頭の中身は文系でした。大学時代は周りのみんなについていけずに悩んだものでした。

なぜわざわざこんなことを言うかというと、今回の記事中に何度か「証明」という単語が（数学的な意味で）出てくるからです。もちろん私になにかを証明することができるはずもなく、これは誰かが証明したと言っているのをそのまま真に受けているという意味です。まあ、まちがってはいないと思いますが…。

プログラミング基礎の基礎

昔々、アラン・チューリングという名前の天才がいました。この人はイギリスで第二次世界大戦中の暗号解読に大きな成果をあげた人ですが、コンピュータ関係の基礎を打ちたてた人でもあります。彼の考え出したことが今のコンピュータの理論的背景になっているのです。

たとえば、彼の考え出したことのひとつに「チューリングテスト」というものがあります。これは「機械が知性を持つかどうか判定する」というテストです。これは自分の相手が完全に見えない状態、たとえばスクリーンを経由して、対話を行うというものです。相手が人間であるかコンピュータであるか教えられずに、ある程度の時間対話したうえで区別することができるかどうかをテストします。もし、対話を通して人間とコンピュータを区別することができなければ、それはコンピュータが十分に知性を持つことを意味しているとしても良いというのが彼の結論です。なんだかフォークト・カンプフテスト（映画「ブレードランナー」でレプリカントを区別するためのテスト）みたいですね。

彼は「チューリングマシン」という概念も考え出しました。チューリングマシンは、左右に無限の長さを持つテープと1つの読み書きヘッドによって構成される機械で、ヘッドがテープからデータを読みとり、それに従って記号操作を行い、その結果をテープに書き込むことを繰り返すことによって一種の計算を行います（**図1**）。

これは（まだ当時はコンピュータは存在していませんでしたが）、一種の仮想的なコンピュータで、彼はこの単純な機械によってあらゆるアルゴリズムが表現可能であることを証明したのです。いや、その証明は私には理解できませんが（笑）。

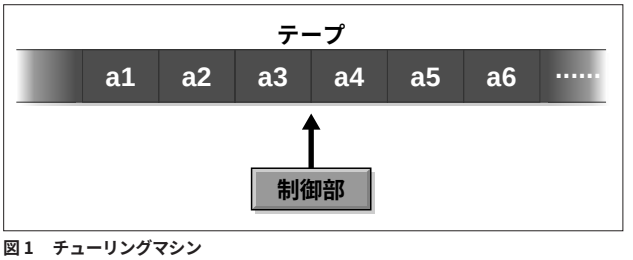


図1　チューリングマシン

チューリングマシンは非常に原始的な「コンピュータ」なので、あらゆるアルゴリズムが表現可能であっても、実際にこれでプログラムを行うのは現実的ではありません。しかし、あらゆる「アルゴリズムが表現可能である」ためにはそれほどたいした機能が必要ではないということがわかります。

チューリングマシンの単純さは、プログラミング言語がその本来の目的を果たすために必要な最低限の機能がとても小さいことを示しています。たとえばRubyの場合、そのリファレンスであるヤギ本（『Rubyデスクトップリファレンス』オライリー ISBN4-87311-023-8）で、言語そのものを解説した部分が全138ページ中、たったの24ページしかないことからもうかがえます。

チューリングマシンについてもっと知りたい人には、

```
http://www.jaist.ac.jp/~u-ryo/notes/class/ida/
CognitiveScience/turing.html
```

などからリンクをたどってみてください。

プログラミングの基礎

さて、役に立つのか立たないのかよくわからない基礎はまだ続きます。かなり時代が下がるとエドガー・ダイクストラが登場します。

彼は『GO TO statements considered harmful』という論文でプログラミング言語には以下の制御の流れがあれば、あらゆるアルゴリズムが表現できると証明しました。

- **逐次実行**
- **条件判断**
- **ループ**

かつてはプログラミング言語の制御構造には任意の場所にジャンプするGOTO文が必須であると考えられていました。ダイクストラの論文はそんなものは不要で、この3つのわかりやすい制御構造だけですべてが記述できると断言したことで当時としては画期的でした。

もちろん、私にはその証明は理解できません。が、結論は信じることにします。

逐次実行とは、命令を順番に実行していくことです。プログラミング言語は普通、並んでいる行を順番に実行します。これが逐次実行です。これを制御と呼ぶのは変な気もしますが、中には各命令の直後に次はどの命令を実行するかを指定する変な言語もありますし、さらにひどいことに逐次実行が進む方向（上下左右）を変えることができる二次元プログラミング言語というものもあります。どちらも異様に使いにくいですけどね。後者はジョークとして設計されたものですが、前者は本気だというところに恐いものがあります。

条件判断とは、ある条件が成立しているときだけ一連の命令を実行するものです。if文のことですね。

ループは、ある条件が成立するまで実行を繰り返すものです。while文のことですね。

if文

Rubyで条件分岐といえばifです。いや、Rubyでなくても普通のプログラミング言語ではどれもifですね。

ifのもっとも単純な形式は以下のようなものです。

```
if 式
  文
end
```

ifは式の結果が「真」であれば文を（複数の文があればそれを順番に）実行し、「偽」であればなにも実行しません。

ifを1行に書こうと思えば、式と文の間を明示するために、thenかセミコロン（;）を入れる必要があります。

```
if 式 then 文 end
if 式 ; 文 end
```

ですから、

```
if true
  puts "hello"
end
```

は、式（true）の値が真ですから、文である「puts "hello"」が実行されます。必ず実行されるんじゃ、条件分岐の意味ないですが。

では、Rubyでは、どのような値が「真」で、どのような値が「偽」なのでしょうか。定義から言えば、

・falseとnilが「偽」
・「偽」でない値はすべて「真」

です。すなわち、数字の0も文字列の""も含めて、falseとnil以外のすべての値は「真」になります。

「false」が英語で「偽」ですから、偽なのは当然だと思われます。ではnilとはなんでしょう？ 英和辞典によればnilとは、

無、零、無い、存在しない

という意味なのだそうです。ですから、Rubyにおいてもnilは「存在しないことを示す値」です。たとえば、代入されていない変数の中身はnilですし、配列の範囲を超えて要素を取り出そうとしたときに得られる値もnilです。

また、Rubyでは一般的に、

・真としてtrueを返す手続きは偽としてfalseを返す
・真としてtrue以外を返す手続きは偽としてnilを返す

という慣習があります。たとえば、「defined？ ()」はある式が定義されていれば、その種別を示す文字列を返しますが、定義されていなければnilを返します。ですから、

```
defined?(FOOBAR)
```

は、（FOOBARが定義されていなければ）nilを返しますが、

```
defined?(ARGV)
```

は、文字列"constant"を返します。やや、余談になりますが、Rubyでは「defined？」のような真偽値を返すもの（述語と呼びます）は名前の末尾に「？」をつける習慣があります。

「偽」としてfalseとnilの両方があることは、ときどき、

真 - 偽 - 未定義

という3つの状態を表現するのに使われることがあります。nilは偽というよりも「そもそも条件が成立しない」ということを表現していると考えることができます。

さて、真偽値についてはここまでにして、またif文に戻りましょう。さきほどは「条件が成立するときにこれを実行する」というif文の一番簡単な形式について説明しましたが、if文にはまだいくつかの形式があります。

「条件が成立するときにはこれを、成立しないときにはこっちを」というタイプの条件判断が欲しい場合があります。その場合にはelseの付いたif文を使います。

```
if 式
  文1
else
  文2
end
```

これは式の値が真であれば文1を、偽であれば文2を実行します。まあ、Rubyに限らず他の言語でも似たような形式ですよね。

次に「この条件が成立するときにはこれを、別の条件が成立するときにはこっちを、すべての条件が成立しない場合にはこれを」というタイプの条件判断にはelsifの付いたif文を使います。

```
if 式1
  文1
elsif 式2
  文2
else
  文3
end
```

これは式1の値が真であれば文1を、式2の値が真であれば文2を、すべての式が偽であれば文3を実行します。elsifの部分は実際には好きなだけ繰り返すことができますし、elseの部分は省略できます。

ここで、他の言語の経験がある人はelsifがCのようなelse ifでも、Pythonのようなelifでもないことに気をつけてください。PerlやAdaが得意な人はまちがえないと思います。もしわからなくなったら、「えるすいふ」と素早く発音すると「えるしふ」になると思い出してください。私がelsifを選んだのはまさにそういう理由からですから。

RubyにはPerlから引き継いだif文のもうひとつの形式があります。それが「if修飾子」です。if修飾子は「後置if」とも呼ばれていて、文の後にifをつけるものです。形式は、

```
文 if 式
```

のようなものです。たとえば、

```
puts "hello" if a == b
```

は、aとbの値が等しければ"hello"を出力します。ifは文の後ろについていますが、条件チェックは先に行われます。考えてみれば当然のことですが。

```
if 式 then 文 end
```

と、

```
文 if 式
```

は、まったく同じ意味です。好きなように使えばよいのですが、お勧めの使い分けとしては、条件を強調するときには普通のifを、文を強調するときにはif修飾子を使うというものがあります。

たとえば、変数$DEBUGがセットされているときにはデバッグ用のメッセージを出力したい場合には、

```
if $DEBUG
  puts "this is debug message"
end
```

とするよりも、

```
puts "this is debug message" if $DEBUG
```

としたほうが簡潔な印象があります。

unless文

条件分岐にはもうひとつunless文もあります。unless文はif文のちょうど反対で、条件が成立していないときに文を実行するものです。

```
unless 式
  文
end
```

unlessはif notと同じ意味ですから、主に意図を明確にするために用いられます。unless文にもif文同様にelseをつけることができます。

```
unless 式
  文1
else
  文2
end
```

これは式の値が偽であれば文1を、真であれば文2を実行します。でもこれは、

```
if 式
  文2
else
  文1
end
```

と同じですからあんまり意味がないですよね。ifにはelsifがありますが、unlessにはunlesifのようなものはありません。ifを使ってください。

unlessもifの反対ですから、当然ifと同様に修飾子形式が使えます。

```
文 unless 式
```

これは式の結果が偽のときだけ文を実行します。

while文

条件分岐の次はループです。ループのもっとも基本的なものはwhileです。whileの形式は以下の通りです。

```
while 式
  文
end
```

whileは式の結果が「真」であるあいだ、文を（複数の文があればそれを順番に）繰り返し実行し、「偽」になれば繰り返しを終了します。最初に式を評価した結果が「偽」であれば、文はまったく実行しません。

whileを1行に書こうと思えば、式と文の間を明示する

ために、doかセミコロン（;）を入れる必要があります。

```
while 式 do 文 end
while 式 ; 文 end
```

whileにも修飾子形式があります。形式は、

```
文 while 式
```

になり、これは、

```
while 式
  文
end
```

とまったく同じ意味です。まったく同じ意味ということは、修飾子形式では、後のほうに登場していても、式のほうが先に評価されるということです。

ただ、ループの場合本体を毎回実行して、繰り返しの終了条件のチェックはループの最後に行いたい場合があります。そのためにPascalではrepeat … untilという文がありますし、Cにもdo … whileがあります。

Rubyでは、そういう場合には後述のbreakを使う方法もありますが、beginとwhile修飾子というものもあります。

```
begin
  文
end while 式
```

という形式の場合、まず文を実行し、次に式を評価します。beginにはrescueやensureを付けてはいけません。でもねえ、この形式はあるので紹介しますけど、あんまりわかりやすい文法じゃなかったですよねえ。失敗か（苦笑）。

until文

ifに対してunlessがあるように、whileに対してはuntilがあります。

```
until 式
  文
end
```

untilは式が偽であるあいだ、文を繰り返します。また、while同様にuntilにも修飾子形式があり、begin文に付加されると式の評価の前に文を実行する点も同じです。

ループの中断

先にも説明した通り、逐次実行、条件分岐、ループの3つがあれば任意のアルゴリズムが記述できるのですが、とはいえ途中でジャンプできると便利なこともあります。

ジャンプといえばgotoですが、残念なことに（または幸いなことに）Rubyにはgotoはありません。その代わりもうちょっと「高級な」ジャンプがいくつかあります。今回はそのうちループに関するものを紹介しましょう。

ループに関するジャンプは以下の3つです。

●break

現在実行中のループを中断します。これを使えば最後に条件チェックするループも簡単に記述できます。

```
while true
  文
  break if 式
end
```

ここではwhile trueで無限ループを表現しています。そして、文を実行したあとで条件をチェックして、それが成立していればbreakで繰り返しを中断するわけです。

あるいは、beginとwhile修飾子の組み合わせよりもこっちのほうがわかりやすいかもしれませんね。

●next

ループ本体の実行を中断して末尾までジャンプします。主に以下のような使い方をします。

```
while line = gets()   # 1行ずつ読み込み
  next if line == ""  # 空行なら処理しない
  文                  # lineに対する処理
end
```

C言語のcontinueに相当します。continueという名前でないのはnextのほうが短いからです。Perlでもnextですね。

●redo

ループ本体の実行を最初からやりなおします。つまり条件チェックの直前にジャンプします。C言語にも相当するジャンプがないことからもわかるようにあまり使われません。

これらのジャンプをまとめると、**図2**のようになります。

for文

配列などの要素それぞれに対して繰り返すためには、whileのような単純なループよりもforのほうが便利です。たとえば、配列要素を出力するのであれば、while文を使うなら、

```
i=0
while i<ary.size
  puts ary[i]
  i+=1
end
```

のように、やや面倒になりますが、forを使えば、

```
for i in ary
  puts i
end
```

と単純になります。forは実際にはこのあと説明するイテレータを使って実現されています。

イテレータ

イテレータは奥が深いので、この先連載1回分まるまる使って説明しようと思いますが、簡単にいうとメソッドの一種で「ユーザーが定義できる制御構造」です。

さきほどのfor も内部的にイテレータを使って実現されています。ここでは仕組みは説明せず、いくつか便利なイテレータを紹介するだけにとどめます。

```
loop do          # loopは無限ループするイテレータです
  文
end

3.times do       # 文を3回繰り返します
  文
end

ary.each do |i| # forは内部的にこれと同じことをしています
  文
end
```

知られざるRuby
(中上級者向け)

end

前回はファイルをオープンするメソッドopenについて探求しましたが、今回は制御構造ということで、Rubyの制御構造を構成する重要なキーワードendについて考えてみます。

文のまとまりを表現する方法はプログラミング言語ごとにそれぞれあります。たとえば、Cであれば、

```
{ 文; }
```

でしょうし、Pascalであれば、

```
begin 文 end
```

でしょう。また、Pythonのように「インデントの深さで決める」という変わり種もあります。Pythonでは、

```
if a == b:
  foo
  bar
baz
```

という風にして文のまとまりの範囲を決めます。上の例をRuby的に書けば、

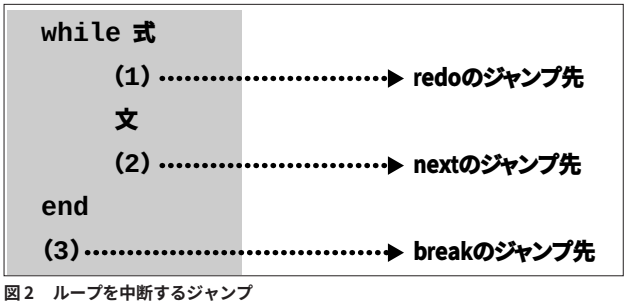


図2　ループを中断するジャンプ

```
if a == b
  foo
  bar
end
baz
```

という感じでしょうか。簡潔ですし（Rubyより1行少ないですね）、この形式を好む人もいます。

文のまとめ方はそれぞれですが、実は大きく分けると、

・複文派
・ブロック派

に別れるのだということをご存じでしたでしょうか。複文派とは「複数の文を文括弧でくくったものはひとつの文である」とする言語です。たとえば{}を文括弧とするCや、begin、endを文括弧にするPascalは複文派です。この派の言語では、単文が置けるところにはどこにでも複文が置けるというルールがあります。ですから、たとえばCの条件式で、

```
if (a == b)
  foo;
```

と書いたものは、fooの部分を複数の文に置き換えたければ、{}でくくった複数の文を置けばよいわけです。

```
if (a == b) {
  foo;
  bar;
}
```

このやり方には2つほど問題があります。ひとつは単文から複文への書き換えが面倒なことです。たとえば上記の変更では、ただbarという文が追加したいだけなのに、fooの前に「{」を、barの後に「}」を追加する必要があります。

もうひとつの問題は「ぶらさがりelse」という問題です。またまたCの例になりますが、

```
if (a == b)
  if (c == 5)
    foo;
else
  bar
```

と書いてあった場合、これをどう解釈しますか？

```
if (a == b) {
  if (c == 5) {
    foo;
  }
}
else {
  bar;
}
```

と解釈したくなるでしょうが、実際にはインデントに関係なく、

```
if (a == b) {
  if (c == 5) {
    foo;
  }
  else {
    bar;
  }
}
```

と解釈されます。

もう一方のブロック派は複文をとる文は、いつもブロック（文のまとまり）をとることが決まっている言語です。ブロック派言語には上記の2つの問題は存在しません。たとえばRubyであれば、

```
if a == b
  foo
end
```

にbarを追加したければ、ただfooの後の行に追加するだけです。他の部分を変更する必要はありません。「ぶらさがりelse問題」についても、

```
if a == b
  if c == 5
    foo
```

```
  end
else
  bar
end
```

と、

```
if a == b
  if c == 5
    foo
  else
    bar
  end
end
```

は明らかに異なりますから、混乱を招く可能性はありません。このようにユーザーにとっての使いやすさの観点からみれば、ブロック派言語の圧勝と言えるでしょう。最近開発された言語ではブロック派が主流になっているように感じられます。たとえば、Rubyと同じスクリプト言語に分類されるものに限っても、

- **まとまりはいつも{}でくくる必要があるPerl**
- **まとまりはインデントで表すPython**
- **まとまりの終わりをendで表すRuby**

は、いずれもブロック派言語です。上記の3つの言語では同じブロック派でも記法がそれぞれまったく異なるのが面白いですね。

Rubyを初めて見た人はendによる文区切りが結構異様に感じられるようです。endを使う言語の中でもっとも有名なPascalの人気は下降する一方ですから、無理もないのかもしれません。

では、Rubyはなぜ end を文の区切りに選んだのでしょうか。それにはいくつかの理由があります。

●endはAlgolから続く伝統的な文区切り

Rubyは比較的「保守的」な部分があります。最近優勢なC、C++、Javaとは違いますが、endにはここ数十年文区切りとして使われてきた伝統があります。

●beginやcaseがきれい

たとえば現在のRubyではbegin文は以下のような構文になります。

```
begin
  文
rescue 例外クラス
  文
ensure
  文
end
```

仮にこの構文を{}で表現しようと思うと以下のようになるでしょう。

```
begin {
  文
}
rescue（例外クラス） {
  文
}
ensure {
  文
}
```

なんだか間が抜けているようですし、あまりきれいではありません。case文でも同じような状況があります。ま、主観や趣味の問題かもしれませんが。

●オートインデントが可能

Emacsであればruby-modeを使ってオートインデントができます。予約語を使った言語のオートインデントは記号を使ったものよりも難しいのですが、ruby-modeによってオートインデントが可能になる見通しがついたことでendを採用する決心がついたというのは正直な話です。

とはいえendに欠点がないわけでもないです。

●文区切りに別の行が必要となるので行数が増える

しかし、逆に文の並びの終わりがはっきりわかってよいという側面もありますから、行が増えることがただちに問題であるということにはなりません。

●文区切りが識別子と区別がつきにくく埋没しがち

最近のエディタは予約語に色がつけられますから、この点はさして問題にならないかもしれません。

Ruby開発日記

Linux Conference 2001

9月26日から9月28日まで、東京の明治記念館においてLinux Conference 2001が開催されました。例年、Linux Conferenceはプログラム委員から講師候補に直接コンタクトがあったのですが、今年はCall for Paper（論文応募）方式に変更されました。

正直、論文を用意するのも大変なので、今年は発表しないつもりでいたのですが、広島市立大の木山さんが応募した論文が2本も通ったので、セッションをひとつ作るために「埋め草」として話しませんか、と誘われたので引き受けることにしました。頼まれるとなかなかイヤとは言えないのです。

ということで、スピーカー特権で、26日から参加してきました。プログラムは、

http://lc.linux.or.jp/

から参照できます。

26日には、『Rubyを256倍使うための本 ― 無道編』の著者である青木峰朗さんによるチュートリアル『Ruby活用～オブジェクトは恐くない』に参加しました。青木さんのチュートリアルは、オブジェクト指向設計に注目して、いかに共有を押し進めるか、いかに「良い」プログラムを書くかということについて解説がありました。発表資料は、

http://www.loveruby.net/~aamine/ja/lc2001/

から入手できます。とはいえ、青木さんはこの手の発表は初めてだったようで、かなり緊張していました。『無道編』や『紅玉制覇編』でのなめらかな語り口は聞かれませんでした。ちょっと残念。また、チュートリアルでは実際に段階的に「良い」プログラムを開発していくという形式だったのですが、「良い」プログラムになっているはずなのに、なんとなくだんだん複雑になっていくという皮肉な現象も観測されました。

27日午後には「プラットフォーム:Ruby」と名付けられたRuby関連セッションが開催されました。内容は木山さんによる「オブジェクト指向スクリプト言語Rubyへの世代別ごみ集め実装手法の改良とその評価」、木山さんと同じ大学の佐原大輔さんによる「RubyのVM化に向けて」、それから私による「Rubyの心理学」の3つの発表が行われました。

木山さんの世代別GC（ガベージコレクション）の実装は、その基本的実装について昨年11月のLinux Conferenceでも発表されましたし、私は実際に一緒に開発も行いましたので、私にとっては耳新しいものではありませんでしたが、GCとはなにか、というところから始まって、世代別手法がどのような仕組みで効率を改善するのかという点と、また実際にどの程度の改善が行われるかについて、よくまとまった発表でした。実際に全体の実行時間が39.3％、GC実行時間が88.7％も削減されたという話でした。

もちろんすべてのケースでここまで高速化されるわけではないのですが、GCの実行速度は、ふだんはあまり問題にならず、問題になるような場合というのは大量のオブジェクトを取り扱うケースで、そのような局面において世代別GCが有効であること、また、今年に入ってからの改善でさらに高速化されたことが示されました。ただ、木山さんにとって気になる点がいくつか残っているそうで、今のRubyに取り込むよりも、将来のVMバージョンに組み込んだほうが良いのではないかと言っていました。

佐原さんのVMの実装の話はなかなか興味深いものでした。VMというのは仮想機械（Virtual Machine）の略で、プログラムを仮想的なCPUの機械語（よくバイトコードと呼ばれます）に変換して、実行しようとするものです。また、この仮想的なCPUのエミュレータのことをVMと呼びます。

私はこの実装についてはなにもタッチしていません。まだプロトタイプレベルで、Rubyプログラムを直接実行するレベルには達していないとはいえ、独自にここまで開発を行う人が出てくるというだけも、私にとっては驚異です。実際、Rubyは処理系開発者よりも、プログラマーに対してやさしいように設計されているので、処理系を開発するのは結構大変なのです。

さて、今回実装されたVMは、まだ構文解析を行うパーサはなくて、アセンブラしか解釈できないうえに、クラスライブラリも数値しかないとのことですが、それでもフィボナッチ関数（fib）や竹内関数（tak）の実行は、現在のRubyと比較して3倍近く高速化されるのだそうです。

また、目標は現状よりも10倍速い処理系を実装することだそうです。本人も難しいだろうとは自覚しているようですが。

Ruby実装者の立場から聞くと、静的言語向けのVMであるJava VMをベースに設計したのはどうだろうか、とかいろいろ疑問があるにはあるのですが、なかなか頼もしい発表でした。

私のものは前2つと比較すると恥ずかしいくらい不真面目な内容でした。要するにRubyってのはどういうふうに人間向けにデザインされているかとかいうような内容です。

簡単にまとめると、プログラミング言語は実はマン・マシン・インターフェイスであり、ヒューマンインターフェイスの原則が適用されるということと、そしてその結果、言語設計は人間に注目して行われるべきということを紹介したつもりです。

私の発表資料は、

http://www.ruby-lang.org/ja/lc2001/

で公開しています。

セッション終了後、明治記念館の一室を借りて、Ruby BOFが開かれました。BOFとはBird Of a Featherの略で、ひとつのことに興味がある人が集まってワイワイやる集会のことです。今回も日本にはRubyユーザーグループがない話から、VMの実装の話まで多岐にわたって話が制限時間一杯まで行われました。

28日にもライトニングトークをはじめとして、Rubyの話がいくつかあったようなのですが、この日は娘の誕生日なのであきらめて帰りました。家族からこれ以上不評を買ってもいけませんから。

サーバ百科
基礎から学ぶネットワークの仕組み

最も基本的なクライアントサーバプロトコルである telnetの基本と使用方法を解説する。

第4回　telnetサーバを使う

文：安田幸弘
Text : Yukihiro Yasuda

telnetを知らないLinuxのユーザーはおそらくいないだろう。telnetは、ネットワークを通じてほかのマシンにログインするために使われるプロトコルで、インターネットのプロトコルの中でも、最も古く、そして基本的なプロトコルだといえる。イーサネットの仕様ができるよりも早く、すでに1972年頃には基本的な骨格が固まり、1980年代に入って4.2BSDに実装されて以来、現在に至るまで広く使われている歴史の古いプロトコルだ。

今回は、最も基本的なクライアントサーバプロトコルとして、telnetのインストールと使用方法を解説するが、まず簡単にtelnetの基本を確認しておくことにしよう。「とりあえず使いたい」という方は、基本知識の部分は飛ばしていただいてもかまわない。

telnetの基本知識

telnetは、ネットワークを通じてほかのマシンにログインするためのプロトコルだ。具体的には、リモートマシンにインストールされたtelnetサーバにローカルマシンのtelnetクライアントを使って接続し、ローカルマシンの端末をリモートマシンの端末として使えるようにする。

一般のインターネットユーザーの中には、telnetを「要するに通信ソフトみたいなソフト」と誤解している人もいるようだが、通信ソフトと呼ばれる種類のターミナルエミュレータとtelnetクライアントはまったくといっていいほど異なるソフトだ。

まず、「端末」、あるいは「ターミナル」という言葉だが、これはコンピュータを操作するためのCRTやキーボード、プリンタ、マウスなどを備えたマシンのこと。コンソール（操作卓）と呼ぶこともある。

Linuxワークステーションのようなマシンでは、本体とコンソールが一体になっているが、昔の大型コンピュータでは本体と端末装置が別々になっていて、1台のCPUに何台かの端末機をシリアルケーブルなどでつないで使うものだった。ターミナルエミュレータの画面制御モードに名前を残している「VT100」や「VT52」といった型番は、数十年前に一世を風靡した端末装置の機種名である。

その後、パソコンが高性能化してきたことで、高価な専用端末装置の代わりにパソコンが使われるようになってきたのだが、パソコンを端末装置として使えるようにするソフトが、いわゆるターミナルエミュレータ、あるいは通信ソフトと呼ばれるソフトだ。昔の端末装置は、本体の近くに置かれるのが普通だったが、いわゆる「通信ソフト」という場合は、モデムなどを使って遠隔地からホストを利用することができる。つまり、通信ソフトは、昔の端末機のケーブルを電話回線を使って思い切り長く延して使えるようにするソフトだということができるかもしれない（**図1**）。

端末装置、あるいはターミナルエミュレータは、キーボードから入力された文字を送信し、ホストコンピュータから送られてきた文字を画面に適切に表示することで、その

ための画面制御機能や、通信制御機能を持つのが特徴だ。要するに通信ソフトは、端末機能を実現するために必要なすべての機能をパッケージ化したソフトである。

ちなみに、X Window Systemのコマンドシェルを利用するために使うXTermやKTermといったソフトもターミナルエミュレータと呼ばれるが、その理由はワークステーションのキャラクタコンソールをXのウィンドウでエミュレートするためのソフトであることを考えれば納得できるだろう。XTermは、端末専用機をケーブルでつなぐ代わりに、Xの中にソフトウェア的な端末装置を実現する。

前置きが長くなってしまったが、端末の機能を実現するための通信ソフトに対して、telnetが提供するのは中間的なネットワーク仮想端末（NVT：Network Virtual Terminal）と、それに必要な通信制御である（**図2**）。telnetが提供する仮想端末は、ごくシンプルなラインプリンタのようなダム端末的な機能だけだが、オプションを使って実際の端末が必要とする機能に対応できるようになっていて、telnet単独では端末機能を持たない。実際に端末として使われるのは、XTermのようなターミナルエミュレータやコンソールということになる。

また、ローカルマシンの端末側からは、同様にリモートのサーバがサポートする端末について気にする必要はなく、あくまでもネットワーク仮想端末として情報をやりとりすればいい。

telnetでは、さまざまなオプションがサポートされているが、時代とともにさまざまなオプションが追加されてきた。たとえば当初、telnetは7ビットのASCIIキャラクタしか利用することができなかったが、8ビットバイナリモードのオプションが追加されたことで、ヨーロッパの国際文字セットや日本語をはじめとするマルチバイト文字セットが使えるようになっている。このほかにも各種のオプションが追加されており、RFCに公開されている。主なオプションは1980年代に固まっているが、RFCのインデックスを“telnet”や“option”でgrepすると、RFC化され

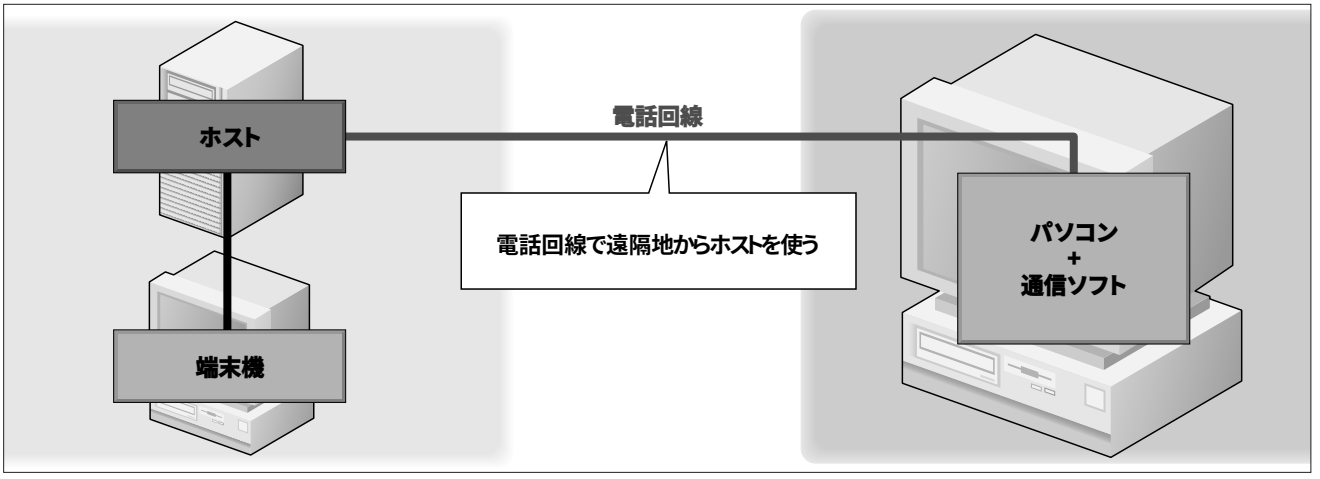


図1　パソコン通信端末ソフト

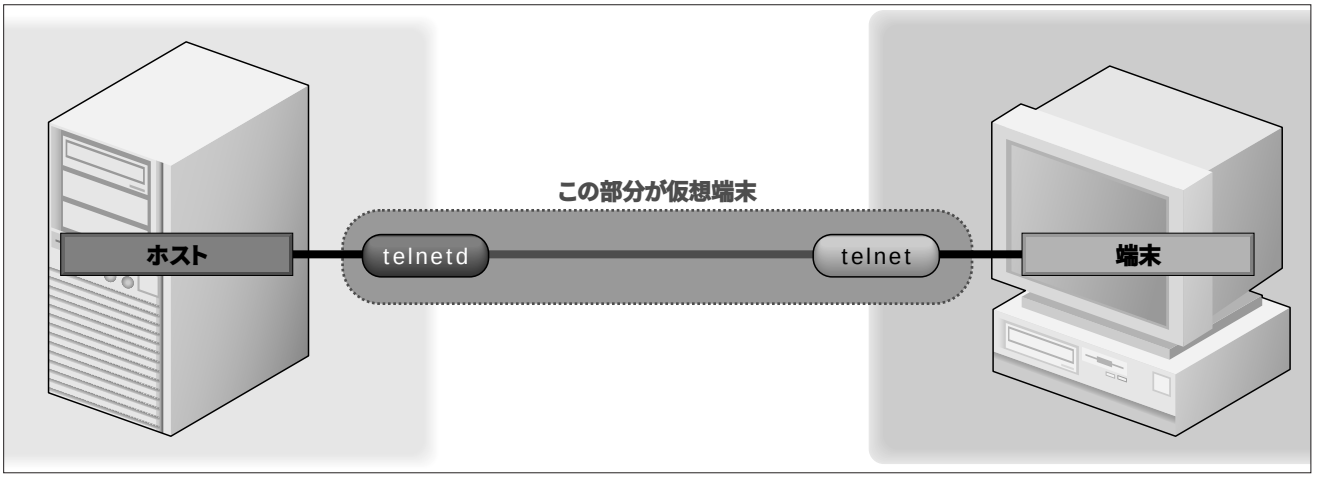


図2　telnetは中間的な仮想端末を提供する

ているオプションの一覧を得ることができるだろう。

■telnetの手順

次にtelnetによる通信のシーケンスを見てみよう。

telnetの通信手順は、まずサーバがTCPポート23への接続を監視し、接続があれば標準的な手続きでTCPセッションを張る。

TCPセッションが確立すると、telnetのネゴシエーションを行い、ここでNVTが持たない画面制御などの高度な端末機能についてのオプションが設定される。

双方に共通するNVTの制御コマンドは、端末側、あるいはホスト側からIAC（Interpret as Command）コマンドと、それに続く制御コマンドで伝えられ、受け付けられると必要な情報が返される。

高度な端末機能を実現するオプションのネゴシエーションの手続きも、やはり端末側、あるいはホスト側から必要な機能についての要求が出され、その要求に対してサポートできるかどうかの返事が返されるのだが、NVTと違って相手がどのような機能をサポートしているのかは、最初の段階ではわからない。そこで、それぞれが利用したい機能をDO、またはDONTのコマンドで伝え、それに対してWILL、またはWONTで受け入れが可能かどうかの返事を返す。つまり、ある機能を使いたいという要求なら「DO 機能名」、使いたくなければ「DONT 機能名」を相

```
$ telnet
telnet> set options
Will show option processing.
telnet> open broadway
Trying 192.168.46.38...
Connected to broadway.localdomain.
Escape character is '^]'.
RCVD DO TERMINAL TYPE
SENT WILL TERMINAL TYPE
RCVD DO TSPEED
SENT WILL TSPEED
RCVD DO XDISPLOC
SENT WILL XDISPLOC
RCVD DO NEW-ENVIRON
SENT WILL NEW-ENVIRON
RCVD IAC SB TERMINAL-SPEED SEND
SENT IAC SB TERMINAL-SPEED IS 9600,9600
RCVD IAC SB X-DISPLAY-LOCATION SEND
SENT IAC SB X-DISPLAY-LOCATION IS "com.localdomain:10.0"
RCVD IAC SB NEW-ENVIRON SEND
SENT IAC SB NEW-ENVIRON IS VAR "DISPLAY" VALUE "com.localdomain:10.0"
RCVD IAC SB TERMINAL-TYPE SEND
SENT IAC SB TERMINAL-TYPE IS "XTERM"
RCVD WILL SUPPRESS GO AHEAD
SENT DO SUPPRESS GO AHEAD
RCVD DO ECHO
SENT WONT ECHO
RCVD DO NAWS
SENT WILL NAWS
SENT IAC SB NAWS 0 80 (80) 0 30 (30)
RCVD WILL STATUS
SENT DO STATUS
RCVD DO LFLOW
SENT WILL LFLOW
RCVD WILL ECHO
SENT DO ECHO

Red Hat Linux release 7.1 (Seawolf)
Kernel 2.4.9-6Y on an i686
login:
```

画面1　セッション開始時のオプションのネゴシエーションの例

手側に送る。これに対して、その要求を受け入れる場合は「WILL 機能名」、受け入れない場合は「WONT 機能名」を返すことになる。

なお、オプションの設定は主にセッションの開始時に集中的に行われるが、たとえば端末の画面サイズを変更した場合など、セッション中に変更することもできる。

画面1はtelnetの「set options」コマンドで表示させたオプションのネゴシエーションのようすで、これを見ると、実際にtelnetプロトコルがどのような情報をやりとりしているかがわかる。

telnetサーバのインストール

telnetは、リモートログインという限られた機能を提供するため、サーバやクライアントのインストールは非常に簡単だ。たとえばRed Hat系のディストリビューションなら、スーパーユーザーになって「rpm -ihv RPMファイル」のコマンドを実行し、telnetのバイナリパッケージをインストールするだけでよい。

ただし、インストールしただけではサーバは起動しない。telnetは、通常inetdまたはxinetdから呼び出されるので、Red Hat 6.xならinetd、7.x以上はxinetdでtelnetが起動できるようにする。また、/etc/servicesにサービス名（telnet）に対応するTCPポートの指定が必要なので、inetdやxinetdの設定を行ってもtelnetサーバが反応しない場合は、/etc/servicesにtelnetの設定を加える。

●inetdでの設定

① /etc/inetd.confにtelnetdの行（リスト1）を追加

② kill -SIGHUP `cat /var/run/inetd.pid`を実行してinetdを再起動

●xinetdでの設定（Red Hat 7.xの場合）

① /etc/xinetd.d/telnetをチェック

「disable = yes」になっていれば、「disable = no」に変更する

② kill -SIGUSR2 `cat /var/run/xinetd.pid`を実行してxinetdを再起動

●servicesの設定

/etc/servicesに以下の行があることをチェック

```
telnet          23/tcp
```

なおtelnetに限らず、inetdやxinetdから呼び出されるサーバが接続を待ち受けるポートの番号は、/etc/servicesで変更することができる。たとえば、待ち受けるポートの番号を2323に変更するには、/etc/servicesのtelnetの行を次のように変更する。

```
telnet          2323/tcp
```

それぞれデフォルトの設定が異なる複数のtelnetサーバをインストールした場合は、/etc/servicesに異なるサーバのための行を追加し、inetdやxinetdに設定項目を追加すればよい。

■telnetサーバの起動オプション

telnetdは、inetd.confやxinetd.confの中で引数を設定することで、デフォルトの動作を変更することができる。もちろん、telnetdのバージョンや種類、コンパイル時のオプションによって、使えるオプションの種類は異なるが、Red Hat 7.xのディストリビューションに含まれるnetkitベースのtelnetdでは次のようなオプションを指定できる。

-debug ［ポート番号］

inetd、xinetdを使わず、telnetdを直接起動する。デバッグや動作の確認などの必要があるときに、このオプションを使う。接続を待ち受けるポート番号を指定することもできる。

-D （options | report | netdata | ptydata）

デバッグ用オプション。このオプションを付けて起動すると、それぞれのデバッグモードに応じてtelnetクライアントにさまざまな情報が表示される。

デバッグモードの内容は次の通り。

options	**telnetオプションのネゴシエーションを表示**
report	**optionsに加え、キー入力の情報などの付加情報を表示**
netdata	**データストリームの内容を表示**

▼リスト1　inetd.confに加えるtelnetデーモンの設定

```
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
```

ptydata　ローカルの疑似ターミナルに表示されデータを表示

-h

接続時にホスト情報の表示を抑制する。

-L login_program

/bin/login以外のログイン用プログラムを指定する。

-n

TCPキープアライブを使わない。異常終了したクライアントとの接続が持続する場合などにこのオプションを指定する。

たとえば、-hオプションで接続時にホスト情報を表示させないようにするためには、次のように設定する。

●inetdでの設定

/etc/inetd.confの「telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd」の行の最後に「-h」を加え、inetdを再起動する。

●xinetdでの設定

/etc/xinetd.d/telnetに次の行を加え、xinetdを再起動する。

```
server_args     = -h
```

このほか、netkit-telnetdでは、コンパイル時のオプション指定により、認証モードの指定（-a 認証モード）、暗号化telnet使用時のデバッグオプション（-edebug）、セキュアIDカード使用の指定（-s）、IPサービスタイプ（TOS）の指定（-S TOS）、認証タイプの指定（-X 認証タイプ）を有効にすることができる。

変数	意味
%l	現在の端末番号
%h, %n	システムのFQDN
%D, %o	NISドメイン名
%d, %t	現在の日時
%s	OSの名前
%m	ハードウェアのタイプ
%r	OSのリリース
%v	OSのバージョン
%%	「%」の文字

表1　/etc/issue.netで利用できる変数

さらに、BSD系のUNIXに付属するtelnetdでは、このほかにもいくつかのオプションを指定することができる。

■ホストのメッセージ表示

/etc/issue.netがあれば、telnetdはログインする前に端末側に/etc/issues.netの内容を表示する。

このファイルの書式には、**表1**のような変数で現在のホストの情報を表示することができる。

たとえば、「%s （%m） %r」という行を/etc/issue.netに書いておくと、telnetで接続したときにloginプロンプトの前に「Linux （i686） 2.4.9-6s」などの文字列が表示される。なお、telnetdに-hオプションを付けることで、このファイルの内容の表示が抑制される。

■サーバへの接続の制御

telnetサーバは、特に接続制御に関するメカニズムを持っていない。そのため、特定のホストからの接続を拒否したり、あるいは許可したい場合は、inetdからtcpdを起動して接続制御を行ったり、xinetdから呼び出される場合はxinetdが持つ接続制御メカニズムを利用することになる。

これらの接続制御の方法は、本連載の第3回で解説した通りである。ここでは簡単にtelnetの接続制御について、具体的な設定内容をご紹介しておこう。

●inetdを使う場合の接続制御

まず、/etc/inetd.confの設定で、リスト1のようにtelnetの行でtcpdが使われていることを確認する。

tcpdが使われていない場合は、前述の**リスト1**のように起動するサーバとしてtcpdを指定し、tcpdからin.telnetdを呼び出すように変更しておく。

次に、/etc/hosts.allow、/etc/hosts.denyを編集する。/etc/hosts.denyには、「ALL： ALL」の1行だけを記入し、すべてのサーバに対してすべてのホストからのアクセスを禁止して、許可する接続だけを/etc/hosts.allowで指定するとよい。

すべてのホストからtelnetdへの接続を許可する場合は、/etc/hosts.allowに、「in.telnetd： ALL」の行を加える。

特定のホストからtelnetdへの接続を許可する場合は、/etc/hosts.allowに、「in.telnetd： ホスト名」の行を加える。

また、特定のネットワークからtelnetdへの接続を許可する場合は、/etc/hosts.allowに、「in.telnetd： ネットワークアドレス/ネットマスク」の形式で許可するネット

ワークの行を加える。たとえば、192.168.0.0から192.168.0.127までのネットワークにアクセスを許可する場合は、「in.telnetd： 192.168.0.0/255.255.255.128」を書き加えればよい。

●xinetdを使う場合の接続制御

xinetdは、tcpdと同等の接続制御メカニズムを内蔵しているので、この機能を使う。

/etc/xinetd.d/telnetのファイルに、「only_from ホスト」の形式で接続を許可するホストまたはネットワークを指定する。

接続を許可するホストの指定には、IPアドレス、ネットワークアドレス/ネットマスク、ホスト名、ネットワーク名などが使え、たとえば特定のホストだけを許可する場合、「only_from 192.168.0.1」、ネットワークなら「only_from 192.168.0.0/255.255.255.128」のように書き加える。

なお、設定を変更した場合は、xinetdを再起動する。

■telnetクライアントの使い方

/usr/bin/telnetなどの標準的なtelnetクライアントの最も基本的な使い方は、接続先のホスト名を引数として「telnet 接続ホスト名」のコマンドを実行する。ほとんどの場合、これで十分だが、このほかにも**表2**のようなオプションを指定することができる。

さらに、telnetクライアントのバージョンやコンパイル時のオプションにより、これ以外のオプションが利用できるものもある。特にKerberosを認証システムに使用している場合、認証関連のオプションが利用できるようになる。

ところで、telnetクライアントは、エスケープ文字を打つことでコマンドモードに入り、各種のtelnetのコマンドを使って操作することができる。やはり、認証や暗号化のオプションを付けてコンパイルされたクライアントと、そうでないクライアントでは利用できるコマンドは異なるが、ここでは特に一般的なRPMで提供されているnetkit-telnetクライアントで利用できる主なコマンドの使い方を簡単に説明しよう。

open ホスト名 ［オプション］

ホストへのtelnetセッションを開く。このとき、「-l ユーザー」のオプションを付けると、そのユーザー名をloginの引数にする。また、「-ポート番号」でポートを指定することで、任意のポートに接続する。

close

現在のセッションを終了する。

logout

接続先からログアウトしてセッションを終了する。

quit

セッションを切断し、telnetコマンドを終了する。

display ［設定項目］

指定した設定項目の値を表示する。表示する設定項目が指定されなければ、現在設定されているすべての項目を表示する。

set 設定項目 設定値

設定項目の設定値をセットする。ここで設定できる設定項目は**表3**の通りだが、設定項目の中にはいくつかの条件の下で有効になるものがあるので、詳細はマニュアルページを参照していただきたい。

unset 設定項目

設定項目の設定値を消去する。

toggle 設定項目

設定項目の設定値を切り替える。切り替えることができ

オプション	意味
-8	8ビットデータの入出力を可能にする。EUCコードなどの通信に必要
-L	8ビットデータの出力を可能にする
-E	コマンドモードに移行するためのエスケープキャラクタを無効にする
-e エスケープ文字	コマンドモードに移行するためのエスケープキャラクタの定義を変更する。エスケープ文字のデフォルトは「^」（0x1d）
-S TOS	IPのサービスタイプを指定する。パケットの優先度やコストなどを設定することができるが、通常、このオプションは使用しない
-a	オートログイン。環境変数USERの値でloginを起動させ、プロンプトでユーザー名の入力を省略する
-l ユーザー	指定したユーザー名でオートログインを実行する
-c	~/.telnetrcを無効にする
-d	デバッグ情報を表示
-n トレースファイル	デバッグ用のトレース情報の記録用ファイルを指定する
-b ホストの別名	ローカルマシンの別名を設定する
-r	操作をrloginに似たものにする。エスケープ文字は「~」になる
ポート番号	接続先のポート番号を指定する

表2　telnetコマンドのオプション

る設定項目は、setの記述を参照。

mode モード

キャラクタモードを設定する。

character LINEMODEオプションを無効にする
line LINEMODEオプションを有効にする

以下のモード設定は、LINEMODEでのみ有効。各モードの前に「-」を付けることで設定内容を逆にすることができる。

edit 文字編集を有効にする
isig シグナルのトラップを有効にする
litecho 入力された文字のエコーを有効にする
softtabs タブ記号をスペースに展開する

設定項目	意味
ayt	“ Are you there? ” を送出させる文字列
echo	入力文字のエコーバックを切り替えるための文字
eof	ファイル末尾シーケンス（EOF）を送出させる文字（LINEMODE）
erase	1文字抹消シーケンス（EC）を送出させる文字（LINEMODE）
escape	コマンドモードに入るためのエスケープ文字
flushoutput	出力中断シーケンス（AO）を送出させる文字（localchars）
forw1	代替行末文字1（LINEMODE）
forw2	代替行末文字2（LINEMODE）
interrupt	プロセス割込シーケンス（IP）を送出させる文字（localchars）
kill	行抹消シーケンス（EL）を送出させる文字（localchars）
lnext	次の文字をリテラルにするシーケンスを送出させる文字（LINEMODE）
quit	プロセス終了シーケンス（BRK）を送出させる文字（localchars）
reprint	行を再表示させるシーケンスを送出させる文字（LINEMODE）
rlogin	rloginに慣れている人のためのエスケープ文字
start	XONを送出させる文字
stop	XOFFを送出させる文字
susp	プロセスをサスペンドするシーケンスを送出させる文字（LINEMODE／localchars）
tracefile	トレースファイル名
worderase	1語抹消シーケンス（EL）を送出させる文字（LINEMODE）
以下の項目はtoggleで切り替えることができる	
autoflush	オートフラッシュを有効にする
autosynch	オートシンクを有効にする
binary	バイナリ情報の送受信を可能にする
crlf	改行文字を<CR><LF>にする
crmod	受信した改行文字を<CR><LF>にする
inbinary	バイナリ情報の受信を可能にする
localchars	LINEMODEが設定されていない場合に、flush、interrupt、quit、erase、killがそれぞれao、ip、brk、ec、elのシーケンスに変換される
outbinary	バイナリ情報の送信を可能にする
skiprc	~/.telnetrcを無視する
以下の項目はデバッグ用の設定	
debug	デバッグを有効にする
netdata	ネットワークを流れるデータを16進数で表示
prettydump	“ netdata ” の表示を見やすくする
options	telnetオプションのやりとりを表示する
termdata	端末データを16進数で表示

表3　setコマンドで利用できる設定項目

send シーケンス

指定されたシーケンスを送出する。送出できるシーケンスは**表4**の通り。

status

接続の状態を表示する。

slc 引数

LINEMODE時の特殊文字の設定・変更を行う。slcの引数は以下の通り。

check 現在の特殊文字の設定を確認する
export 特殊文字の機能をローカルでの設定に変更する
import 特殊文字の機能をリモートでの設定に変更する

environ 引数

環境変数の値を閲覧・設定する。ここで指定できる引数は以下の通り。

define 変数 変数を有効にする
undefine 変数 変数を無効にする

シーケンス	意味
ao	すべての出力をフラッシュ
ayt	“ Are You There ” シーケンス。アイドル状態などで端末の利用を確認する場合などに使う
brk	ブレーク
ec	1文字抹消
el	1行抹消
escape	現在のエスケープ文字
ga	“ Go Ahead ” シーケンス。半二重回線などで入力の終了を伝えるときなどに使う
ip	プロセス割り込み
nop	“ No operation ”。何もしない
eor	レコード終了
abort	プロセスの中断
susp	プロセスの停止
eof	ファイルの末尾
synch	コマンドの同期。フローコントロールの問題などで画面表示を急いで止めたいときなどに使う
getstatus	STATUSを要求する

表4　sendコマンドで指定できるシーケンス

export 変数	自動的に変数をリモートに送る
unexport	自動的に変数をリモートに送らない
send	変数をリモートに送る
list	変数の一覧を表示する。「*」が表示されている環境変数はexportされている変数を示す

z

telnetの実行を中断し、コマンドプロンプトに戻る。fgで復帰する。

! コマンド

シェルを起動し、コマンドを実行する。

?

簡単なヘルプメッセージを表示する。

■~/.telnetrc

最近は、リモートマシンもローカルマシンも、洗練されたUNIX系OSが使われるようになり、特にクライアント側で複雑な設定をする必要がないケースがほとんどだ。しかし、何かの設定が必要な場合や、特定のホストだけで異なる設定が必要になる場合、telnetの引数をいちいち変える必要があり、面倒に感じることもある。

そのような場合は、リモートマシンごとに設定内容をホームディレクトリの~/.telnetrcに書き込み、接続先に合わせてtelnetのパラメータを自動的に設定できる。

.telnetrcの書式は、**リスト2**のように、まずホスト名のラベルを書き、次行から行頭にスペースを入れて設定内容を列挙する。

なお、.telnetrcでは行頭が「#」で始まる行はコメント行として読み飛ばされる。

▼リスト2 ~/.telnetrcの書式

```
ホスト名
   設定項目
   設定項目
     ⋮
ホスト名
   設定項目
   設定項目
     ⋮
```

使い捨てパスワードによるログイン

現在、広く使われているtelnetは、平文パスワードがネットワークを流れる点がセキュリティ上の問題点とされ、インターネットではtelnetを利用せず、安全なSSHなどを使うようになっている。

telnetがインターネットの成立以前に開発されたことを考えればやむを得ないが、このようなtelnetの欠点を補う方法がいくつか考えられている。その中のひとつが、ワンタイムパスワードと呼ばれる使い捨てのパスワードでログインする方法だ。ワンタイムパスワードの原理は、あるパスワードの「種」から計算できる一連のパスワードを次々と使い捨てていくというもの。パスワードの「種」さえ盗まれなければ、いくらパスワードを盗まれても、そのパスワードはもう使えないので、平文でパスワードが流れるネットワークでも、安心して計算したパスワードを流すことができる。

ワンタイムパスワードは、パスワードの管理がやや面倒であることから、あまり使われていないが、システムに大きな改造を加えずに平文パスワードを使用する危険を減じることができる。ふだんはSSHなどを使うとしても、何かのときの用意に、telnetをワンタイムパスワードで使えるようにしておくと安心だ。

■インストールと初期化

残念ながらLinuxで使える標準的なワンタイムパスワードのモジュールは少ない。しかしtarボールから自分でコンパイルしなくても、インターネットの検索やRPM Find（http://www.rpmfind.net/）などで「opie」をキーにして探せば、いくつかのバイナリが見つかるので、適当なopieパッケージをインストールすればいいだろう。

モジュールをインストールした後、ワンタイムパスワードを「opiepasswd -c」で初期化する。このコマンドを実行すると、パスフレーズのプロンプトが表示されるので、適当なパスフレーズを入力する。これで/etc/opieディレクトリに、ユーザーのワンタイムパスワード用の「種」とマスターパスワードの情報が作成される。

■テスト

初期化が終了したら、opieloginコマンドを実行して、ワンタイムパスワードが使えることを確認しよう。

opieloginを実行すると、loginプロンプトが表示される

ので、自分のユーザー名を入力する。入力したユーザー名に対応する「種」があれば、「種」の情報を「otp-md5 498 ab1234 ext」などの形式で表示し、続いて「Response or Password：」というプロンプトが表示される。

ワンタイムパスワードは別のターミナルのウィンドウでopiekeyコマンドで計算できる。上記の例であれば、「opiekey 498 ab1234 ext」と「種」の情報を引数としてopiekeyコマンドを実行すればよい。すると、パスワードの入力プロンプトが表示されるので、ここに通常のパスワードを入力する。すると、「HALF SILT BILL MITE DRUB RUSH」のような文字列が表示される。これがワンタイムパスワードだ。opieloginのパスワードプロンプトにこの文字列を入力し、ログインできればワンタイムパスワードは有効である。

■telnetdでopieloginを使う

最後に、telnetdでのログインに使うプログラムをデフォルトの/bin/loginから/usr/bin/opieloginに変更する。

inetdを使っている場合は、/etc/inetd.confのtelnetの行を**リスト3**のように変更する。

xinetdなら、/etc/xinetd.d/telnetに次の設定を書き加え、再起動する。

```
server_args = -L /usr/bin/opielogin
```

これで、telnetで接続すると、パスワード入力時のプロンプトがopieloginのものに変っているはずだ。ここでワンタイムパスワードを入力すればログインできる。安全なネットワークからのログインであれば、このプロンプトに通常のパスワードを入力してログインすることもできる。

なお、ワンタイムパスワードの計算は適当なパソコンを使うか、あるいは携帯用にopiekeyコマンドを使ってまとめて計算させたものをプリントアウトしておいてもいい。まとめて計算させる場合は、opiekeyコマンドに「-n 100」などのオプションを指定する。-nで指定した数だけ、ワンタイムパスワードのリストを生成するので、プリンタなどで印刷しておけばいい。

このほかにも、opieにはいろいろな使い方があるが、詳しい使い方はopieのマニュアルを参照していただきたい。

■暗号化telnet

telnetではすべての通信が平文で行われるが、重要なデータを扱う場合には平文での通信は避けたい。そのため、telnetには暗号化機能が含まれているのだが、暗号技術輸出規制によって米国以外では暗号化機能を含むtelnetが利用できない。現在では米国の規制が事実上、解除されているのだが、いずれにしても暗号化機能を有効にしたtelnetパッケージが手軽に入手できるわけではないので、別のソフトウェアを用意する必要がある。

telnetを暗号化したソフトウェアには、いくつかの種類がある。たとえば、SSLを使った暗号化通信が可能なSSLtelnetと呼ばれるソフトウェアや、汎用の認証システムのKerberosに付属する暗号機能付きのtelnetなどである。また、最近はtelnetというよりrloginの仲間だが、SSHと呼ばれる暗号化通信のソフトウェアパッケージが広く使われている。

どの方法でも、専用のtelnetクライアントが必要になるなど、緊急時に手近なクライアントを使ってログインするという用途には向かないが、暗号化が必要であれば、多くのディストリビューションに標準添付され、ユーザーも多く、設定も手軽なOpenSSHが便利だろう。

高度なセキュリティを必要とするのであれば、OpenSSHの通信に公開鍵を使うなどの設定をすることが望ましいが、単に平文パスワードを流さず、通信経路を暗号化するというだけであれば、パッケージをインストールするだけでよい。具体的には、OpenSSHのサーバを起動させたホストに「ssh ホスト名」のコマンドで接続するだけで暗号化された通信セッションが張られるので、セキュリティの心配なく、パスワードプロンプトにパスワードを入力することができる。OpenSSHの詳しい使い方は、マニュアルまたは本誌の特集などを参考にしていただきたい。

画面2　Windows用のワンタイムパスワード計算ソフト

▼リスト3　ワンタイムパスワードを使うためのinetd.confの設定

```
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd in.telnetd -L /usr/bin/opielogin
```

Oracle8iで作るWebサイト入門

前回まではもっぱらSQL＊Plusを使って、SQL文を手動で使ってきました。なんで今どきコマンドラインと思われたかもしれませんが、それがSQL文を理解する一番早い道であるということが今回実感できます。実際にスクリプトを書いてみると、皆さんがSQLをマスターしていることに驚くはずです。

第6回　書籍検索サイトを作る

文：おもてじゅんいち／かざぐるま
Text : Junichi Omote/Kazaguruma

前回、CSVからデータベースに読み込んだ書籍情報を使って、Web検索ページを作ってみましょう。検索の機能は、

- **検索条件を入力する**
- **検索結果を一覧表示する**
- **一覧から選択された書籍の詳細情報を表示する**

という流れにします。まずは検索条件の入力を**画面1**のように、

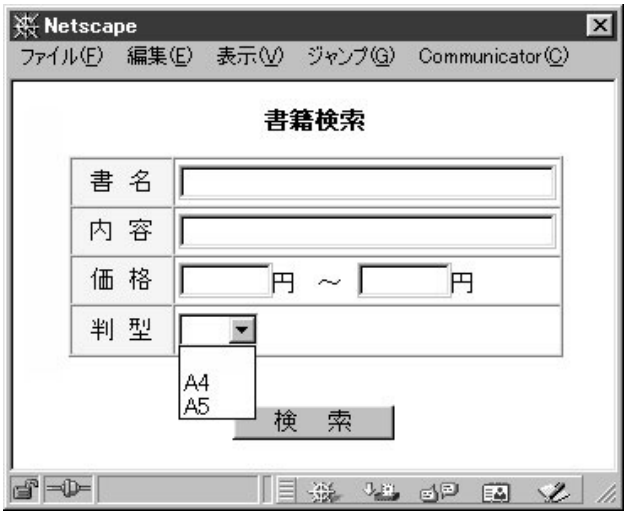
- **書籍タイトルに含まれる語句で検索**
- **内容に含まれる語句で検索**
- **価格の範囲で検索**
- **判型の種類で検索**

ということにします。もちろん希望があればほかの項目を検索の対象にできますが、ここでは、

- **含まれる語句で検索**
- **値の範囲で検索**
- **候補をポップアップメニューで選択して検索**

という異なった3種類の検索方法を行ってみるために、このような検索条件としました。検索を実行するスクリプトは、検索条件入力画面のフォームに入力されたデータを受け取り、データベースに対してSELECT文を実行します。そのときに受け取ったデータで検索が行われるように、SELECT文のWHERE句を適切に記述すればよいことになります。

こういうスクリプトを作成するときは、そのSELECT文の書き方がポイントで、なかなか一発で正しいSELECT文が書けなかったり、検索結果が思うようにならないことがよくあります。そのつどスクリプト内のSQL文を書き直して実行し直すということを繰り返すのも面倒ですから、まずはSQL＊Plusを使って正しいSQL文を決定しておき



画面1　検索条件入力画面

ます。SQL＊Plusなら結果がすぐにわかりますし、簡単に何度でもSQL文をテストすることができます。

■語句の検索

まずは語句の検索です。タイトルと内容の項目について語句の検索を行うのですが、とりあえずはタイトルについて実験してみます。たとえばTITLE列の内容が「はじめて」である行をSELECTするなら、

```
select * from book where title = 'はじめて'
```

となりますね。実行結果は**画面2**です。しかしこれでは「はじめて」というタイトルを検索することになり、「はじめて」という語句を含むものの検索にはなりません。「含む」という検索をさせるには、

```
… where title like '%はじめて%'
```

というように、＝のかわりにlikeという演算子を使います。そして検索語句の前後に「%」が記述されています。この「%」はワイルドカードと呼ばれ、ここに何かの文字があるという意味になります。つまり、'%はじめて%'と書いておくと、

```
はじめてのLinux
生まれてはじめてのOracle
PHPを使うのははじめて
```

などが検索にヒットします。前後に「%」を書いてあるのは、「はじめて」という語句が、先頭、文中、末尾どこにあっても検索ヒットするためのもので、もし'はじめて%'という記述にすると、上記のうちヒットするのは、

```
はじめてのLinux
```

だけになります。「はじめて」のうしろになんらかの文字があるもののみヒットするのです。「PHPを使うのははじめて」というデータは「はじめて」の前にも文字があるため、これもヒットさせるには'はじめて%'ではなく、'%はじめて%'と記述しなければなりません。「はじめて」が先頭にあるものだけ検索したいとか、末尾にある場合だけとかいう場合によって「%」の記述を使い分けるのです。

検索結果は**画面3**のようになりました。ほかにも語句を変えて実験してみてください。CONTENTS列についても同様ですね。

■数値の範囲検索

次は価格の検索です。価格は数値ですが、同じようにwhereを使った検索ができます。**画面4**はある特定の値のものを検索する場合です。

```
where price = 2200
```

数値の場合は、"を付けないことに気をつけてください。

これではまだ範囲の検索にはなっていません。範囲で検索するには<や>を使って上限と下限を記述します。

```
where price >= 2000 and price <= 2500
```

andは「かつ」という意味で、この場合は「2000以上かつ2500以下」という条件になります。検索結果は**画面5**です。正しい結果になっています。

■候補メニューを作る

次はポップアップメニューから候補を選択する検索で

```
SQL> select title from book where title = 'はじめて';

レコードが選択されませんでした。

SQL> select title from book where title='はじめて読む8086';

TITLE
--------------------------------------
はじめて読む8086
```

画面2　完全一致の検索

```
SQL> select title from book where title like '%はじめて%';

TITLE
--------------------------------------
はじめて読む8086
はじめて読むBASIC 改訂新版
はじめて読むC言語
はじめて読むMASM
はじめて読むアセンブラ
はじめて読むマシン語
はじめて読む486
```

画面3　部分一致の検索

す。判型には「A4」や「A5」といった文字が格納されていますから、検索自体は語句の検索でよいのですが、検索条件入力画面にあらかじめ候補を表示してやる必要があります。候補として「A4」「A5」「B5」などをあらかじめ固定して書いておくのもひとつですが、それではちょっと芸がないので、現在データベースに存在する判型のみを候補としてメニューに表示するようにしてみましょう。

ということは、検索条件入力画面を表示するときに前もってデータベースにアクセスして、判型（TYPE列）に今存在するものを取り出しておいて、メニューを作ればよいことになります。TYPE列の内容を取り出すには、

```
select type from book
```

というSELECT文になりますが、これだと**画面6**のように同じ「A4」でもデータの数だけ表示されてしまいます。そこで、「A4」とか「A5」とかをまとめて1回ずつ表示するために、

```
select type from book group by type
```

というように、group by 句を使うとgroup byの後に記述された列の内容について同じものはひとまとめにすることができます。結果が**画面7**で、この結果をメニューにしておけば、今存在する種類のデータだけを選択することができるようになります。

リスト1が検索条件入力画面のソースです。メニューを作るためにいったんデータベースにアクセスする必要がありますから、PHPスクリプトにします。

```
SQL> select title,price from book where price =
2200;

TITLE                                PRICE
------------------------------ ----------
標準C++の基礎知識                      2200
```

画面4　価格での検索

```
SQL> select title,price from book where price >=
2000 and price <= 2500;

TITLE                                PRICE
------------------------------ ----------
標準C++の基礎知識                      2200
はじめて読む486                        2427
```

画面5　価格での検索（範囲）

①がメニューを作っている部分で、TYPE列の内容を読み出して、<option …>タグに加工しています。①のスクリプト部分の実行結果は、

```
<option value='A5'>A5
<option value='A4'>A4
```

になります。

検索結果一覧画面

検索結果一覧画面は、検索条件入力画面のフォームデータを受け取って検索を実行し、該当したデータを一覧表示します（**画面8**、**リスト2**）。

検索条件が、タイトル、内容、価格、判型と複数あるので、複数の条件が入力されたときには、それらの条件をAND（かつ）でつなぐことにします。たとえば、タイトルに「はじめて」が含まれていて、2000円以上で判型がA4のものを検索する場合は、

```
where title like '%はじめて%' and price >= 2000 and
type = 'A4'
```

というwhere句になればよいのです。

```
SQL> select type from book;

TYPE
----------------------------------------
A5
A5
A5
A5
A5
A5
A5
A4
A4
A5
```

画面6　TYPE列を表示

```
SQL> select type from book group by type;

TYPE
----------------------------------------
A4
A5
```

画面7　判型の候補を読み込む

▼リスト1　検索条件入力スクリプト（form.php3）

```
<html>
<head>
<title></title>
</head>
<body>

<br>
<center>
<b>書籍検索</b><br>
<br>

<form action="list.php3" method="post">
<table border="1" cellpadding="2"
cellspacing="0">
<tr>
<td bgcolor="#f5f5f5">　書　名　</td>
<td><input type=text size=30 name="f_title"></td>
</tr>

<tr>
<td bgcolor="#f5f5f5">　内　容　</td>
<td><input type=text size=30
name="f_contents"></td>
</tr>

<tr>
<td bgcolor="#f5f5f5">　価　格　</td>
<td><input type=text size=6 name="f_price1">円
  ～　<input type=text size=6 name="f_price2">円
</td>
</tr>

<tr>
<td bgcolor="#f5f5f5">　判　型　</td>
<td>
<select name="f_type">
<option value="">
<?php
  $conn = OCIlogon("ascii","linuxmag","orcl");

  $stmt = OCIparse($conn,"select type from book
group by type");
  OCIdefinebyname($stmt,"TYPE",&$type);
  OCIexecute($stmt);
  while(OCIFetch($stmt)) {
    print("<option value='$type'>$type\n");
  }

  OCIlogoff($conn);
?>
</select>
</td>
</tr>

</table>

<br>
<br>
<input type="submit" value="　検　索　">
</form>
</center>

</body>
</html>
```

①

リスト2の①から⑦の部分でSQL文を作っています。SQL文はいったん$sqlという変数に文字列として作っておいて、

```
OCIparse($conn,$sql);
```

というようにデータベースに渡します。SQL文が条件によって複雑で長くなりそうな場合はこのような方法をとりましょう。

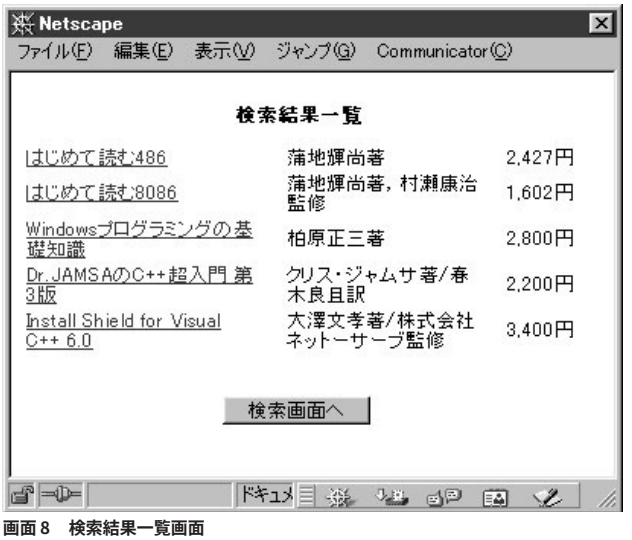
①はselect文の固定部分で、②～⑥がそれぞれの列に対しての条件です。それぞれは条件入力フォームに入力があった場合にだけwhere句に記述するものとし、条件が複数並ぶ場合はandでつないでいます。この部分の処理を要約すると、

- **それぞれの条件項目に入力がある場合は以下を実行**
- **その時点でのSQL文の末尾がwhereでなかったら（すでに何かの条件式が書かれていれば）、andを付け足す**
- **条件式を付け足す**

という処理を項目ごとに行っています。

SQL文の末尾にwhereがあるかどうかの判断は正規表現を使って行っています。正規表現を使用する場合の関数はereg()です。正規表現についてはここでは詳しく説明できませんので、他の解説書か本誌併載の「**Perlスクリプト入門**」を参照してください。

すべての条件の記述が終わったら、⑦で一応できあがったSQL文をチェックします。ここでは、もし何の条件も



画面8　検索結果一覧画面

▼リスト2　検索結果一覧表示スクリプト（list.php3）

```
<?php
    $conn = OCIlogon("ascii","linuxmag","orcl");
?>
<html>
<head>
<title></title>
</head>
<body>
<br>
<center>
<b>検索結果一覧</b><br>
<br>
<table border="0" cellpadding="2" cellspacing="0">

<?php
        $sql = "select id,title,author,price from book where";          ①
        if ($f_title) {
                $sql .= " title like '%$f_title%'";                     ②
        }
        if ($f_contents) {
                if (!ereg("where$",$sql))       $sql .= " and";         ③
                $sql .= " contents like '%$f_contents%'";
        }
        if ($f_price1) {
                if (!ereg("where$",$sql))       $sql .= " and";         ④
                $sql .= " price >= $f_price1";
        }
        if ($f_price2) {
                if (!ereg("where$",$sql))       $sql .= " and";         ⑤
                $sql .= " price <= $f_price2";
        }
        if ($f_type) {
                if (!ereg("where$",$sql))       $sql .= " and";         ⑥
                $sql .= " type = '$f_type'";
        }
        if (ereg("where$",$sql))        $sql = ereg_replace("where$","",$sql);   ⑦
        $sql .= " order by id";

    $stmt = OCIparse($conn,$sql);
        OCIdefinebyname($stmt,"ID",&$id);
        OCIdefinebyname($stmt,"TITLE",&$title);
        OCIdefinebyname($stmt,"AUTHOR",&$author);
        OCIdefinebyname($stmt,"PRICE",&$price);
        OCIexecute($stmt);
        while(OCIFetch($stmt)) {                                        ⑨
                print("<tr>\n");
                print("<td><a href='disp.php3?f_id=$id'>$title</a></td><td width=20></td>\n");
                print("<td>$author</td><td width=20></td>\n");
                $p = ereg_replace("([0-9][0-9][0-9])$",",\\1",$price);  ⑧
                print("<td>$p 円</td>\n");
                print("</tr>\n");
        }
?>

</table>
<form>
<input type="button" value="検索画面へ" onclick="JavaScript:location.href='form.php3'">
</form>
</center>
</body>
</html>
<?php
    OCIlogoff($conn);
?>
```

入力されていなければ、SQL文は①の状態のままなので、末尾がwhereになっており、このままではSQL文がエラーになるので、末尾のwhereを消去します。

あとは、このSQL（SELECT）文を実行して、検索されたデータを表示するだけです。一覧表示にはタイトル、著者名、価格だけを表示しています。

このとき、データベース内の価格データは数値だけが格納されているのですが、表示するときには3桁ごとに「,」（カンマ）で区切られていたほうが見やすいので加工を施します（⑧）。これも正規表現の置換で行っており、末尾から数えて3桁分の数字の直前に「,」を入れています。

一覧表示画面としての処理はこれで十分ですが、このあと、書籍名をクリックしたときにその詳細情報を表示しなければなりません。そのために、一覧のタイトル表示の部分にリンクを張って、次の詳細情報表示画面に移行するようにします。リンク先は詳細情報表示スクリプトのdisp.php3ですが、クリックされた書籍名によって詳細情報の内容は変わらなければならないので、リンクするときに、

```
disp.php3?f_id=11
```

といったように、URLの後ろにクエリー文字列としてID番号を送るようにします。このIDを読みとってそれに対応する書籍の詳細情報を表示すればよいのです。

このリンクを作成している部分が**リスト2**の⑨です。一覧表示の各行に対して、それぞれのIDがクエリー文字列として付加されるようになっています（**画面9**は、各データに付けられたIDを示しています）。一覧表示のブラウザのソースをのぞいてみれば一目瞭然で、**リスト3**のようになっています。

```
SQL> select id,title from book;

        ID TITLE
---------- ------------------------------
        12 はじめて読む8086
        13 はじめて読むBASIC 改訂新版
        14 はじめて読むC言語
        15 はじめて読むMASM
        16 はじめて読むアセンブラ
        17 はじめて読むマシン語
        18 Windowsプログラミングの基礎知識
        19 標準C++の基礎知識
        20 標準C++の基礎知識 実践編
        11 はじめて読む486
```

画面9　IDを表示

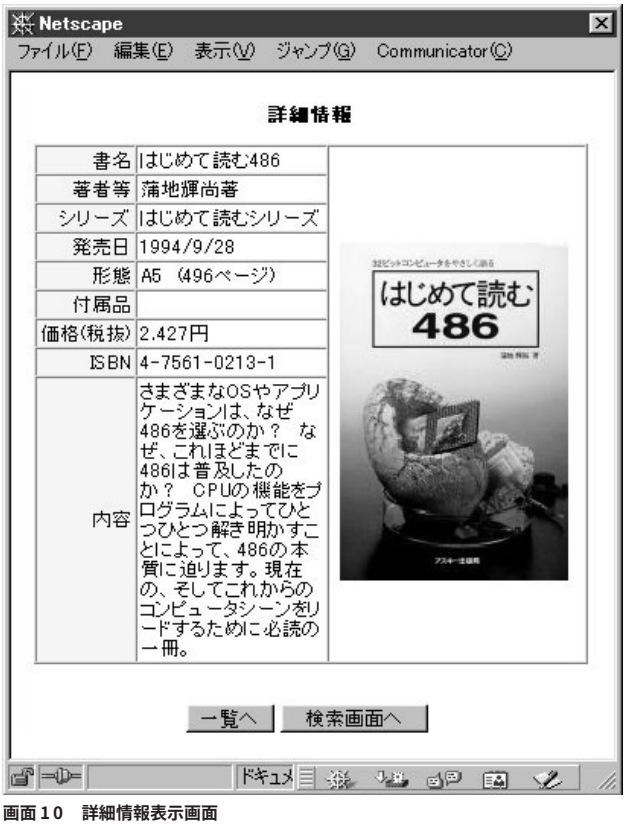
▼リスト3　一覧表示の実行結果（ブラウザのソース）

```
 ⋮
<td><a href='disp.php3?f_id=11'>はじめて読む
486</a></td> ……
 ⋮
<td><a href='disp.php3?f_id=12'>はじめて読む
8086</a></td> ……
 ⋮
<td><a href='disp.php3?f_id=19'>標準C++の基礎知識
</a></td> ……
 ⋮
```

詳細情報表示画面

詳細情報表示画面は**画面10**のようになります。ここでは、一覧画面からクエリー文字列として受け取ったIDに該当する書籍のデータをデータベースから読み込んで表示するだけです（**リスト4**）。

いくつかの表示で工夫している部分があります。①では、一覧表示のときと同じように、価格を3桁区切りで表示しています。②は発売年月日の表示ですが、Oracle8iの日付データはyy-mm-ddという形になっているために（なぜかいまどき年が2桁）、このままでは見にくいので形を変えています。まずyy-mm-ddという文字列を年、月、日のデータに分割します。



画面10　詳細情報表示画面

▼リスト4　詳細表示スクリプト（disp.php3）

```
<?php
    $conn = OCIlogon("ascii","linuxmag","orcl");
?>
<html>
<head><title></title></head>
<body>
<br>
<center>
<b>詳細情報</b><br>
<br>
<?php
    $stmt = OCIparse($conn,"select * from book where id=$f_id");
        OCIdefinebyname($stmt,"TITLE",&$title);
        OCIdefinebyname($stmt,"AUTHOR",&$author);
        OCIdefinebyname($stmt,"SERIES",&$series);
        OCIdefinebyname($stmt,"SELDATE",&$seldate);
        OCIdefinebyname($stmt,"TYPE",&$type);
        OCIdefinebyname($stmt,"APPENDIX",&$appendix);
        OCIdefinebyname($stmt,"PRICE",&$price);
        OCIdefinebyname($stmt,"ISBN",&$isbn);
        OCIdefinebyname($stmt,"CONTENTS",&$contents);
        OCIexecute($stmt);
        OCIFetch($stmt);

        $p = ereg_replace("([0-9][0-9][0-9])$",",\\1",$price);          ①
        $d = explode("-",$seldate);
        if ( $d[0] < 70 )              $d[0] += 2000;                   ②
        else                           $d[0] += 1900;
?>
<table border="1" cellpadding="2" cellspacing="0">
<tr>
<td bgcolor="#f5f5f5" align="right" valign="top" nowrap>書名</td>
<td><?php print $title?></td>
<td rowspan=9 width=170 align=center valign=center>
<br>
<img src="photo/<?php print $isbn?>.jpg" width=150 height=222>          ③
<br>
</td>
</tr>
<tr><td bgcolor="#f5f5f5" align="right" nowrap>著者等</td>
<td><?php print $author?></td></tr>
<tr><td bgcolor="#f5f5f5" align="right" nowrap>シリーズ</td>
<td><?php print $series?></td></tr>
<tr><td bgcolor="#f5f5f5" align="right" nowrap>発売日</td>
<td><?php print "$d[0]/$d[1]/$d[2]"?></td></tr>                         ④
<tr><td bgcolor="#f5f5f5" align="right" nowrap>形態</td>
<td><?php print $type?></td></tr>
<tr><td bgcolor="#f5f5f5" align="right" nowrap>付属品</td>
<td><?php print $appendix?>  </td></tr>
<tr><td bgcolor="#f5f5f5" align="right" nowrap>価格(税抜)</td>
<td><?php print $p?>円</td></tr>
<tr><td bgcolor="#f5f5f5" align="right" nowrap>ISBN</td>
<td><?php print $isbn?></td></tr>
<tr><td bgcolor="#f5f5f5" align="right" nowrap>内容</td>
<td><?php print $contents?> </td></tr>
</table>
<?php
    OCIlogoff($conn);
?>
<form>
<input type="button" value="一覧へ" onclick="JavaScript:history.back()">
<input type="button" value="検索画面へ" onclick="JavaScript:location.href='form.php3'">
</form>
</center>
</body>
</html>
```

```
$d = explode("-",$seldate);
```

は、「-」を区切りとして$seldateに格納されている文字列を分割し、それぞれを配列$dに格納するという処理です。たとえば、$seldateが"99/10/12"の場合、

```
$d[0] = "99"
$d[1] = "10"
$d[2] = "12"
```

という結果になります。そして99や00となっている年表示を4桁に直します。ここでは70未満を2000年代、70以上を1900年代と考えて上位2桁を付け加えています（Oracle8iの日付型が2桁なのでとりあえずは仕方ないのですが、この処理では2070年に問題が発生します……）。

これを④の部分で、

```
$d[0]/$d[1]/$d[2]
```

とprintしています。これで、「1999/10/12」と表示されるようになります。

③は写真データを表示する部分です。各書籍データでユニークな情報としてISBNコードがありますので、ISBNコードをファイル名にした画像データを用意することとし、③の部分で<img …>タグを発生させています。

画面のレイアウトはすべてHTMLのタグで作られていますから、自由に作ることができます。データベースからのデータの読み出しと加工だけはこのままで、レイアウトをいろいろと別なものに変えてみてください。

スクリプトのデバッグ

スクリプトもSQL文もなかなか一発で動いてくれることはないものです。ましてSQL文をいろんな変数の合成で作る場合（がほとんどですが）ならなおさら、できあがったSQL文が正しいかどうか、想像しながら進めて行くことになります。そんな場合は、スクリプトをじっとながめながら悩むよりも、できたSQL文を表示させて確認するほうが一目瞭然です。一覧表示のスクリプトではSQL文をいったん$sqlに格納していますから、OCIparseを実行する前に、

```
print "[$sql]";
```

のような行を記述して、SQL文を画面に表示させてしまうのです。そうすれば思いどおりのSQL文ができているかどうかが確認できます。エラーが起こった場合だけでなく、複雑なSQL文を作ったときは必ず表示させて確認するようにしてください。

PHPはエラーが発生した場合、実行結果のブラウザ画面上に、結構詳しいエラーメッセージと箇所を指摘してくれますから助かりますが、それはあくまでも文法上の間違いなど、PHPのパーサ（構文解析部）にわかるエラーしか検出できません。文法上に間違いはないけれども、思った結果にならないという場合は自分で調べる必要があります。

たとえば、先ほどの一覧画面から詳細情報表示画面にリンクするときのID番号渡しなど、正しいID番号がクエリー文字列となってURLについているかどうかが問題になるのですが、クエリー文字列があろうがなかろうが、またID番号が正しいかどうかなどは、PHPのパーサにも、Webサーバにも、ブラウザにもわかるはずがありません。

そんな場合、つまり明確なエラーにはなっていないけれども、思ったとおりの結果になっていないときは、ブラウザのソースを見るのが一番早道です。もちろん原因はPHPのスクリプトやSQL文にあるはずなのですが、スクリプトの状態で見ても実行時に何が起こっているかを考えるのは難しいものです。

Webアプリケーションでのスクリプト実行の流れは**図1**のようになっています。これらすべてが正しく連携してはじめて思いどおりの結果が得られるのですから、不具合があるときはスクリプトだけをながめるのではなく、SQL文や実行結果のHTMLソースのほうを確認する習慣をつけましょう。たいていはそのほうが早く原因を発見できます。

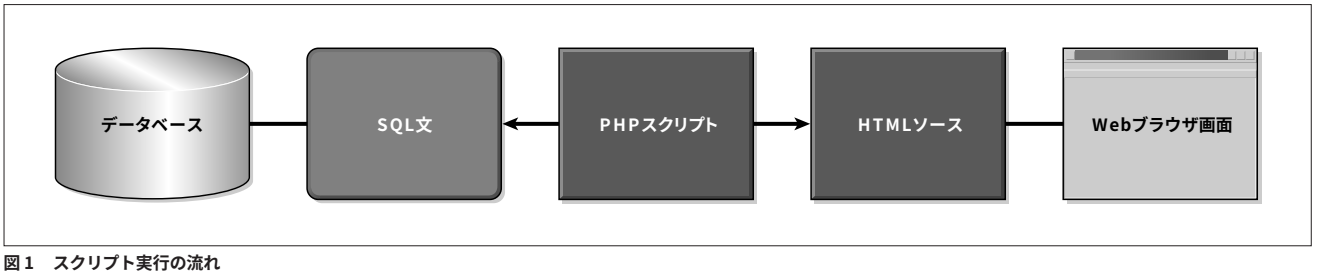


図1　スクリプト実行の流れ

Perlスクリプト入門
すぐにできる実践Perl

CGIで実現できる機能は、前回紹介した掲示板以外にもたくさんあります。ちょっとしたPerlスクリプトを作成するだけで、高機能なWebページが作成できるのもPerlの魅力です。

第10回　CGI Tips

文：おもてじゅんいち／かざぐるま
Text : Junich Omote/Kazaguruma

前回まではフォーム入力の例と掲示板CGIを作ってみましたが、今回はこれからCGIを作っていくときに知っていれば便利なCGIのワザを紹介します。いずれもよく目にする機能ばかりです。機能満載のWebページも、このようなちょっとしたCGIの積み重ねになっているのです。

チャット――自動更新するページ

前回作った掲示板と同じくらい有名になったものにチャットがあります。チャットはあたかも会話をしているかのように、どんどん発言していけるものですが、基本的には発言を順次表示するという点で掲示板によく似たスクリプトとなっています。

画面1　チャットCGI

掲示板との違いは、複数の発言者がどんどん発言できるために、ボタンをクリックしたりしなくても次々と発言が表示されていくところにあります。また掲示板とは違って、新しい発言は前の発言の下に表示されます。

チャットでは、入力するものはハンドル名と発言内容だけになります（**画面1**）。一般的にはハンドル名を決めてログイン（入室）するなどの仕組みを用意して、自動的にハンドル名が付けられるのですが、ここではとりあえず実験のために1画面で収まるようにしたために、毎回ハンドル名を入力する必要があります。このあたりは実用化に向けて皆さんで改良してください。

チャットCGIのポイントは発言の表示が自動的に更新されることにあります。そこで発言の表示画面を自動更新する仕掛けを作るのですが（ここでは5秒ごと）、自分の発言を入力するフォームまで更新されてしまうと、せっかく途中まで入力した文字が消えてしまいます。このため、発言を入力するフォーム部分と発言の表示部分とを別のフレームにして、自動更新は発言表示部だけが行われるようにします。

チャットCGIのために必要なファイル構成は**表1**のようになります。

chat.html、chat_form.cgi、chat_disp.cgiの3ファイル

でチャットの1画面を構成することになります。chat.txtは発言を記憶しているデータファイルです。ですから、chat_form.cgi、chat_disp.cgiには実行可のパーミッションを、chat.txtには書き込み可のパーミッションを設定しておいてください。また、フォーム処理を使いますので、cgi-lib.plも同じディレクトリにコピーしておいてください。

リスト1～3がそれぞれのソースです。chat.html（**リスト1**）はフレームを設定するHTMLだけのファイルですから、わからない方は<frameset……>タグの使い方をマスターしておいてください。

chat_form.cgi（**リスト2**）は、前回の掲示板CGIをベースにしています。ここでは表示する機能は必要ありませんので、フォームに入力されたデータをファイルchat.txtに書き込む処理を行っています。データ項目数が減っただけで、掲示板と異なる部分は特にありません。

chat_disp.cgi（**リスト3**）も、基本は前回の掲示板と同じで、むしろ古い発言から表示すればよいため、掲示板よりも簡単になっています。ただしそのままでは自動更新表示を行ってくれないため、HTMLの<head>内に、①の

▼リスト3　チャットCGI表示部（chat_disp.cgi）

```
#!/usr/bin/perl

require 'cgi-lib.pl';
&ReadParse;

print <<'EOD1';
Content-type:text/html

<html>
<head>
<META HTTP-EQUIV="Refresh"
          CONTENT="5; URL=chat_disp.cgi">          ①
</head>
<body>
EOD1

    # 発言の表示
    open(IN,"chat.txt");

    while(<IN>) {
        ($handle,$mes) = split(/,/,$_);
        print "<b>$handle></b>$mes<br>";
    }

    close(IN);

print <<"EOD2";
</body>
</html>
EOD2
```

ファイル名	内容
chat.html	フレーム記述
chat_form.cgi	新規発言フォーム部分
chat_disp.cgi	発言表示部分
chat.txt	発言データファイル
cgi-lib.pl	ライブラリ

表1　チャットCGIのファイル構成

▼リスト1　チャットCGIフレーム部（chat.html）

```
<html>
<head>
<frameset rows="140,*">
        <frame src="chat_form.cgi">
        <frame src="chat_disp.cgi">
</frameset>
</head>
</html>
```

▼リスト2　チャットCGIフォーム部（chat_form.cgi）

```
#!/usr/bin/perl

require 'cgi-lib.pl';
&ReadParse;

print <<'EOD1';
Content-type:text/html

<html>
<head>
</head>
<body>
<h2>My Chat</h2>
<b>新規発言</b><br>
<form method="post" action="chat_form.cgi">
ハンドル名
<input type="text" size=10 name="handle"><br>
<input type="text" size=34 name="mes">
<input type="submit" value="発言する">
</form>
EOD1

    # 新規発言の登録
    $handle = $in{'handle'};
    $mes = $in{'mes'};

    if($mes) {
        open(OUT,">>chat.txt");
        print OUT "$handle,$mes¥n";
        close(OUT);
    }

print <<"EOD2";
</body>
</html>
EOD2
```

文を記述します。

```
<META HTTP-EQUIV="Refresh"
      CONTENT="5; URL=chat_disp.cgi">
```

これはPerlではなくHTMLの機能なのですが、「5秒後にchat_disp.cgiというURLにジャンプする」という意味です。ここでは自分自身にジャンプするということになりますから、ジャンプして読み込んだファイル（自分自身）にも同じ記述があり、5秒後に再びジャンプする……といった動作を繰り返します。

つまり、chat_disp.cgiは5秒ごとに再読み込みされ続けるのですが、その間に別のユーザーによって発言が入力され、chat.txtにその発言が追加されていれば、chat_disp.cgiの表示も更新されるという仕組みです。

<META HTTP-EQUIV…… >タグは、引越ししたWebページで「5秒後に自動的にジャンプします」というようなメッセージとともによく使われます。「CONTENT="」のあとの数字を変更すればジャンプするまでの時間を設定できます。ジャンプするまでの時間を短い間隔にすれば更新がリアルタイムに近くなりますが、あまり短い間隔にすると、そのたびにサーバへのアクセスが起こり、サーバや回線への負荷が大きくなることに気をつけてください。

なお、このチャットCGIでは、自分の発言を登録したときでも発言の表示部は5秒ごとに更新されますから、しばらく待たなければ自分の発言が表示されません。もし、自分の発言のときだけすぐに更新したければ、JavaScriptを使って、上のフレームから下のフレームを再読み込みさせるといったスクリプトを書かなければなりません。

メールを送信する

以前少し紹介しましたが、CGIからメールを送信することもできます。CGIからメールを送信する場合は、メールサーバであるsendmailを使います。ですから、このCGIを動作させるにはsendmailがインストールされている必要があります。

sendmailは通常、SMTPデーモンとして動きますが、コマンドラインから起動して手動でメールを送信することもできるので、ここではPerlからパイプを使って送信先やタイトル、本文などの必要な情報をsendmailに送ります。

それではフォーム入力されたデータをメールで送信するCGIを考えてみましょう。入力フォームとして、以前使ったフォームを流用します（**リスト4**）。今回はこれにEメールアドレスの入力欄を追加します（**リスト4**の①）。

このフォームデータを受け取ってメールを送信するCGIが**リスト5**になります。前半のフォームからデータを受け取ってそれぞれ加工する部分は以前と同じです。「#メールの送信」というコメント部分からの記述がメールを送信するためのものです。コメントのすぐ下の、

```
$mailpath = '/usr/sbin/sendmail';
```

はsendmailのパスを記述します。もしパスがわからなければ、whichかwhereisコマンドを使って、

```
$ whereis sendmail
```

としてみてください。これでsendmailのフルパスが表示されます。

その次の行にある、

```
$toaddr = 'dareka@domain.com';
```

は送り先のメールアドレスです。ここでの例のようにフォームからメールを送る場合は、たいていそのページの運営者側にメールが送られてくるものでしょうから、ここには受け取りたい自分の（自社の）メールアドレスを記述しておきます。

この$mailpathと$toaddrを使って、

```
open MAIL,"| $mailpath $toaddr";
```

というように、あたかもファイルをオープンするような記述があります。ここでは、本来ファイル名が入るべきところに、「|」というパイプを表す記号があります。これはあとに続く変数を展開すると、

```
| /usr/sbin/sendmail dareka@domain.com
```

となり、コマンドラインでパイプを使う場合の後半部分と同じような記述になるのです。そして、パイプの前半の代わりにMAILというファイルハンドルに出力することによって、このあとに続くprint文の出力データがsendmailに流れて行くことになります。

あとはフォームから受け取った各データをMAILハンド

ルに出力すればよいので、

```
print MAIL "Subject: <<mail_form.cgi>>¥n";
print MAIL "From: $in{'email'}¥n";
print MAIL "¥n";
print MAIL "登録情報¥n-----¥n";
print MAIL "$addr¥n";
print MAIL "〒$in{'zipcode'}¥n";
  …
close MAIL;
```

とprint文を記述して、最後にMAILハンドルをクローズします。

ここで、print文の3行目までは決まりがあり、「Subject:」に続けてメールのタイトルを、「From:」に続けて送信元のメールアドレスを出力します。これにより、受け取ったメールの送信元とタイトルが正しく認識されます。これらメールヘッダ部分と本文とはかならず空行で区切らなければならないので、3行目で改行のみを出力しています。このあと、MAILハンドルをクローズするまでがメールの本文データになります。

この決まりさえ守っていれば、データを加工したり、条件判断やループをしたりしてもprint文の出力がすべてメールで送信できるようになります。

リスト5のCGIから送られてくるメールの内容は、**リスト6**のようになります。この結果とスクリプトを見比べてみてください。

CGIでのメール送信は、この例のようにアンケートや予約フォームのデータを送ったり、フォームでなくてもCGIの処理によってその結果を通知メールとして送ったり、逆にページを見ているユーザーに向けて会員登録の結果を送ったりと、いろいろな利用方法が考えられます。

▼リスト4　メール送信フォーム（mail.html）

```
<html>
<head></head>
<body>
<form action="mail_form.cgi">
<input type=radio name="addr" value="com">会社
<input type=radio name="addr" value="home">自宅
<br><br>
郵便番号<input type=text name="zipcode">
<br><br>
都道府県
<select name="pref">
   <option value="0">茨城
   <option value="1">群馬
   <option value="2">山梨
   <option value="3">埼玉
   <option value="4">東京
   <option value="5">千葉
   <option value="6">神奈川
</select>
<br><br>
コメント<br>
<textarea name="comment">
</textarea>
<br><br>
E-mail <input type=text name="email"><br>  ①
<input type=checkbox name="mailreq">
メール配信を希望する
<br><br>
<input type=submit value="送　　信">
</form>
</body>
</html>
```

ブラウザ情報を取得する――環境変数

CGIスクリプトが動くときには、実はWebサーバからかなり多くの情報が提供されています。

まずはどんな情報が提供されているのかを表示してみましょう。それらの情報は環境変数によって渡され、Perlでは連想配列「%ENV」に格納されています。**リスト7**が環境変数とその内容を一覧表示するCGIスクリプトです。

実行結果は**画面2**のようになりますが、サーバとブラウザの環境によって内容は変わりますので、皆さんも一度試してみてください。

CGIで使えそうな環境変数は、**表2**のようなものがあります。

HTTP_USER_AGENTなどは、ブラウザの種類を判別するためによく使われます。特に、最近では携帯電話などからのアクセスも考えられるので、携帯電話用のWebページとPC用のWebページを自動的に振り分けることができます。

ただしこれらの環境変数はユーザー側の情報流出とも言えるものなので、ブラウザや端末の環境によってはデータが設定されていない場合もあります。実際に使う場合は、できるだけ多くの環境でテストしておく必要があるでしょう。

アクセスカウンタ――SSIを使う

CGIはフォームのaction先として記述するか、直接CGIファイルへのリンクを設定して動作するものです。つまり、ユーザーから見れば、必ず「～.CGI」というファイルにア

クセスしなければ動作させられないものでした。

Web上でよく見かけるアクセスカウンタですが、そのWebページがアクセスされるたびに1ずつカウントを増やしていって、そのカウント数をサーバ側で記憶しておくという仕組みは、CGIを使えば簡単にできそうです。しかし、通常のアクセスカウンタはトップページのindex.htmlファイルなどで稼動しています。index.htmlはCGIファイルではありませんから、スクリプトを記述しておくことはできません。

このような場合にSSI（Server Side Include）が使われます。これは、通常のHTMLファイル内にCGIを実行せよという記述ができるもので、ファイル自体はCGIではありませんが、HTMLファイルの一部分にCGIの実行結果を反映させることができます。つまり、アクセスカウンタを例にとれば、カウンタの部分にだけCGIの出力が反映されれば、そのほかの部分は通常のHTMLでよいのです。

SSIの記述例は**リスト8**のようになります。CGIを部分的に起動させるSSIは、

```
<!--#exec cgi="./counter.cgi"-->
```

という、HTMLのコメント文の中に記述します。

SSIにはそのほかに「#exec cmd」など、OSのコマンドを実行させることができるものもありますが、セキュリティの面から考えるとこれは非常に危険なことなので、できる限り「#exec cgi」にとどめておくことをお勧めします。

SSIで実行されるCGIですが、これは通常のCGIと同じような作り方になります。ただ、SSIの記述された部分に反映されるデータだけを出力すればよいので、余計なタグなどを出力する必要はありません。

リスト9はアクセスカウンタ用のCGIスクリプトです。スクリプト自体は簡単で、サーバ側に用意しておいたカウント数記憶ファイル（counter.txt）から数字を読み出し、1を加算してまた元のファイルに書き込みます。そしてその数字のみをprint文で出力しています。

実行結果は**画面3**のようになり、あとの装飾は**リスト8**のHTMLファイル側で自由に行ってください。

グラフィカルなアクセスカウンタ

最近のアクセスカウンタは、ほとんどが凝ったグラフィックスの数字を使ったものになっています。ということで、時代に遅れないようにグラフィカルなアクセスカウンタに挑戦してみましょう（**画面4**）。

とはいっても、アクセス数をカウントして記憶しておくという仕組みそのものは同じで、ようは表示するときにグラフィックスを使えばよいだけです。ですから、文字で数字を表示する代わりに、0から9までの1桁の画像を用意しておいて表示することになります。ここでは、それぞれのファイル名を0.gif、1.gif……とします（**図**）。もちろん、どんな画像にするかは皆さん次第です。自分でデザインするのもよいでしょうし、フリーの素材集を提供しているサ

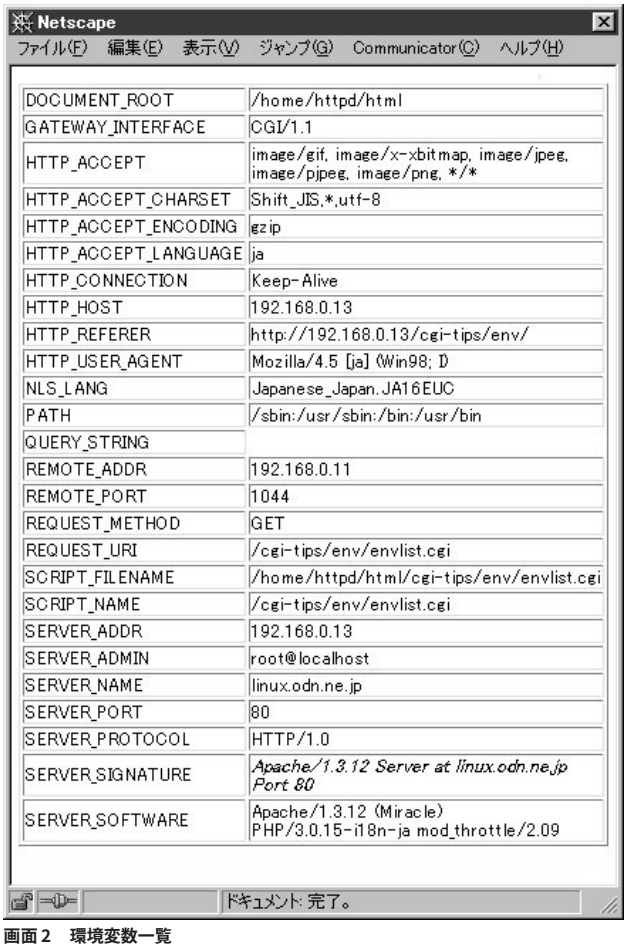
▼リスト5　メール送信CGI（mail_form.cgi）

```
#!/usr/bin/perl

require 'cgi-lib.pl';
&ReadParse;

if ($in{'addr'} eq 'com') { $addr = "会社"; }
else                      { $addr = "自宅"; }

@plist = ('茨城','群馬','山梨','埼玉'
                ,'東京','千葉','神奈川');
$pno = $in{'pref'};

$comm = $in{'comment'};
$comm =~ s/¥n/<br>/g;

# メールの送信
$mailpath = '/usr/sbin/sendmail';
$toaddr = 'dareka@domain.com';
open MAIL,"| $mailpath $toaddr";
        print MAIL "Subject:
<<mail_form.cgi>>¥n";
        print MAIL "From: $in{'email'}¥n";
        print MAIL "¥n";
        print MAIL "登録情報¥n-----¥n";
        print MAIL "$addr¥n";
        print MAIL "〒$in{'zipcode'}¥n";
        print MAIL "$plist[$pno]¥n";
        print MAIL "コメント：$comm¥n";

        if ($in{'mailreq'}) {
          print MAIL "メール配信：希望する¥n";
        }
        else {
          print MAIL "メール配信：希望しない¥n";
        }
close MAIL;

print <<'EOD1';
Content-type:text/html

<html><head></head><body>
情報を送信しました。
</body></html>
EOD1
```

イトも結構あります。

それではCGIスクリプトを考える前に、画像のカウンタは最終的にどんなHTMLになればよいかを考えてみましょう。カウンタを6桁で表示するとすると、2145という表示は、

```
<img src="0.gif">
```



DOCUMENT_ROOT	/home/httpd/html
GATEWAY_INTERFACE	CGI/1.1
HTTP_ACCEPT	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
HTTP_ACCEPT_CHARSET	Shift_JIS,*,utf-8
HTTP_ACCEPT_ENCODING	gzip
HTTP_ACCEPT_LANGUAGE	ja
HTTP_CONNECTION	Keep-Alive
HTTP_HOST	192.168.0.13
HTTP_REFERER	http://192.168.0.13/cgi-tips/env/
HTTP_USER_AGENT	Mozilla/4.5 [ja] (Win98; I)
NLS_LANG	Japanese_Japan.JA16EUC
PATH	/sbin:/usr/sbin:/bin:/usr/bin
QUERY_STRING	
REMOTE_ADDR	192.168.0.11
REMOTE_PORT	1044
REQUEST_METHOD	GET
REQUEST_URI	/cgi-tips/env/envlist.cgi
SCRIPT_FILENAME	/home/httpd/html/cgi-tips/env/envlist.cgi
SCRIPT_NAME	/cgi-tips/env/envlist.cgi
SERVER_ADDR	192.168.0.13
SERVER_ADMIN	root@localhost
SERVER_NAME	linux.odn.ne.jp
SERVER_PORT	80
SERVER_PROTOCOL	HTTP/1.0
SERVER_SIGNATURE	*Apache/1.3.12 Server at linux.odn.ne.jp Port 80*
SERVER_SOFTWARE	Apache/1.3.12 (Miracle) PHP/3.0.15-i18n-ja mod_throttle/2.09

画面2　環境変数一覧

```
<img src="0.gif">
<img src="2.gif">
<img src="1.gif">
<img src="4.gif">
<img src="5.gif">
```

というタグが並べばよいだけです。つまり、6桁に満たない部分に0を付けて、それを6つの数字に分割して、それぞれを<img……>タグで囲めばよいということです。

そのスクリプトが**リスト10**の最後の4行になります。sprintfは前回も出てきましたが、これで0を付け加えながら6桁にします。次に、foreach()内の、

```
split(//,$s)
```

▼リスト6　chat_form.cgiが送信したメール

```
登録情報
-----
会社
〒234-0001
神奈川
コメント：こんにちは
メール配信：希望する
```

▼リスト7　環境変数を表示するCGI

```
#!/usr/bin/perl
print "Content-type:text/html¥n¥n";
print "<html><head></head><body>¥n";
print "<table border=1>¥n";

foreach (sort keys %ENV) {
  print
"<tr><td>$_</td><td>$ENV{$_}</td></tr>¥n";
}

print "</table></body></html>¥n";
```

	環境変数	意味
リモート（ブラウザ）側の情報	REMOTE_HOST	リモート（ブラウザ）側のホスト名
	REMOTE_ADDR	リモート（ブラウザ）側のIPアドレス
	REMOTE_PORT	リモート（ブラウザ）側のポート番号
	HTTP_USER_AGENT	ブラウザの種類／バージョン
	HTTP_REFERER	このCGIが呼び出される前に参照していたURL
サーバ側の情報	SERVER_NAME	サーバのホスト名
	SERVER_ADDR	サーバのIPアドレス
	SERVER_PORT	サーバのポート番号
	SERVER_SIGNATURE SERVER_SOFTWARE	httpサーバの名前、バージョンなど
	DOCUMENT_ROOT	httpサーバのドキュメントルートディレクトリ
CGIスクリプトの情報	SCRIPT_NAME	スクリプトの仮想パス名
	SCRIPT_FILENAME	スクリプトの物理パス名

表2　CGIで利用できる主な環境変数

で分割しているのですが、ここでは分割するための区切り文字がありません（//）。実は、splitは区切り文字パターンが「//」のときは1文字ずつに分割してくれます。これで$_には1桁ずつの数字が格納されていますから、それを<img……>タグと一緒に出力するだけです。

このように、見た目はグラフィカルで美しい表示も、画像データをうまく使えば簡単なスクリプトで実現することができます。

ヒアドキュメントあれこれ

今回も、HTMLの固定部分で使っていたヒアドキュメントですが、以前少し触れたように、ヒアドキュメントは「固定の部分を出力する」ものではなくて、「複数のprint文を記述しないでまとめる」ためのものです。

つまり、固定部分にしか使ってこなかったのでうっかり忘れていたかもしれませんが、print文と同じように変数展開やメタ文字の使用が可能です。print文でできることはすべてヒアドキュメントでもできるのです。たとえば、配列@ilistに0～9までのいずれかの数字が格納されていれば、

画面3　アクセスカウンタ

画面4　グラフィカルなアクセスカウンタ

```
print <<"EOD1";
<img src='$ilist[0].gif'>
<img src='$ilist[1].gif'>
<img src='$ilist[2].gif'>
<img src='$ilist[3].gif'>
<img src='$ilist[4].gif'>
<img src='$ilist[5].gif'>
EOD1
```

というように、<<"EOD1"; の部分をダブルクォートで囲めば、ヒアドキュメントの中身もダブルクォートで囲まれたものとして、変数展開が行われます。つまり、ここでは$ilist[ ]はすべてその変数の値に展開されて、先ほどの画像データのファイル名になるのです。

それだけではありません。ヒアドキュメントがprint文の代わりだということは、画面への出力だけでなく、ファイルやパイプでつないだメールなどへの出力にも使えるということです。ようするに、ファイルハンドルがオープンできていれば、print文と同様にヒアドキュメントでの出力が可能なのです。

実用的かどうかは別として、てっとりばやくCSVファ

▼リスト8　アクセスカウンタSSI（counter.html）

```
<html>
<head></head>
<body>

アクセス数は <!--#exec cgi="./counter.cgi"-->
です<br>

</body>
</html>
```

▼リスト9　カウンタ用CGI（counter.cgi）

```
#!/usr/bin/perl

print "Content-type:text/html¥n¥n";

open IN,"counter.txt";
$n = <IN>;
close IN;

$n++;

open OUT,">counter.txt";
print OUT $n;
close OUT;

print "$n";
```

イルを作るなら、

```
open OUT,">test.csv";
print OUT <<"EOF1";
名前,住所,電話番号
$name[0],$addr[0],$tel[0]
$name[1],$addr[1],$tel[1]
$name[2],$addr[2],$tel[2]
  ……
EOF1
close OUT;
```

となりますね。

このように、ヒアドキュメント先頭行のprintに続けてファイルハンドルを書けばよいのですから、先ほどのメール送信も、

```
print MAIL <<"EOM1";
Subject: <<mail_form.cgi>>
From: $in{'email'}

登録情報
-----
$addr
〒$in{'zipcode'}
$plist[$pno]
コメント：$comm
EOM1
```

▼リスト10　グラフィカルなカウンタ用CGI（counter.cgi）

```
#!/usr/bin/perl

print "Content-type:text/html¥n¥n";

open IN,"counter.txt";
$n = <IN>;
close IN;

$n++;

open OUT,">counter.txt";
print OUT $n;
close OUT;

$s = sprintf("%06d",$n);
foreach(split(//,$s)) {
        print "<img src='$_.gif'>";
}
```

と書けてしまいます。なによりイメージが掴みやすくなりますし、毎行記述していた「\n」もなくなって、すっきりしていますね。空行部分はそのまま空行にしておけばよいだけです。

ヒアドキュメントをうまく使えばスクリプトを書く手間が省けるだけでなく、見やすくなることによってスクリプト記述のミスが減ったり、あとで修正したり見なおしたりする場合にもとてもわかりやすいものになります。ヒアドキュメントを使うことによるデメリットはありませんから、どんどん使うことをお勧めします。

●

最後に、スクリプト開発においては、あとで自分、もしくはほかの人が見たとしても理解しやすい記述にしておくことがなによりも大切です。自分で書いたスクリプトでも、数カ月もたつと何をやっているのかわからなくなることさえあります。書いているときは頭の中に設計図があるからいいようなものですが、そんなものは次のスクリプトにとりかかるとあっけなく消えてしまうものです。

本当はスクリプトを書くための設計書や仕様書などをきっちり作っておくのが理想的ですが、そこまでの手間をかけなくても、次のようなことを実践すれば特別の労力なく、後々保守のしやすいスクリプト開発ができるでしょう。

- **変数名はできるだけ長く、意味のあるものにする**
- **コメント文をできるだけ多く書く**
- **特別な理由がない限りは計算式などはトリッキーでない方法を使う**
- **サブルーチン化することを考える**
- **スクリプト完成後でもよいから、そのスクリプトが何をするものかをスクリプトの冒頭に書いておく（複数行で）**

こういったことを、少しずつでも習慣付けるようにしてください。きっとあとでよかったなあと思うときがくるはずです。

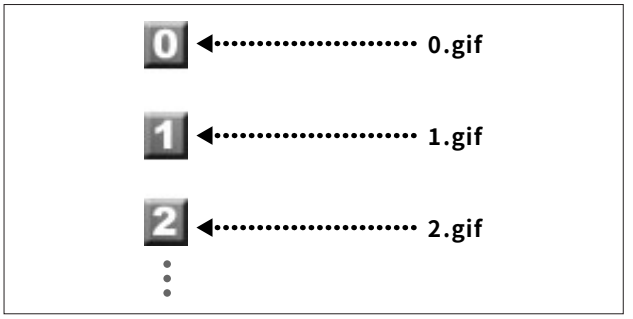


図　アクセスカウンタ用文字の画像

[超] 入門シェルスクリプト

今回は前回に引き続いて、文字列の検索や置換の際に使われる正規表現について取り上げ、メタ文字を利用した柔軟な置換方法や、スクリプト中で正規表現を利用して分岐処理を行う方法などについて説明したのち、日本語の助詞の重複をチェックするスクリプトを改良する。

正規表現で検索や置換を行う（後編）

文：大池浩一
Text:Koichi Oike

前回の冒頭でも述べたが、「正規表現」（Regular Expression）とは、テキストの検索や置換といったマッチ処理に使われる、極めて強力なパターン記述言語だ。正規表現では、通常の文字の並びの中に、「行の先頭（末尾）」「任意の文字」「列挙した文字のいずれか」「文字（列）の繰り返し」「複数の文字列からの選択」といった情報を表わす特殊な記号（メタ文字）として埋め込むことで、柔軟な検索や置換処理が可能になる。

特に、Linuxを含むUNIX系OSには、単機能ツールをはじめ、さまざまなスクリプト言語やエディタ、データベースに至るまで、正規表現の機能を内蔵したプログラムが数多く存在する。bashなどのBourne系シェルには、正規表現の機能は内蔵されていないが、シェルスクリプト中でgrepやsed、exprといったツールを利用することで、正規表現による検索や置換、分岐処理を行うことが可能だ。こうしたスクリプトが行っている処理を理解するにも、正規表現の知識が要求される。つまり、Linuxのソフトを使いこなすうえで、正規表現の知識は避けて通れないのだ。

前回は、grepやPerl、Rubyを利用して、正規表現によるテキストの検索方法を説明した。今回は、sedやPerl、Rubyを利用して、正規表現を利用したテキストの置換処理について説明する。検索に比べて要求される知識は増えるものの、マッチした文字列の一部だけを置換したり、単語の順番を入れ替えるなど、単純な文字列の置換では実現できない処理が可能になる。

正規表現を使って柔軟な置換を行う

使い慣れれば便利な正規表現だが、初心者が習得するのはなかなか難しい。位置、範囲、繰り返し、選択といった情報を表す特殊な記号「メタ文字」をパターン中に埋め込むため、実際のパターンは暗号のような文字と記号の羅列になってしまうからだ。実際、他人が作った複雑なパターンの意味を解釈するのは上級者でも大変だ。

さらに、直前に「¥」を書くかどうかなど、メタ文字の記述方法がプログラムごとに微妙に異なっていたり、あるプログラムで使えるメタ文字が別のプログラムには用意されていないこともある。

たいていのツールで共通して使える基本的な正規表現については、前回の本連載で詳しく説明しているので、そちらを参照してほしい。

以下では、

- **sed、Perl、Rubyを利用した置換処理の方法**
- **柔軟な置換のためのメタ文字の使い方**
- **置換文字列で使える便利な表現**

などについて説明したのち、前回作成したスクリプト「checkjyosi」を改良して、日本語の助詞が重複している部分を一目でわかるように変更する。

● sed、Perl、Rubyを利用した置換処理

正規表現を利用した置換が可能なツールとして、sed、Perl、Rubyを取り上げ、それぞれの簡単な使い方と日本語への対応について解説する。

(1) sedを利用した置換

正規表現による文字列の置換を行うコマンドとして第一に挙げられるのがsed（ストリームエディタ）だ。エディタといっても、viやEmacsのように対話的なソフトではなく、処理内容を指示した命令（スクリプト）に基づいて一括処理を行うスクリプト言語だ。スクリプト次第でさまざまな処理が可能だが、今回は置換に限って話を進める。

sedを使って文字列の置換処理を行う際の一般的な書式は以下のようになる。

```
sed 's/パターン/置換文字列/g' ファイル名
```

最初の引数はスクリプトで、置換（substitute）を意味するコマンド「s」に続けて、正規表現のパターンと置換文字列を「/」で区切って記述し、同一行で複数回の置換を行うフラグ「g」を付ける。このフラグを付けない場合は、最初にパターンにマッチした部分だけが置換される。なお、パターン中に含まれるメタ文字（「＊」や「｜」など）がシェルによって処理されるのを防ぐため、パターン全体を「'」で囲む必要がある。

2番目以降の引数には、内容（テキスト）を読み込むファイル名を指定する。ファイル名は複数指定でき、パイプ経由で他のコマンドの出力を読み込むことも可能だ。ファイルやパイプから入力されたテキストは、1行ずつパターンにマッチするかどうか調べられ、マッチしない場合はそのままの内容、マッチする場合は置換された内容が標準出力（通常は端末画面）に出力される。

たとえば、

```
$ sed 's/LINUX/Linux/g' hoge.txt
```

とすると、hoge.txtの内容に含まれる「LINUX」がすべて「Linux」に置換された状態で、すべての行が端末画面に表示される。

一方、置換処理を行った行だけを出力したいなら、-nオプションとフラグ「p」を付けて、

```
$ sed -n 's/LINUX/Linux/gp' hoge.txt
```

とすればいい。入力行を自動出力する機能は-nオプションの働きで抑制され、置換が行われた行の内容だけがフラグ「p」によって出力される。

日本語を含むテキストに対して、正規表現によるマッチを正しく行うには「マルチバイトパッチ」と呼ばれる修正が必要だ。たいていの国産ディストリビューションに含まれるGNU版のsedは、すでにマルチバイトパッチによる修正が行われた状態になっているため、パターンや置換文字列に日本語の文字列を指定できる。

(2) Perlを利用した置換処理

スクリプト言語のPerlは、WebサーバのCGIなどにも広く用いられている。Perlの特長のひとつは強力な正規表現にあり、他のプログラムには見られない便利なメタ文字が数多く用意されている。ただし、今回もこれらについては触れない。

Perlを使ってsedと同様の置換処理を実現するには、指定したファイルの内容を1行ずつ読み込み、スクリプトに書かれた処理の終了後に自動出力する-pオプションと、スクリプトの内容を直接コマンドラインに記述する-eオプションを使って、

```
perl -pe 's/パターン/置換文字列/g' ファイル名
```

と書く。スクリプト部分がsedと同じ書式なのは、Perlがsedの表記法を取り入れているからだ。

たとえば、sedで挙げた例と同じ置換処理は、

```
$ perl -pe 's/LINUX/Linux/g' hoge.txt
```

とすれば行える。

もし、置換した行だけを出力したいなら、-pオプションを（sedと同じ）-nオプションに置き換え、スクリプト部分の先頭に「print if」を追加して、

```
$ perl -ne 'print if s/LINUX/Linux/g' hoge.txt
```

とすればいい。sedと違ってフラグ「p」は用意されていないので気をつけよう。

正規表現の日本語対応については、Perlのバージョンによって状況が異なる。1つ前の安定版で、現在でも広く使われているバージョン5.005_03には日本語パッチが用意されており、国産ディストリビューションの多くがこのパッ

チで修正されたPerlを利用している。

```
$ perl -v
```

として、「Japanization patch…」という行が含まれていれば、日本語パッチで修正済みのPerlだ。

ただし、正規表現などを日本語対応にするには、Perlを「jperl」というコマンド名で起動する必要がある。もし、jperlというコマンドが存在しない場合は、「su -」としてrootになった状態で、

```
# ln -s /usr/bin/perl /usr/bin/jperl
```

として、perlからjperlにシンボリックリンクを張っておく。日本語テキストを置換する場合は、perlの代わりにこのjperlを利用して、

```
$ jperl -pe 's/ほげ/ふが/g' hoge.txt
```

などとすればいい。

なお、今回は取り上げられないが、Kondara 2.0などに含まれる最新のPerl（バージョン5.6.0）では、ユニコードがサポートされており、UTF-8でコーディングされた日本語テキストをそのまま扱える。

(3) Rubyを利用した置換処理

国産のオブジェクト指向スクリプト言語として人気の高いRubyは、Perlと同様の正規表現クラスライブラリを持っている。また、当然ながら日本語にも最初から対応している。

Rubyを使ってsedと同様の置換処理を行うには、Perlと同じ-pオプションと-eオプション（意図的にPerlと同じ機能が割り当てられている）を使って、

```
ruby -pe 'gsub(/パターン/, "置換文字列")' ファイル名
```

と書く。「gsub」は、正規表現による置換を行うためのメソッド（関数のようなもの）だ。

たとえば、sedで挙げた例と同じ検索処理は、

```
$ ruby -pe 'gsub(/LINUX/, "Linux")' hoge.txt
```

とすればいい。

日本語への対応はもっとも簡単だ。最初から日本語に対応しているので、パッチによる修正やコマンド名の変更は必要ない。起動時に-Kオプションを指定し、処理する日本語テキストのコーディングに応じて、「e」（日本語EUC）、「s」（シフトJIS）、「u」（UTF-8）を直後に書けばいい。なお、「n」（日本語処理なし）を付けるとデフォルトと同じ動作になる。

たとえば、日本語EUCで書かれたテキストを置換する場合は、「-Ke」を付けて、

```
$ ruby -Ke -pe 'gsub(/ほげ/, "ふが")' hoge.txt
```

などとする。複数のコーディングを同時に扱うような場合は、パターンの直後にコーディングをフラグとして指定することもできる。

●柔軟な置換のためにメタ文字を使う

前回説明した正規表現のメタ文字（**表1**）をパターンで使うと、置換対象となる文字列を柔軟に指定できる。もっ

Column

同名ファイルへの上書きに注意

初心者が犯しがちがちなミスのひとつに、「入力ファイルと同名のファイルをリダイレクト先に指定してしまう」ことが挙げられる。たとえば、「sed 's/LINUX/Linux/g' hoge.txt > hoge.txt」としてしまうのだ。

シェルによるリダイレクトの処理は、コマンドの実行に先立って行われ、出力先のファイルはこのとき初期化される。つまり、sedがhoge.txtを読み込む段階では、すでに元の内容が失われているわけだ。

こうした事態を避けるために、リダイレクト先のファイルは元のファイルとは別の名前にする必要がある。たとえば、「sed 's/LINUX/Linux/g' hoge.txt > hogenew.txt」のようにすれば問題ない。さらに、「mv hogenew.txt hoge.txt」とすれば、元のファイルを上書きできる。

一方、PerlやRubyには、指定したファイルの内容を直接書き換える-iオプションが用意されている。たとえば、「perl -pi.bak -e 's/LINUX/Linux/g' hoge.txt」とすると、リダイレクトすることなくhoge.txtの内容を直接置換できるのだ。元のファイルの内容は、-iの直後に指定した文字列（この例では「.bak」）を付加したファイル名で保存される。

保存の必要がなければ、「-pi -e」のように、-iの直後を空白にすればいい。こちらの方法は、対象となるファイルが複数ある場合にも対応できる。

とも簡単に使えるメタ文字は、複数の文字列からの選択を意味する「｜」だろう。

たとえば、hoge.txt中の「LINUX」と「リナックス」を、どちらも「Linux」に置換したい場合は、両者を「｜」で区切って並べたパターンを指定すればいい。PerlやRubyでは、それぞれ、

```
$ jperl -pe 's/LINUX|リナックス/Linux/g' hoge.txt
$ ruby -Ke -pe 'gsub(/LINUX|リナックス/, "Linux")'
hoge.txt
```

と書ける。

sedでは、いくつかのメタ文字（表1の※印）の前に、「¥」（バックスラッシュまたは円記号）を付ける必要がある。上の例では、「¥｜」を使って、

```
$ sed 's/LINUX¥|リナックス/Linux/g' hoge.txt
```

とすればいい。

もちろん、もっと複雑なパターンを指定して置換処理を行うことも可能だ。たとえば、空白文字（スペースまたはタブ）のみで構成される行を対象として、空白文字をすべて取り除く処理を考えてみよう。

まず、空白文字1文字を表すパターンは、「スペースまたはタブ」のいずれか1文字ということだから、文字クラスを使って「[¥t]」と書ける（¥の前はスペース）。大カッコ内の「¥t」は、正規表現や文字列中のタブコードを意味する表記法で、PerlやRubyで利用できる。

このパターンのうしろに「＋」を付けて「[¥t]+」とすると、1個以上の空白文字にマッチするようになる。スペースとタブが混在している場合でも大丈夫だ。

このままだと、空白文字以外の文字が行内に含まれている場合でもマッチしてしまうので、行全体が空白文字であることを表わす必要がある。それには、行頭にマッチする「^」と、行末にマッチする「$」を使って、「^[¥t]+$」と書けばいい。

なお、sedの場合は、タブコードを意味する「¥t」が利用できないため、タブを直接記述する（コマンドラインでは、Ctrl-Vキーを押してからTabキーを押す）。また、「＋」の代わりに「¥+」を使う必要もある。

パターンにマッチした部分を出力から削除するにはどうすればいいだろうか。「削除」ではなく、「長さ0の文字列（空文字列）に置換する」と考え、置換文字列に何も指定しなければいい。

よって、それぞれのコマンドラインは、

```
$ jperl -pe 's/^[ ¥t]+$//' hoge.txt
$ ruby -Ke -pe 'sub(/^[ ¥t]+$/, "")' hoge.txt
$ sed 's/^[     ]¥+$//'  hoge.txt
```

(1)1文字にマッチ	
.	（改行以外の）任意の1文字にマッチ
¥文字	メタ文字の場合はリテラルとして扱う。通常文字の場合はメタ文字列になるものと、そのままリテラルとして扱われるものがある。
[…]	列挙した文字の1文字にマッチ（文字クラス）
[^…]	列挙した文字を除く1文字にマッチ（否定文字クラス）
(2)繰り返し	
＋	1回以上無制限の繰り返し（※）
＊	0回以上無制限の繰り返し
?	0回または1回の繰り返し（※）
{m,n}	m回以上n回以下の繰り返し（m,nの一方は省略可）（※）
{m}	m回の繰り返し（※）
(…)	文字列を繰り返しの対象とする（※）
(3)選択	
｜	両側のパターンのいずれかにマッチ（※）
(…)	選択範囲を変更する（※）
(4)位置指定	
^	行の先頭にマッチ
$	行の末尾にマッチ
¥b	単語の境界にマッチ
(5)前方参照（バックリファレンス）	
¥1、¥2、…	N番目に格納した文字列にマッチ
(…)	カッコ内の部分パターンにマッチした文字列を記憶領域に格納する（※）

表1　sed、Perl、Rubyで使える基本的なメタ文字（列）
（※）の付いたメタ文字（列）をsedで使う場合は、直前に「¥」を付ける必要がある。

となる（sedの大カッコの中はスペースとタブ）。

このケースでは、行の先頭から末尾まで達するパターンを調べるため、マッチする可能性は1行に1回しかないことは明らかだ。このような場合、行内での複数回の置換処理は無意味なので、Perlやsedではフラグ「g」を外し、Rubyでは「gsub」の代わりに「sub」というメソッドを利用するといい。

以上、要点をまとめると、

- **複数の文字列をまとめて置換するには、パターン中でメタ文字「|」で区切って並べればいい。**
- **置換文字列に何も指定しないと、パターンにマッチした部分が削除された状態で出力される。**
- **1行に1回しかマッチしないことがパターンから明らかな場合、Perlやsedではフラグ「g」を外し、Rubyでは「gsub」の代わりに「sub」を利用する。**

ということになる。

●置換文字列で使える便利な表現

続いては、置換文字列で使うと便利な変数や表記法について説明する。これらは正規表現のメタ文字ではないので、パターンに記述することはできない。

(1)パターンにマッチした文字列を取り出す

パターンにマッチした文字列全体を置換文字列で指定するには、Perlでは「$&」、Rubyでは「￥￥&」、sedでは「&」という表記法を利用する。

たとえば、

```
$ jperl -pe 's/Linux/★$&★/g' hoge.txt
$ ruby -Ke -pe 'gsub(/Linux/, "★￥￥&★")' hoge.txt
$ sed 's/Linux/★&★/g' hoge.txt
```

とすると、いずれもhoge.txtに含まれる「Linux」が「★Linux★」に置換された状態で出力される。

もっとも、この例では置換文字列にそのまま「★Linux★」と書いても同じ結果になる。「$&」などの記号が役に立つのは、マッチする文字列を置換文字列に直接記述できないような場合だ。

たとえば、hoge.txt中に含まれる7桁の郵便番号（たとえば「272-0825」など）の前に、「〒」を挿入するというケースを考えてみよう。なお、郵便番号の前後はスペースやカンマなどで区切られているものとする。

Perlを使った例を次に示す。

```
$ jperl -pe 's/￥b[0-9]{3}-[0-9]{4}￥b/〒$&/g' hoge.txt
```

パターン中の「[0-9]{3}」は任意の数字3桁、「−」はハイフン（マイナス）、「[0-9]{4}」は任意の数字4桁にそれぞれマッチする。また、「1234-56789」といった文字列の一部にマッチしてしまわないように、スペースやカンマなどの単語境界にマッチするメタ文字「￥b」を先頭と末尾に記述している。

置換文字列では、「〒」のうしろの「$&」が、マッチした文字列全体で置換される。こうしたケースでは、マッチする文字列を置換文字列に前もって記述することはできないから、「$&」を使う価値があるわけだ。

(2)パターンにマッチした文字列の一部を取り出す

正規表現を使った置換処理の大きな特徴として、パターンをいくつかのサブパターンに分割し、それらにマッチした文字列の順番を変えたり、一部だけを別の文字列に置換できる点が挙げられる。

PerlやRubyでは、正規表現のパターン中で「(」と「)」で囲んでサブパターンを指定し、それぞれのサブパターンにマッチした文字列を、「$1」「$2」…という変数に保存する。「$」に続く数字は、パターンの左端から数えた「(」の登場順に対応する。

たとえば、hoge.txtに含まれる「ドライブ△」（△はA〜Zの英字1文字）という表現を、すべて「△ドライブ」に置換したい場合、Perlでは、

```
$ jperl -pe 's/(ドライブ)([A-Z])/$2$1/g' hoge.txt
```

とする。最初のサブパターンにマッチする「ドライブ」が変数$1、2番目のサブパターンにマッチする文字列（任意の英大文字1個）が変数$2にそれぞれ格納され、置換文字列の「$2$1」によって逆順に出力される。「ドライブA」が「Aドライブ」になるわけだ。

Rubyでも同様の変数を使えるが、gsubやsubによる置換処理では注意が必要だ。というのも、gsubの通常の指定方法「gsub(/パターン/, "置換文字列")」では、置換文字列の評価が正規表現パターンのマッチ処理より前の段階で行われるからだ。

このため、置換文字列に$1を含める（Rubyの場合は文字列中に「#$1」と書く）と、1回前のマッチ処理で$1に格納された文字列（最初のマッチ処理の場合は空文字列）で$1が先に置換されてしまい、意図した結果が得られないことになる。

これを解決する方法は2通りある。ひとつは、サブパターンにマッチした文字列を表す別の表記法「￥￥1」「￥￥2」…を置換文字列に指定する方法だ。たとえば、

```
$ ruby -Ke -pe 'gsub(/(ドライブ)([A-Z])/, "￥￥2
￥￥1")' hoge.txt
```

とすると、Perlと同じ結果が得られる。

もうひとつの方法は、Rubyの特徴のひとつである「ブロック（イテレータ）」を使って、

```
gsub(/パターン/) { "置換文字列" }
```

とすることだ。この方法では、置換文字列の評価より前の段階でマッチ処理が行われ、$1などの変数に結果が格納される。あとは、

```
$ ruby -Ke -pe 'gsub(/(ドライブ)([A-Z])/) { "#$2#$1"
}' hoge.txt
```

とするか、あるいは、

```
$ ruby -Ke -pe 'gsub(/(ドライブ)([A-Z])/) { $2+$1 }'
hoge.txt
```

とすれば、Perlと同じ結果が得られる。

sedの場合、$1などの変数はそもそも存在しない。置換文字列では「￥1」「￥2」…という表記法でサブパターンにマッチした文字列を参照できる。また、カッコの直前には「￥」を付ける必要がある。たとえば、Perlの例と同じ結果を得るには、

```
$ sed 's/￥(ドライブ￥)￥([A-Z]￥)/￥2￥1/g' hoge.txt
```

とすればいい。

●マッチ文字列の一部だけを置換する

今度は、マッチした文字列の一部だけを置換する処理について取り上げよう。こうした場合は、残したい部分だけをカッコで囲んでサブパターンとし、置換文字列で$1などを指定する。

たとえば、hoge.txtに含まれる「Linuxマガジン」と「BSDマガジン」の「マガジン」を「 magazine」に置換し、それ以外の「マガジン」はそのまま残すというケースを考えてみよう。

この場合、単純に、

```
$ jperl -pe 's/マガジン/ magazine/g' hoge.txt
```

とすると、たとえば「少年マガジン」まで「少年 magazine」に置換されてしまう。

「Linux」や「BSD」に続く「マガジン」だけを置換するには、これらをパターンの一部として指定し、

```
$ jperl -pe 's/(Linux｜BSD)マガジン/$1 magazine/g'
hoge.txt
```

とすればいい。最初のサブパターンにマッチする文字列（「Linux」または「BSD」）が$1に格納され、置換文字列の先頭でそのまま出力される。それ以外の部分（「マガジン」）は、置換文字列の「 magazine」に置換される。

なお、冒頭でも述べたが、$&や$1、$2…といった記号は、正規表現のメタ文字ではないので、「/パターン/」の内部では使えない。サブパターンにマッチした文字列を、同じパターン内で利用する場合は、前回説明した「前方参照」（バックリファレンス）を利用する。

この節での要点をまとめると、

- **パターンにマッチした文字列全体を置換文字列中で指定するには、Perlでは「$&」、Rubyでは「￥￥&」、sedでは「&」を置換文字列中に指定する。**
- **マッチした文字列を複数の部分に分けて処理するには、後で必要となるパターンの一部を「(」と「)」で囲んでサブパターンとする。**
- **サブパターンにマッチした文字列を置換文字列で利用する場合、Perlでは「$1」、「$2」…、Rubyでは「￥￥1」「￥￥2」…、sedでは「￥1」「￥2」…で参照する。**
- **「$」に続く数字は、パターン左端から数えた「(」の登場順に対応する。**

ということになる。

今月のスクリプト

後半は、与えられたテーマを実現するスクリプトの作成手順を説明する。今月は、前回作成した、

・日本語テキスト中の1文字の助詞の重複をチェックするスクリプト「checkjyosi」

を、チェックすべき部分が一目で判別できるように改良する。具体的には、重複部分が端末画面で反転表示され、行頭にファイル名や行番号を付け加えることにしよう。

●前回のスクリプトの復習

日本語の助詞の中でも、「が」や「を」といった1文字の助詞は、テキスト入力時の打ち間違いや、その後の編集作業などによって重複することがあり、見逃しやすい誤りのひとつだ。

1文字だけで構成される助詞は、「が」「を」「に」「と」「で」「へ」「の」「は」「も」「か」「さ」「ね」の12個ある。このうち、「と」「で」「の」「も」「さ」「か」については、実際にチェックを行ったところ、「こととする」や「〜でできる」、「するものの」といったように、連続していても助詞の重複ではないケースが多いため、今回の検索対象からは除外する。

残りの6文字「が」「を」「に」「へ」「は」「ね」のいずれか1文字にマッチする正規表現のパターンは、前回学んだ文字クラスを使って、

```
[がをにへはね]
```

と書くことができる。

「ががが」や「ををを」のように、同じ文字が連続することを表現するには、上記のパターンをカッコで囲んでサブパターンとし、それにマッチした部分を「¥1」で前方参照（バックリファレンス）すればいい。つまり、

```
([がをにへはね])¥1
```

となる。念のため、「ががが」のように、同じ文字が3個以上続くような場合も考慮して、1回以上の繰り返しを表すメタ文字「＋」を加えて、

```
([がをにへはね])¥1+
```

とすればパターンは完成だ。

前回は、このパターンによる検索をRubyを使って行って、マッチした行をそのまま出力するスクリプトを作成した。しかし、出力を注意深く見ないと、どの部分が重複しているのかわかりにくい（**画面1**）。それほど助詞の重複は見逃しやすいのだ。

●マッチする部分を反転表示で目立たせる

マッチした部分の文字色や背景色が変化すれば、もっとチェックが楽になるはずだ。このような処理は、単なる検索だけでは無理で、出力されるテキストを置換処理で加工する必要がある。

端末に出力する文字の色を変化させるには、それぞれの端末に対応したエスケープシーケンスを利用する。エスケープシーケンスとは、エスケープコード（0x1b）で始まる一連の文字列のことで、カーソルの移動や画面クリアなど、さまざまな画面制御を行える。

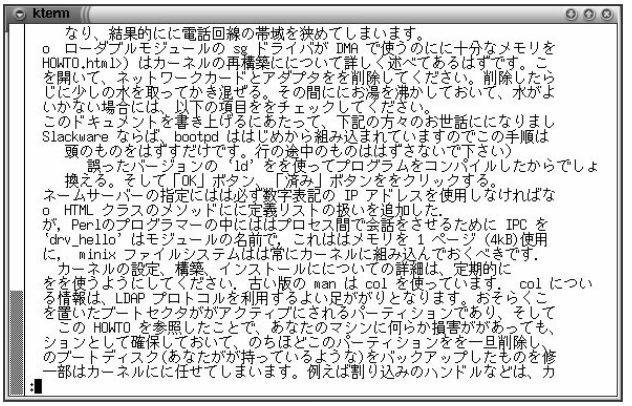
たとえば、ANSI準拠の端末（ktermやrxvtを含む）では、反転表示はエスケープコードに続く「[7m」、通常表示はエスケープコードに続く「[m」という文字列で行う。マッチした文字列の前後にこれらを挿入すれば、マッチした文字列が反転表示されるはずだ。

前回はRubyを使ったので、今回はPerlで処理することにしよう。この処理を行うPerlのスクリプトは、

```
print if s/([がをにへはね])¥1+/¥e[7m$&¥e[m/g
```

となる。

パターンにマッチした文字列全体が格納される変数$&



画面1　どの部分が助詞の重複なのか、ちょっと見ただけではなかなか判別できない。

を置換文字列で使用し、その前に「¥e[7m」、後ろに「¥e[m」を挿入している。「¥e」は、文字列でエスケープコードを使うためのPerlやRubyの表記法だ。

たとえば、hoge.txtに対して、助詞が重複していないかチェックするには、

```
$ jperl -ne 'print if s/([がをにへはね])¥1+/¥e[7m$&¥e[m/g' hoge.txt
```

とすればいい。助詞が重複する（と思われる）部分が反転表示されるので、人間がそれを見れば一目でチェックできるようになる。

●改良版のスクリプトを作成する

前節で作成したPerlのコマンドラインを取り入れた改良版のcheckjyosiをリスト1に示す。内容と使い方について説明していこう。

2～5行目では、コマンドライン引数を1つも指定しなかった場合に、使い方（usage）を標準エラー出力に表示してスクリプトを終了する処理を記述している。

なお、他のコマンドの出力などをパイプ経由で読み込む際、引数を付けないと上記の処理が行われてしまう。こうした場合は、標準入力を意味するファイル名「−」をコマンドライン引数に指定すればいい。

6行目では、前節のPerlのコマンドラインを改良したスクリプトが使われている。重複している助詞にマッチする正規表現のパターンや、反転表示のためにエスケープシーケンスを埋め込む部分は同じだ。

前節のコマンドラインでは、Perlの関数「print」を引数なしで使って、置換処理後の入力行を出力していた。これに対し、**リスト1**では関数「printf」を使って、処理中のファイル名（$ARGV）、行番号（$.）、置換後の入力行（$_）を整形して出力する。これで、複数のファイルをcheckjyosiのコマンドライン引数に指定した場合でも、どのファイルの何行目が重複しているかわかる。なお、後半の「close(ARGV) if eof」は、ファイルを1つ読み終えるごとに、行番号をリセットするための処理だ。

実際に、改良版のcheckjyosiを使って複数のテキストをチェックすると（**画面2**）、助詞が重複していると思われる部分が反転表示されて一目でわかるし、行頭のファイル名や行番号によって、人手での修正作業も楽になる。

なお、出力をlessを使って閲覧する場合は、

```
$ checkjyosi hoge.txt | less -R
```

のように、エスケープシーケンスを解釈する-Rオプションをlessに付ける必要がある。

なお、checkjyosiでは、特定のひらがなが連続している部分を検出するだけなので、マッチした部分が本当に助詞の重複なのかどうか、人間が目で見て確認する必要がある。このため、助詞の重複を自動的に修正するには、前後の文脈を判断するなど、さらに複雑なスクリプトを作成する必要がある。しかし、助詞が重複していると思われる部分をユーザーに示し、対話的な操作で修正を行うPerlやRubyのスクリプトなら、作るのはそれほど難しくない。腕に自信のある人は挑戦してみてはどうだろうか。

画面2　重複する助詞が反転表示で示され、ファイル名や行番号も表示される。

リスト1　日本語テキスト中の1文字の助詞の重複をチェックするスクリプトcheckjyosi（改良版）

```
1: #!/bin/sh
2: if [ $# = 0 ]; then
3:   echo "usage: checkjyosi file..." > /dev/stderr
4:   exit 1
5: fi
6: jperl -ne 'printf("%s%5d:%s", $ARGV, $., $_) if s/([がをにへはね])¥1+/¥e[7m$&¥e[m/g; close(ARGV) if eof' "$@"
```

Linux日記

第27回　メールのセキュリティ（1）

MTAの重要な仕事のひとつに、メールを中継するということがあります。しかし、すべてのメールを無条件に転送するわけにはいきません。

illustration ; Aki

文：榊 正憲
Text : Masanori Sakaki

今年の関東地方は台風の当たり年だったようだ。8月中旬の台風は、勢力が弱くなっていたとはいうものの直撃だったし、9月上旬の台風は、筆者の住む湘南地方に直接上陸した。

8月の台風は、ちょうど東海地方から関東に向かって車で走っているときに、後ろから追われるような形になった。暴風雨圏には入らずに済んだのだが、箱根峠の下り（小田原側）で時間雨量50mmという雨をくらった。さすがにかなりの雨だったが、まだ風が強くなかったので、どうということはなく走ることができた。かつて九州にドライブに行ったときに、台風の直撃で暴風と大雨をくらったことがある。このときは、吹きっさらしのループ橋ではワイパーも効かず、橋の手すりを見ながら走るというかなり辛い状態だった。

9月の台風では、初めて台風の目を経験した。台風が近いはずなのに、風雨がほとんど収まったのである。その後、風向きが逆転して、また大雨になった。気象情報を見たら、筆者の住む場所のごく近所に上陸とのことだったので、台風の目だったわけだ。目はすでにだいぶ崩れていたので、残念ながら青空は見えなかったが。

今回の台風では、幸い停電も瞬断もなく済んだ（近所では停電があったようだ）。昨年の同じ頃、雷雨で半日の停電となり、オリンピックの柔道の決勝戦を見られなかったという悔しい思いをし、UPSの導入が必要だと思いつつ、結局何もしなかった筆者としては、ラッキーであった。

この台風の日は東京に行く用事があったので、暴風雨が一段落したところで出かけた。夜10時前に家に帰り、台風はどうなったかなとNHKのニュースを見たら、「今入ったニュースです」といって、貿易センタービルの一方が炎上している映像になった。キャスターと現地の記者が話している最中に、飛行機が飛来し、炎上するビルの影に入り、「あれ、出てこない」と思ったら反対側から炎が吹き出した。NHKの記者とキャスターは何事もなく話を続けている。「えっ、えっ、どうなってんの。2機めが突っ込んだの」と家族と話しているうちに、局側もこの事態を認識し、さらに大騒ぎになった。その後、ビルは多くの人々を中に取り残したまま倒壊した。犠牲になった人々の冥福を心よりお祈りする。

■危機管理

テロに対する備えをするのも、停電に備えるのも、インターネット接続に際して各種セキュリティ管理を行うのも、すべては危機管理である。危機管理には、2つの側面がある。「問題が起こらないようにする」ことと、「問題が起きたときにどのように対処するか」ということである。どちらも重要であり、一方だけで済むというものではない。我が家のように、停電が過ぎてしまえばUPSの導入のことを忘れてしまうというのは、危機管理がなっていない例である。危機の際には呆然として何もできず、過ぎてしまえば忘れてしまって次の危機に対する備えを何もし

ない日本政府によく似ているかもしれない（今回は少し頑張るようだが）。

コンピュータのセキュリティの話に戻そう。うちのように個人の場合は、すべては心がけということになるのだが、会社などの組織の場合は、一部の人間がきちんと心がけているだけでは、各種自然災害、人的災害／犯罪の被害を防ぐことはできない。一般に、問題が発生したときには、被害は弱いところに集中する。ウィルスメールは、アンチウィルスソフトを導入していなかったり、ウィルス予防データベースの更新を怠っているマシンから広がるだろう。クラッカーは、ろくに管理されていないサーバを踏み台に使うだろう。高価なファイアウォール機器を導入していても、使っている人間が気にしていなければ、組織は脆弱なままなのである。

■迷惑メール

筆者はあまり出歩くわけではないのだが、一応携帯電話を持っている。Docomoのi-mode機なので、インターネット電子メールも使える。これでしばらく前から問題になっているのが、「出会い系サイト」の宣伝をはじめとする迷惑メールだ。今年の夏くらいまでは、Docomoの携帯で使うメールアドレスの初期状態は、電話番号@docomo.ne.jpという形式だったので、機械的にアドレスを列挙するだけで、不特定多数宛のメールを送信できた。この電話番号形式のメールアドレスは、好きな文字列に変更することができるので、自分の名字や名前、ニックネームなどにすることで、このような機械的に生成された迷惑メールは届かなくなる。

迷惑メールが社会問題化してから、Docomoが番号のままでなく、このようにアドレスを変更することをアナウンスし、また初期状態のメールアドレスをランダムな英数字の組み合わせに変更したことで、電話番号のしらみつぶし形式で送られる迷惑メールの被害は、一時的にだいぶ減ったのだが、敵もこの程度では引っ込まない。よく使われるメールアドレス、たとえば名前や名字の組み合わせなど、ありがちな文字列を使ってメールを送るということを始めたようだ。我が家の携帯電話は、契約してすぐに電話番号ではない形式のアドレスに変えていたので、電話番号指定の迷惑メールは届いたことがなかったのだが、Docomoがアドレス変更のアナウンスを始めてから、この種のメールが送られてくるようになった。発信者のアドレスはいろいろ違っているので（だからといって発信者が違うとは限らないが）、ありがちなメールアドレスを自動生成するプログラムでも出まわっているのだろう。

郵便のダイレクトメールやテレビのCMなどと異なり、電子メールを使ったこの手の広告は、受け取り側に金銭的負担がかかるという点で悪質である。一般に、広告情報というのは、それが存在することによって、受け取り側にもメリットが発生するというのが普通だった。テレビCMであれば、それが存在することによってテレビ番組を無料で視聴できるわけだし、新聞のチラシがなくなれば、新聞代が高くなる可能性がある。広告を出す側、見る側の双方にメリットがあり、ビジネスモデルとして成立しているわけだ。ダイレクトメールはメリットはないが、特に損するものでもない（森林資源やゴミの問題はあるが）。しかし電子メールによるダイレクトメールは、受信するだけで費用が発生し、それを受信したことによるメリットが（出会い系サイトを探している人以外には）まったくないというのが問題だ。広告メールを受信することで、たとえば通話料金の割引があるといったような、受信側のメリットがまるでない。これは不愉快きわまりない事態である。結局、Docomoが一定量のパケットについて無料にするという対処を行っているわけだが、要するに、広告を出す側がシステムの性質を悪用して、費用はDocomo持ちで（さらに量が増えれば、受け取り側持ちで）無料で広告を出しているという形だ。

携帯電話のメールも含めて、電子メールシステムは非常に便利なものなのだが、この利便性は、善意のユーザーにも悪意のユーザーにも等しく提供されている。利便性を悪用することで、誰かが損をしたり、あるいは便利なものが使いにくくなってしまうというのは残念なことである。

今回は前回から続きで、電子メールシステムのセキュリティの話である。

■sendmailのセキュリティホール

常時稼働し、外部からのネットワーク接続を受け付けるサーバプロセスは、攻撃の対象になりやすい。特にsendmailなどのメール配信用のサーバ

Column

スパムメール

スパム（spam）メールというのは、不特定多数に向けて大量に配信されるメールの総称である。悪質なダイレクトメールや、各種悪徳商法の勧誘などがこれに相当する。SPAMというのは、もともとは肉の缶詰（かなり塩辛い）の商標で、この製品のコマーシャルが「スパム」という名称を連呼したことから、この名前が付いたという説がある。

は、パスワードなどによる認証を行うことなく接続でき、しかも対話的な処理を行うという点で注意が必要だ。

MTAは、受信したメールを単なる文字列データとして処理するだけなので、メールに添付されたウィルスプログラムによって、MTAが影響を受けることはない。もっとも、受信したメールを処理するプログラム、たとえばメールによってデータベースを自動処理する場合などは、そのレベルで問題が発生することはあるだろう。

MTAのレベルでの被害は、SMTP接続セッションのレベルで、侵入を試みる、あるいは障害を発生させるという形で行われる。かつて騒ぎを起こしたインターネットワーム事件は、sendmailのSMTP接続で使われるコマンドに、デバッグ用の機能が残っていたことが原因だった。この機能により、sendmailにSMTPで接続したときに、sendmailデーモンのユーザー権限（rootである）で侵入できてしまったのである。現在では、サーバとして動作するプロセスについては、このような機能を（少なくともリリース版には）用意しないのが常識となっている。また、可能であれば、プロセスをroot権限で動作させないことが望ましい。また、このような裏機能の利用とは別に、意図的にエラーを発生させ、そのときのプログラムの動作を利用する侵入などもある。

現在に至るまで、sendmailに限らず、多くのサーバプログラムにセキュリティホールが発見され、それに対するパッチがリリースされるという状況が繰り返されている。今日、通常のディストリビューションには、既知のセキュリティホールをふさいだソフトウェアが含まれているが、それでもセキュリティホールが存在しないという保証があるわけではない。sendmailに限らず、なるべく新しいバージョンのソフトウェアを運用し、まめにパッチ（特にセキュリティ対策のもの）を当てるとか、最新のリリースにアップグレードするといった対策が必要である。

■過負荷に対する対処

ユーザーが多く、サイトで扱うメールの絶対量が多いといった理由による過負荷の場合は、回線やサーバの増強をはかることになるが、こういった状態とは別に、一時的に過負荷状態が発生することがある。これには次のような状況がある。

・大量のメールの受信

短時間に集中してメールを受け取ると、回線容量やコンピュータの処理能力が不足し、メール配信エラーなどが発生することがある。これは、何らかのエラーなどにより発生する場合もあるし（たとえば、大量にメールを送信するMLサーバで設定を間違えたなど）、悪意のあるユーザーによる攻撃によって発生することもある。このような状態が続くと、受信メールをスプールするディスクがあふれてしまうこともある。こうなると、メールを新規に受信できなくなってしまう。また、メールの数が少なくても、巨大なメールがいくつも送られてくると、やはりスプールはあふれてしまう。

対処としては、スプールの残り容量を常時モニタし、容量がある限界に近づいたら、古いメールを抹消するなどの方法がある（このような方法を採るプロバイダもある）。また、MTAのレベルで、受信メールのサイズを制限することも可能だ。1本のメールサイズの上限（たとえば5Mバイトなど）を設定すると、それ以上のサイズのメールの受信を拒否するようになる。もっともこれは添付ファイルがある場合にも適用されてしまうので、判断の難しいところだ。

外部のさまざまなアドレスから大量に送られてくるメールを制限するのは難しいが（極端な場合は、ほとぼりが冷めるまで、回線を切断したり、サーバを止めるなどしか方法がない場合もある）、特定サイトからのメールであれば、MTAのレベルで特定ドメインからの着信を拒否したり、ルータやサーバのフィルタリング機能で、特定のMTAからの接続を拒否するという対

Column

UNIXの普及の善し悪し

個人レベルへの各種UNIXの普及は、「インターネットの治安」という意味では、危険な要素の増大という側面もある。特に最近の高速回線を使った（準）常時接続の広がりがそれに輪をかけている。

こういったUNIXマシンは、インターネット用の各種サーバとなりえる。インストールを行った正規のユーザーが何もしていなくても、ちょこっと設定ファイルを書き換えれば（あるいは何もしなくても）、いろいろなサービスを動かすことができる。悪意のユーザーは、きちんとパスワードを設定していないこのようなマシンをいくらでも見つけることができるだろう。

事実上常時接続として運用される各種の定額接続サービスでは、IPアドレスは固定ではないし、DNSにも名前が登録されているわけでもないので、外部からアクセスされてもたいした問題はないと思うかもしれないが、そんなことはない。悪意のユーザーがIPアドレスをスキャンした時点で接続可能であれば、各種攻撃を行うための「踏み台」として使われることがあるのだ。

応が可能である。

・大量のメールの発信

大規模なメーリングリストの運用や、メールで増殖するウィルスなどにより、大量のメール送信が行われることがある。メーリングリストなど、正規の運用であれば、設備の拡充を検討しなければならないだろう。ウィルスの増殖は、感染したユーザーのマシンが自身のコピーを大量に送信するという形で行われる。そのため、複数のコンピュータが感染したりすると、一時的に大量のメールが送信されることになる。ウィルスが自身のコピーをメールで配信するのを防ぐには、まずはウィルスを検出し、動作する前に排除することである。

・大量のメールの中継

悪意を持つユーザーがMTAの中継機能を不正使用して、スパムメールやウィルスメールの配信を行うというパターンである。この問題は、過負荷の問題というよりは不正メールの配信の問題になる。この問題については次に説明する。

■メール中継機能

メールシステムを実際に運用するうえで、最低限行わなければならない対策は、MTAの中継機能を不正に使われないようにすることである。

まずはMTAの中継機能について復習しよう。sendmailのようなMTAを運用しているUNIXホストを使ってメールを配信する場合、いくつかのパターンがある。

・ローカルMUAを使って送信

ucbMailのようなローカルMUAを使い、内部からsendmailを起動し、宛て先ユーザーを管轄するMTAホストに直接送信する場合は、メールは発信側MTAから受信側MTAに直接送られるので、ネットワークを介したメール中継という処理は行われない。

・ハブサーバの運用

メールの送信にハブサーバを使っている場合、部署単位などの末端MTAからのメールは、ハブサーバに送られる。そしてハブサーバは、このメールを宛て先ユーザーを管轄するMTAに送信する。受信にハブサーバを使っている場合は、外部からのメールをまずハブサーバが代表して受信する。そしてハブサーバはこのメールを末端のMTAに送信する。送信と受信のどちらの場合も、ハブサーバは、受信したメールを別のMTAに送信することになる。これはメールの中継処理である。

・POP／IMAPクライアントの使用

ローカルMUAを使用せず、POP／IMAP対応のメールクライアントプログラムを使っている場合、受信メールの閲覧、ダウンロードにはPOP／IMAPを使う。しかし、クライアント上で作成したメールの送信にはSMTPが使われる。クライアントMUAはMTAに対してSMTP接続を行い、作成したメールをMTAに送信する（クライアントが接続するMTAホストの情報は、クライアント上で静的に設定されている）。このメールを受信したMTAは、宛て先情報に基づいてこのメールを別のMTAに送信する。これは中継処理である。

現在の電子メール環境、つまり、ネットワーク接続されたクライアントの使用、大規模な組織におけるハブサーバの運用などでは、MTAのメール中継機能は不可欠だ。MTAがメールを中継することによって、自由度の高い電子メールシステムを構築することができるのである。

■中継機能の不正利用

通常の電子メールにはFrom：行があり、たとえメール本文にシグネチャや自己紹介がなくても、だれから送られてきたメールかわかる。メーリングリストなど、メールを実際に発信した実体がFrom：行のユーザーとは異なる場合もあるが、これは現実世界でもよくあることだ。大量に発送する手紙の差出人の住所と、発送代行業者の住所が違うといったことはよくある。

このような理由による発信実体の食い違いは特に問題はない。しかし、悪意を持つ人間が、差出人を詐称した手紙を送ることも可能である。紙の手紙の場合は、偽の差出人住所を記すだけでできてしまう。同じように電子メールでも、発信アドレスを詐称してメールを送ることができる。発信者のアドレスを詐称することで、発信者や受信者に迷惑をかけることもできるし、あるいは架空の発信アドレスを使って、発信者が誰だかわからなくすることもできる。こういった詐称でどのような悪さを行えるかにはここでは触れないが、このようなメールが有害であることは明らかだ。

アドレス詐称メールは、ucbMailのようなローカルMUA環境では作りにくい。From：行は、ホスト名やユーザー名などに基づいて、システムが作成するからだ。自分でシステム設定を変更できる環境なら可能だが、きちんと管理されているUNIXホスト上ではかなり難しくなる（もっとも、telnetで直接SMTP接続を行えば、何でもできてしまうが）。しかし、Windowsなどのクライアントコンピュータ上で、

POP／SMTP対応のメールクライアントアプリケーションを使っている場合は、いとも簡単にできてしまう。

このようなメールクライアントは、作成したメールをSMTPで適当なMTAに送信する。この段階では、送信されるメール中にはFrom：行があり、発信者のエンベロープ情報もメールクライアントが作成する。つまり、発信者の情報は、完全にクライアント側で作成しているのである。これは、メールクライアントアプリケーションの設定パラメータで指定するのが普通だ。すなわち、これらの情報はユーザーが自由に変更できる。発信者の名前やアドレス情報を変更すれば、好きなアドレスでメールを送信することができるわけだ。

たとえば、プロバイダと個人契約しているユーザーの場合を考えてみよう。ユーザーは、user@mail.provider.ne.jpというメールアドレスを使用し、mail.provider.ne.jpというメールサーバを使うようにプロバイダから指定されている。このような形でメールを使うために、ユーザーが使うメールアプリケーションでは、SMTPサーバ、POPサーバとしてmail.provider.ne.jpというホストを指定し、ユーザーのメールアドレスとしてuser@mail.provider.ne.jpという文字列を登録することになる。プロバイダから指定されたuser@mail.provider.ne.jpというメールアドレスは、メールを送信する際の発信アドレスとして使われるとともに、POPで受信メールをダウンロードする際のユーザーアカウントとしても使われる（userがPOP接続のユーザーアカウントとなる）。したがって、普通にメールを使う場合は、メールアドレスを指定されたもの以外にしてしまうと、メール受信ができなくなってしまうので意味がない。

SMTPサーバmail.provider.ne.jpは、契約ユーザー宛のメールを受信するために、任意の発信アドレスを持つメールを受信する。また、契約ユーザーが作成したメールをクライアントMUAから受信し、宛て先ユーザーに送るために、中継機能を持っている。このようなMTAに対して、クライアントMUAが作成した偽の発信者アドレスを持つメールを送るとどうなるか？MTAはこのメールを受信し、宛て先を評価し、その宛て先に中継してしまうのである。いとも簡単に、発信アドレスを詐称したメールを送れるのだ（**図1**）。

たとえば、ウィルスの配布を意図しているのであれば、偽りの発信アドレスを使うだろう。また、性悪なダイレクトメール業者などは、適当な発信アドレスを使って大量のメールを送信する。偽りのFrom：であれば、クレームのメールなどは返ってこないし、相手が発信者を突き止めるのも難しくなる。また、低速回線で接続された環境では、数万通といった大量のメールを送るのは現実的ではないが、多数のアドレスをエンベロープ情報で指定するという形でSMTPサーバに送れば、大量のメールのコピーは、中継先のSMTPサーバで作成され、送信される（**図2**）。MUAからSMTPサーバまでは、メールが1本送られるだけである（エンベロープ情報は巨大であるが）。

このような、MTAの中継機能の悪用を総称して、中継機能の不正利用とか不正中継という。MTAの中継機能は、必要なものではあるのだが、いろいろな危険性も合わせ持っているので

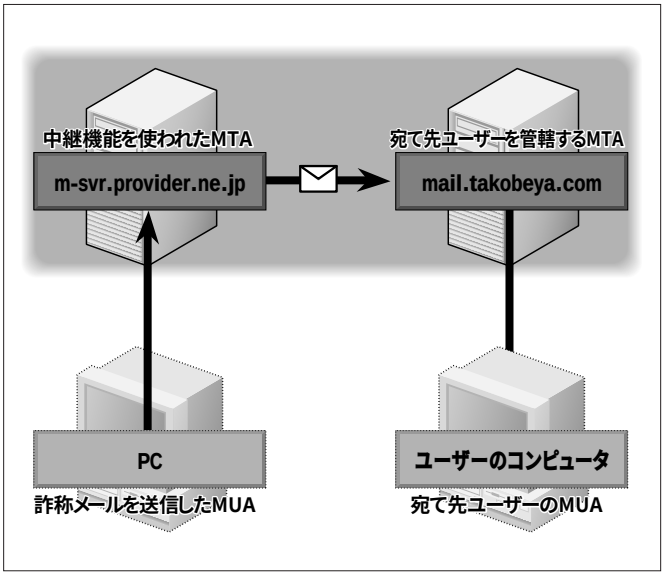


図1　アドレスを詐称したメールの送信

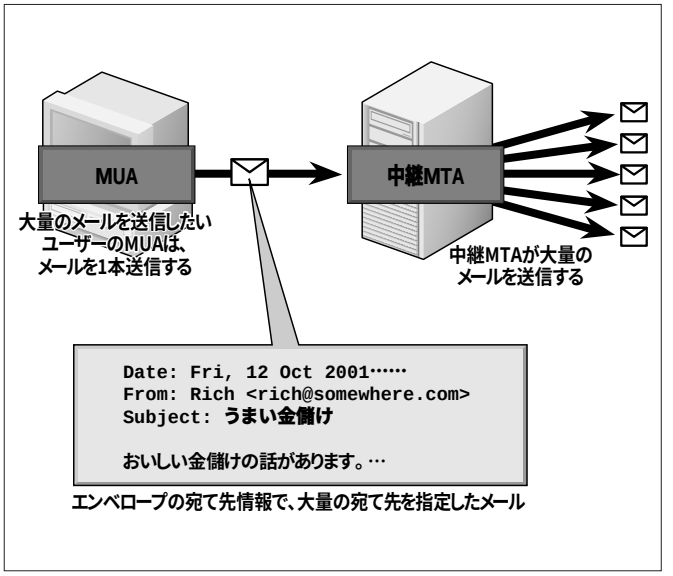


図2　SMTPサーバを使った大量のメール送信

ある。

このようにして作成された詐称メールは、Received：ヘッダを見ることで、ある程度判断することができる。

Received：ヘッダは、中継、受信などの処理を行ったMTAが順に付加していく。Received：ヘッダは、毎回先頭に付加されていくので、最後のものから順に見れば、メールがどのような経路で送られてきたのか、配信履歴がわかる。POP／IMAPメールクライアントから送られたメールの場合、一番最後のReceived：ヘッダが、発信者のプロバイダや組織のMTAになっているはずである。メールのFrom：に記述されているメールアドレスと、Received：に示されているMTAのホスト名を比較することで、中継機能を不正に使った詐称メールかどうかを、ある程度は調べることができる。

リスト1に示したメールの例では、メールはm-svr.provider.ne.jpという国内のプロバイダから送られてきているが、From：は、kiriyama@tdf.int、つまりtdfという国際機関（地球防衛軍だろうか？）のkiriyamaというユーザーになっている。あきらかに怪しい発信アドレスである。

では、From：行がyamada@ascii.co.jpで、最初の中継MTAがm-svr.provider.ne.jpであった場合は、詐称メールだろうか？ これはReceived：だけではわからない。yamada@ascii.co.jpという発信アドレスを騙って、誰かがm-svr.provider.ne.jpを中継MTAとして使ったということも考えられるが、m-svr.provider.ne.jpが、独自ドメイン名を使ったメールサービスを提供しているかもしれないからだ。この場合は、From：行のアドレスと、中継MTAがまったく異なるドメイン名になるのが普通だ。このような場合は、nslookupを使って、DNSにascii.co.jpのMXレコードを問い合わせてみるといい。MXレコードがm-svr.provider.ne.jp、あるいは同じプロバイダの別のメールサーバを示していれば、ascii.co.jpがメールサーバをこのプロバイダに委託していることがわかる。

■中継の問題

sendmailなどのMTAは、デフォルトでこのような中継処理を行うように設定されている（何もしなければこのように動作する）。したがって、不正中継に使われる可能性がある。自分の管理するMTAが不正中継に使われた場合、次のような被害が考えられる。

・リソースの不正使用

サーバコンピュータや通信回線、スプール容量などのリソースの無断使用といった被害が発生する。大量のメール送信の中継に使われた場合は、本来の業務に支障をきたす可能性もある。

・エラーメールの発生

発信アドレスに、中継に使われたメールサーバが管理するアドレスが使用された場合、うまく送信できなかったメールについて、大量のエラーメールが管理者やアドレスを使われたユーザーに送られてくることになる。

・クレーム

架空の発信アドレスを使ってメールが送られた、つまり、発信者にメールを送れない場合は、中継に使われたMTAの管理者（postmaster）宛にクレームのメールが大量に送られてくるだろう。これはReceived：ヘッダを見ればすぐにわかるからだ。

自分の管理するMTAが不正中継に使われた場合、そのサイト自体が迷惑行為を行ったわけではないにも関らず、迷惑メールの送信に協力したサイト、あるいは防止努力をしていないサイトとして、ブラックリストに登録されることがある。次回で説明するが、迷惑メールの被害を防ぐために、受信側で特定サイトやMTAからのメール受信を拒否するようにしているサイトが多い。通常は、不要なダイレクトメールの送信業者などからのメール受信を拒否するのであるが、それに加えて、不正中継の踏み台にされたサイトも含めることがあるのだ。

この制限を行うためのブラックリストに登録されてしまうと、まともなメールも含めて、受信が拒否されてしまい、業務に支障をきたす可能性がある。

▼リスト1　アドレス詐称の例

```
Received: from m-svr.provider.ne.jp (m-svr.provider.ne.jp [210.XX.XX.XX])
        by mail.takobeya.com (8.9.3/3.7W-2.8compat) id QAA00972;
        for <masa@takobeya.com>; Fri, 12 Oct 2001 11:48:34 +0900 (JST)
Received: from PC ([192.168.1.1])
        by m-svr.provider.ne.jp (8.9.3/3.7W-2.8compat) with ESMTP id LAA08114
        for <masa@takobeya.com>; Fri, 12 Oct 2001 11:48:32 +0900 (JST)
Date: Fri, 12 Oct 2001 11:48:30 +0900 (JST)
From: Kiriyama <kiriyama@tdf.int>
Message-Id: <4.2.0.58.J.20011012114830@tdf.int>
To: masa@takobeya.com
Subject: 出撃命令

渋谷区に怪獣が出現した。ただちにホーク1号で出撃せよ。

                                                    隊長より
```

Linux ガベコレ

Linux Garbage Collection

目からウロコの用語辞典

文: しのはらひろあき
Text : Hiroaki Shinohara

第20回
【オフィス】（おふぃす）
【StarOffice】（すたー・おふぃす）
【StarSuite】（すたー・すいーと）
【GNOME Office】（ぐのーむ・おふぃす）
【KOffice】（けー・おふぃす）

オフィス
【おふぃす】

① 編者のような零細自営業者にはまったく縁がないパラダイスのこと。1年中エアコンの効いた快適なスペースで、イスは一脚10万円のアーロンチェア完備。デスクはエルゴノミクス対応。指を鳴らすと一般職の女の子がイソイソとお茶を運んできてくれる。ビシッとキメた部長・島耕作（福岡HSC左遷前）たちが、クールに明日の業界のゆくえについて議論を戦わすなか、ときおり深いスリット入りのスカートをはいた藤原紀香が通り過ぎる。そんな夢のような空間。行ってみたいなオフィス。もちろん、われらがLinux magazineも、このようなオフィスで編集作業が行われているはずである。

そんなにイルカ？

② 急激に進行するハードディスクの大容量化。だが、一部には「そんなものを買っても大きすぎて空っぽのままになってしまうのでは？　不必要だよ」などという疑問の声も。このような指摘を放置しておくと、新しいハードディスクを買いたいと申請しても、家計を厳しく管理する奥さまに却下されるおそれがある。自分はこのあいだ計10万円のドモホルンリンクルセットを購入したくせに、である。

そこで考え出されたのが「空っぽのディスク空間というものは存在せず、ハードディスクは情報伝達のための媒質“オフィス”によって常に満たされている」という仮説である。これを「オフィス理論」、または「オフィス＝エーテル理論」と呼ぶ。

オフィスについての記述は古くから散見され、あのデカルトも「ハードディスクにはオフィスという物質が満ちていて、それが渦を巻き、その渦に突かれて市場は動いている」と述べた。また、オランダのホイヘンスは、情報は波動でありオフィスはそれを伝える媒質として働くという説を提案。これが現在のオフィス理論の原型となっている。その後、オフィス理論はイルカのカイル、冴子先生の力を借りて市場全体を占有しようとする「MSオフィス理論」、とりあえず無料でバラまいてシェアを奪取しようとしつつ、プロジェクトが乱立して共食いをはじめるという「Linuxオフィス理論」へと分化・発達してゆく。

オフィスは「オフィススイート」とも呼ばれ、「ワードプロセッサ」「表計算」「プレゼンテーション」の3つの要素から構成されている。いずれも実際に役に立つのは確定申告締切前夜など、年に一度程度しかないが、なぜこのような役に立たないように思えるものが物理学上重要な位置を占めるに至ったのかについては、いまだ学会の論を待つところである。しかし、「閉鎖空間（＝ハードディスク）内にオフィスを封入すると全体を満たすまで膨張しつづける」という特性は、常に大容量のハードディスクを求めるユーザーとメーカーにとって非常に有用であり、「ねえ、オフィス入れたら

ハードディスク足りなくなっちゃったよ。新しい100Gのやつ買ってきていいよね」などと言いやすくしてくれる救世主なのだ。マイケルソン・モーリーとか、アインシュタインとかの否定的見解は気にしないように。

StarOffice

【すたー・おふぃす】

先ごろ日本語対応版が満を持して登場し話題を呼んでいる、NEC製Webベースグループウェア。業務スタイルを革新・スピード経営を支援する企業情報ポータルを標榜する、まったく新しい情報システムソリューションである。最新のStar Office21 V2.1では、通常のInternet ExplorerやNetscape Communicatorに加えてiモード端末からもアクセスできる手軽さをもついっぽう、ジャーナル機能、ストレージソリューション、クラスタソリューションに対応する高度な信頼性がウリ。もちろん低価格であることも特徴で、300クライアントで250万円からとリーズナブルになっている。お問い合わせはNECまで。間違ってSunに電話をかけないでください。これくらい持ち上げておくと、来月からこのページにNECが4分の1コマくらい広告を掲載してくれるかもしれない。

StarSuite

【すたー・すいーと】

詳しくないのでよくわからないが、たぶん星空の見えるスイートルームのことだと思う。ロマンチックである。しかし、それだけだと最近こんなに騒がれている理由としては足りない。実は星空が見えるというか星空の中、つまり宇宙に浮かんでいるホテルのスイートルームなのではないか。ラグランジュポイントあたりに赤プリ大の宇宙ステーションを設置しておき、想い出に残る一夜をすごそうという男女をスペースシャトルで送り出すのである。

「うわー、ステキな星空ねー！　よしおさん！」
「二人の初めての夜に、ピッタリの部屋だね。よしこさん」
「やだ、恥ずかしい……あら、このキーボードはなに？」
「このステーションの操作パネルだよ。制御システムがJavaでできているから、各部屋から容易に操作できるんだ。ちょっとやってみようか。カチャカチャカチャ…」

ピッ！
Exception in thread "main" java.lang.OutOfMemoryError.

「しまった！暴走だ！」
「キャー！　Ctrl＋Alt＋Delも効かないわ！」

ゴゴゴゴゴ……。

「…ステーション全体が、移動を始めたぞ……」
「怖いわ……。どこへ向かっているのかしら」
「Sun製品だけに、太陽に向かっているようだよ」

（編注：StarSuiteはpure Java製品ではありません）

GNOME Office

【ぐのーむ・おふぃす】

GNOME（ノーム）だけに、徹夜仕事に疲れて寝ているあいだに小人さん（ノーム）たちが現れて、代わりに仕事を片づけてくれるオフィススイート、というネタはどうだろう。イ、イヤっ…ダメだ、ダメだっ……。そんなありきたりの…陳腐なネタではっ……！　そもそも…「GNOME」の読みは「ぐのーむ」っ……！

「カイジ」風に悩む編者。それもそのはず、編者はいまだにLinuxデスクトップ環境を触ったことがなかったのだ。もちろん、そのうえで動くオフィス環境をや。

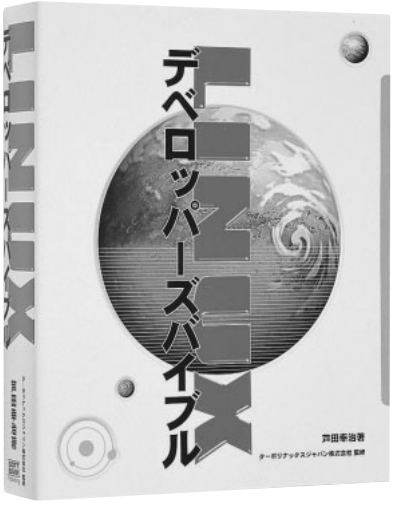
……触ったこともないオフィスの話など…笑止っ……！　ヘタにこの話題に触れれば…オレがSlackwareしか使っていないことが露見してしまうこと…まさに必然っ……！　しかも、そのSlackware環境で…このあいだApache 1.3.22のインストールに失敗っ……。インストールスクリプトに、ルート以下全ファイルをワールドリーダブルへと変更されてしまったことまで…芋づる式にバレるっ……！　くっ…！　これというのもApache Foundation……。やつらがconfig.layoutを変更していたせいっ……！　GNOME Officeなど…この際どうでもいいっ……！（編注：ちっともよくありません。まじめにやってください）

KOffice

【けー・おふぃす】

「KWord」「KSpread」「KPresenter」のみつどもえで、今やすっかり耳にしなくなった「3K」オフィスの実現を狙うオフィススイート。あろうことか、あまつさえ「Kivio」だの「Kontour」「Krayon」、さらに「Kugar」「KChart」とあわせて8Kの達成を目論むにいたっては、労働基準監督署的にそれってOKなの？　と言いたくなる。これ以上劣悪な労働環境で、下々のプロレタリアオフィスワーカーを苦しめるのはやめてほしい。

Books



Linuxデベロッパーズバイブル

芦田幸治 著／ターボリナックス株式会社 監修
ソフトバンク
B5変形判／800ページ　本体価格 4800円

「Linux」ときて「開発」とくると、どうしても話はオープンソースのほうへと進んでしまいがちだ。しかも「バイブル」である。何やらオープンソースによるソフトウェア開発のすばらしさを高らかに謳いあげてしまいそうなタイトルだが、実はそうではなくて、800ページまるごとLinuxにおけるソフトウェア開発技法についての解説書である。

gcc、make、gdbといった伝統的なものから、C、C++、Java、Parlなどのプログラミング言語、ソースコードの管理に使われるCVS、そしてman、linuxdoc用のドキュメント作成に至るまで、とにかくLinuxの開発にかかわる、ありとあらゆるものを網羅的に取り上げている。個々のトピックに関してはその分野の専門書に及ばないものの、通常この手の本では「プログラミングの常識」として省略してしまいそうな事柄についても、ひとつひとつ解説されているので、ソフトウェア開発の技法を学ぶためのテキストとして好適だ。

ファイアウォール&ネットワークセキュリティ実戦テクニック

技術評論社第2編集部 編
技術評論社
B5変形判／416ページ　本体価格 2800円

インターネットへの常時接続環境では、セキュリティを強化して不正侵入などに備えなければならない。これはみなさんもよくおわかりのことと思う。しかし、具体的にどのような対策を講じればよいのかとなると即答できる人は多くない。セキュリティを強化するためには広範な知識と経験、そして最新の情報を集める努力が必要となるからである。

本書は、『Software Design』誌に、1997年から2001年にかけて掲載された数々のネットワークセキュリティ関連の記事に、加筆・再編集を加えて単行本化したものである。近年話題になったアタックの事例を紹介した後、ファイアウォールの構築法や安全にサーバを公開する方法などを具体的に解説している。雑誌記事を元にしているため、全体の構成がやや散漫ではあるが、実戦的な話題を中心とした個々の解説の価値は高い。Linuxマシンをインターネットにつないでいるセキュリティ初級者に勧めたい一冊だ。

TurboLinux 7 Workstationスタートブック

竹田 善太郎、野村 直 著
アスキー
B5変形判／312ページ／CD-ROM2枚付　本体価格 2600円

RPMが何の省略形かご存じだろうか？　ずっと「Red hat Package Manager」だと思っていたのだが、このまえRPMの開発プロジェクトのWebサイト（http://www.rpm.org/）を覗いたところ「RPM Package Manager」と記載されていたのである。正式名の中に自身の省略形が含まれるというヘンなことになっている。いずれにしろ「Red Hat」の文字は外れたようだ。なんでこんな話をしているのかというと、このとき唐突に「単にRPMを採用しているからといって、いつまでも“Red Hat系”と一括りにするのも考えものだなぁ」と思ったりしたからである。実際、Turbolinuxの新バージョンなどは、インストーラも違えば、設定ツールも標準のデスクトップ環境もフォントも……とにかく違うのだ。というわけで、本書のような「専門書」が必要になってくる。入門書としての勘所を押さえた構成と、ノートPCやUSB機器の動作確認など中・上級者にもうれしい情報で幅広い読者に対応する1冊だ。

プログラムはななぜ動くのか

矢沢久雄 著／日経ソフトウェア監修
日経BP社
A5判／296ページ　　本体価格　2400円

商用OSであるWindowsはもちろんのこと、Linuxや＊BSDなどフリーのOSでもKDEやGNOMEのおかげで、プログラムがGUIを備えているのは当然という状況になりつつある。そのため、プログラミングを学ぼうとする初心者にとって、最初の「壁」は昔よりも高くなっているように感じる。また、たとえその「壁」を乗り越えられたとしても、それはある開発環境（たとえばVisual Studioとか）に精通しただけのことで、プログラミング全般に関する知識を得る機会（暇と言ってもいい）があまりないのが現状だ。

本書にはそのような基礎知識、たとえば「CPUができることは？」とか「そもそも2進数って何よ？」といったトピックがちりばめられている。今時の若い人（と言い始めたら年寄り決定）で職業プログラマーを志している人にとって、知っておいて損はないものといえる。読み物としても読みやすく仕上がっており、お勧めである。

はじめてのPostgreSQL

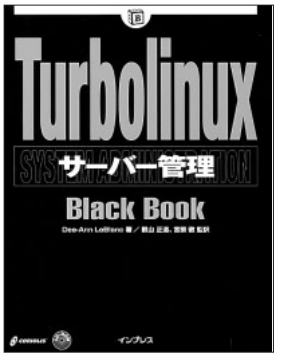
ブルース・モムジャン 著
日本ポストグレスユーザー会 訳

ピアソン・エデュケーション

B5変形判／520ページ

本体価格 4200円

Turbolinuxサーバ管理 Black Book



Dee-Ann LeBlanc 著
観山正道、宮原 徹 監訳

インプレス

B5変形判／580ページ／CD-ROM2枚付き

本体価格 3800円

NFSバイブル

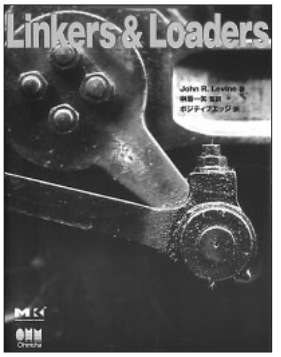
Brent Callaghan 著
株式会社クイープ 訳

アスキー

B5変形判／504ページ

本体価格 5200円

Linkers & Loaders



Jhon R. Levine 著
榊原一矢 監訳
ポジティブエッジ 訳

オーム社

B5変形判／256ページ

本体価格 3200円

読者の声

俺にもいわせろ！

担当のアパートのお隣さんは、つるが家全体を覆い尽くしている、森のようなお家です。お隣さん宅に憧れた担当は、お隣から伸びてきたつるを、つるの気が向くままにしていました。1年が経ち、つるがいい感じになってきたなと思ってきた矢先。突然、大家さんにむしり取られてしまいました。なんでも、お隣のつるが原因で、うちのアパートが停電したことがあるとか……。森に住むって大変！

ご好評/etcディレクトリ

またまたファンタジックな特集の登場ですね(笑)。毎回、このシリーズは目から鱗なので助かります。ありがとうございます。このファンタジックなキャラクターが登場するシリーズは、全体を通すとファンタジーなストーリーがあったりします(笑)？

（伊藤真和さん）

11月号の/etc特集が楽しかった。あそこまで詳しく書かれていると、ついつい読んでしまって……。勉強になりました。

（馬場秀一さん）

/etc特集は役に立ちました。でも、書式や記述例まで記載してあるファイルは、一部しかありません。雑誌の一特集としては、十分と思いますが、今回項目が挙がったファイルについて、すべての書式解説と記述例が記載された本があると良いです（本屋へ行っても、見つかりません。需要がないのか）。個人的には、xntpdの使い方がわからず（英語マニュアルが、全然読めず）苦戦しているところです。

（山下幹雄さん）

/etc以下のファイルは、ディストリビューションやバージョンによって異なりますので、すべての書式解説と記述例が記載された本は、おそらくないだろうと思います。それよりも、目的別に資料を探したほうが、より詳しく、より新しい情報を得ることができるでしょう。

山下さんがお困りのxntpdとは、ネットワークを介して時計を合わせるためののデーモンプログラムで、設定ファイルはntp.confです。記述方法に関して、好評発売中のムック「Linuxインターネットサーバ構築ガイド」の「タイムサーバを構築しよう」で掲載しています。こちらも参考にしてみてください。

忙しい！

創刊号から読んではいるのですが、Linuxを自由に操るにはなかなか至りません。何かするにも、付箋をはさんだ、バックナンバーを引っかき回しています。本業が忙しく、月に5〜6時間程度しかLinuxを触れないのが原因なのですが。つい、イージーなWindowsで事を済ませてしまいます。今使っている、マウス・キーボード・モニタ切り替え機が、まもなくいかれてしまいそうです（マウスポインタがたまに吹っ飛んでしまう。ボールなどをきれいに掃除しても）。ちょうど、プリンストンのが欲しかったので、当たりますように。

（安達正晃さん）

いつも楽しく、真剣に、ぐうたらに拝読いたしております。いくつかのディストリビューションを使用しておりますが、どれも中途半端になってしまい、使いこなしていないなあ、と思っています。十分使いこなしたなあと思うまで、長くお付き合いさせてくださいませ。

（小林敏彦さん）

担当も、やりたいことがたくさんあって、バックグラウンドを使えたらなぁ……なんて思ったりします。でも、いざ暇な時間ができても、ぼけぇ〜っと過ごしてしまうんです。いつまで経っても、進歩のない困り者。

血まみれ

もらったメモリ（64Mバイト）の増設を図る。メーカー製PCなので分解しにくいうえに中に手を入れにくく、ケーブルのジャングルに手を突っ込んで作業していた。気がつくと手が血まみれに。どうやらヒートシンクで切った

模様。こわいこわい（しかもメモリは壊れていた）。

（藤原 勤さん）

パソコンに引っかかれるとは、災難。担当は、最近飼い猫に引っかかれて血まみれです（TT）。

今と昔

電磁界解析や構造解析などの数値計算を数年来ずっとやっている者です。一昔前は汎用機やワークステーションでしか計算できず、計算時間とお金をにらめっこしてました。

今ではLinuxのおかげでほとんどお金をかけずに当時をしのぐ仕事ができてしまいます。皆さん（特に若い人）は当然のことと思ってらっしゃるかもしれませんが、本当にすごいことですよね。Linux開発に携わる人々は何で食べてらっしゃるのでしょう？

（T．Mさん）

以前といっても12年ぐらい前（大阪にいた頃）、マイクロVAX Ⅰや当時発売されたばかりのNEWS（SONY）を使用していましたので「UNIXとPDP-11」を懐かしく読ませていただきました。

今では考えられないことですが、当時CADのデータをテープに落とし、車で阪神高速を通り、1時間半〜2時間の道のりを客先（奈良県）まで運んでいました。昔は仕事の締め切りに間に合わず徹夜したこともあり、その当時が懐かしく思い出されました。

（西山正起さん）

高速道路を、1時間以上かけて原稿を運ぶ必要のない今でも、編集部では締め切り前に徹夜組み続出です。便利なものを発明すると、新しい仕事も発見されちゃうんですね。

Linuxに関わるサービスを提供している会社は、ディストリビューション販売、ITアウトソーシング、教育、商用ソフトウェアの開発・販売などもしているようですね。

FTP版と製品版

ディストリビューションやアプリケーションなどもなるべく製品版を購入してLinuxの発展に貢献できるようにしたいと思っています。まだまだバグが多いようです。

ところで、パッケージの価格について感じることがあります。多くのディストリビューションは、サポート費用がパッケージ自体に含まれているようです。しかし、私の場合、基本システムそのものは、FTP版が手に入るので、自分の環境で使用できることを確認してから購入するようにしています。こうすると、商用ディストリビューションを購入するのに、不要なサポートまで購入することになり、不合理です。FTP版から、サポートなしのアップグレード版が手に入れば、安く済むはずなので、各ベンダーは一考していただきたいと思います。

（鈴木 茂さん）

今月号のCDの付録のTurbolinuxは、大変嬉しかったです。パッケージを購入するか否かで迷っていました。実際にインストールして評価してから購入するきっかけができます。ありがとうございました。

（遠藤浩二さん）

製品版LinuxとFTP版Linuxの違いは、サポート、マニュアル、商用フォントやソフトの有無です。また、FTP版Linuxでも、インストール中にアップグレードインストールを選択することでアップグレードが可能です。これらが不要な方で、かつ、FTP版をダウンロードして、インストールできる方にとっては、製品版でもFTP版でも変わりはありません。

ただ、最近のディストリビューターによるサポートは、以前とは違い、質問回数や範囲が無制限になったり、サポート通話料が無料だったりと、ずいぶん利用しやすくなっているようです。マニュアル本も親切なだけではなく、より読みやすくなっているものが多いです。Linuxの操作に不慣れな方、段階的に学習したい方、サポートを必要とされる方には、製品版Linuxがお勧めです。

インストールの後は？

Windowsしか触ったことのない私にとってはLinuxのインストール手順が非常に難解でした。そんな私ですがLinux magazineの内容は頑張ればついていけそうだったので、頑張ってみようかなと思っています。ですから初心者を見捨てたりしないでくださいね。

（藤井順一さん）

先日、はじめて雑誌付録よりLinuxをインストールしましたが、その後、何をしてよいかわからず、本誌にて勉強中です。

（たまちゃんさん）

Linuxのインストールが完了したらまず、インターネットに接続して、家庭内でLANを組んでみましょう。好評発売中のムック「Linuxインターネットサーバ構築ガイド」や、12月号の特集1・2を参考にしていただいてはどうでしょうか？

付録CD-ROMに収録した
MIRACLE LINUX Standard Edition Version2.0 RC3のインストール

本誌付録のMIRACLE LINUX Standard Edition Version2.0 RC3（以下、MIRACLE LINUX）は正式版リリース前のRC3です。非商用ソフトだけが含まれており、製品版を販売しているミラクル・リナックス株式会社や編集部からのサポートは受けられません。

また、MIRACLE LINUXのバイナリはPentium Pro以上のCPUに最適化されているため、PentiumやAMDのK6-2といった古いCPUを搭載するマシンではインストーラが起動しません。Intel Pentium Pro、AMD Athlon以上を搭載したマシンでお使いください。

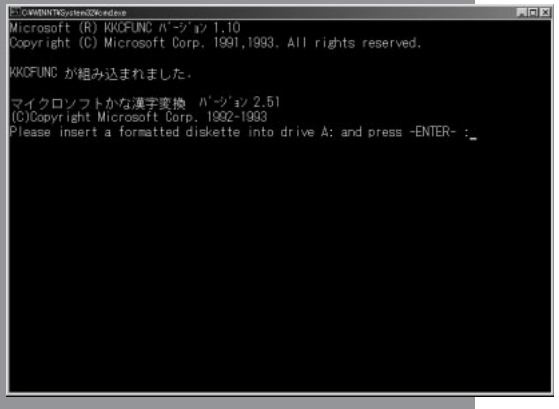
なお、CD-ROMメディアに不良があった場合は、お手数ですが（linux-cd@ml.ascii.co.jp）宛にご連絡くださるようお願いいたします。Linuxのインストールや設定などについてのご質問にはお答えできませんので、あらかじめご了承ください。

1 インストールの前準備

これからMIRACLE LINUXのインストールを始めます。多くのネットワークカード、ビデオカード、モニタは自動認識されますが、自動認識されない場合に備えて、あらかじめハードウェアのマニュアルなどを用意しておくと、インストール中にあわてなくてすみます。

また、レスキュー用のフロッピーを作成するために、空のフロッピーを1枚用意しておいてください。CD-ROMブートできないマシンを使う場合は、別にインストーラ起動用のフロッピーも用意します。

さて、Linuxのインストールには、Linux専用に使えるディスク領域が必要です。ハードディスクでLinux専用に使える領域を作成するか、ディスクを増設するかしてインストールに備えましょう。

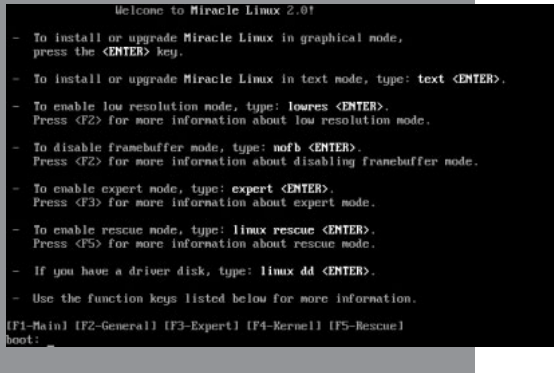


2 ブートディスクの作成

CD-ROMからインストーラを起動できない場合は、以下の手順でインストーラ起動用のフロッピーディスクを作成します。

まず、空のフロッピーをドライブにセットします。次に、Windowsのエクスプローラでcd-romの中の［dosutils］フォルダを開き、［boot.bat］というバッチファイルをダブルクリックします。

DOS窓が立ち上がり、［Enter］を押すとインストーラ起動用フロッピーの作成が始まります。



3 インストーラの起動

作成したフロッピーディスク、またはCD-ROMをドライブにセットしてマシンを再起動します。インストーラが起動して［boot:］というプロンプトが表示されたら、[Enter]を押します。しばらくすると、Xを使ったグラフィカルな画面が表示されます。

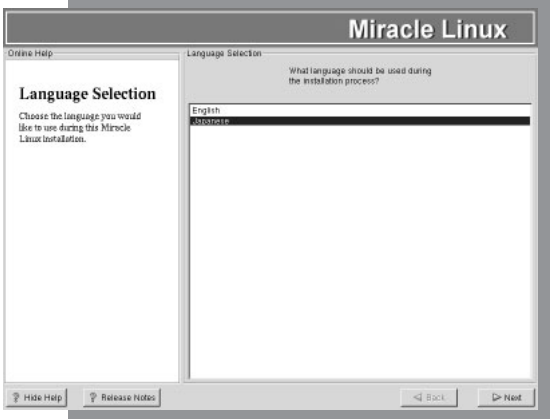
グラフィカルなインストール画面がうまく表示されない場合は、［boot:］の箇所で［text］とタイプして[Enter]を押します。こうすると、テキスト画面のインストーラが起動します。

本誌の付録CD-ROMは、通常の650Mバイトメディアではなく、700Mバイトのメディアを使っています。4倍速以下など一部CD-ROMドライブでは正常に読み込めない場合があります。あらかじめご了承ください。

使用言語の選択

インストーラを英語表示させる場合は［English］を、日本語表示させる場合は［Japanese］を選択します。ここでは、［Japanese］を選択します。

4



キーボードとマウスの設定

デフォルトでは［モデル］に［Japanese 106-key］が、［レイアウト］に［Japanese］が選択されています。日本語キーボードを使用する場合は、このままの状態で［次］を押します。

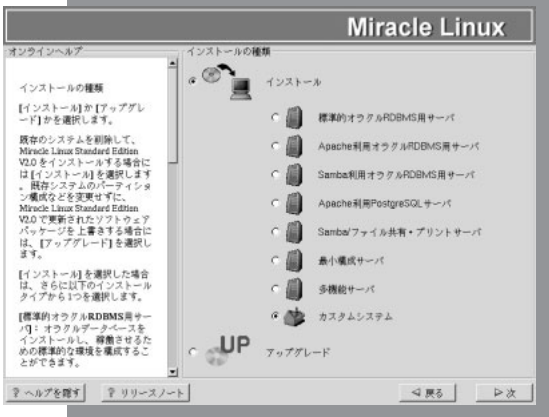
次はマウスの設定です。PS/2タイプの2ボタンマウスを使う場合は［2 Button Mouse（PS/2）］をチェックします。［3ボタンマウスのエミュレート］は、2ボタンマウスの左右ボタンを同時に押すことで、3ボタンマウスの中ボタンを代用する機能です。これは便利な機能なので、チェックしたまま［次］を押します。

5

インストールタイプの選択

［標準的オラクルRDBMS用サーバ］などのインストールタイプを選択できます。インストールタイプのそれぞれについては画面の左側に表示される［オンラインヘルプ］の解説を参照していただくとして、ここではもっとも柔軟にインストールできる［カスタムシステム］を選択したものとしてインストール解説を進めます。

6



パーティション設定ツールの選択

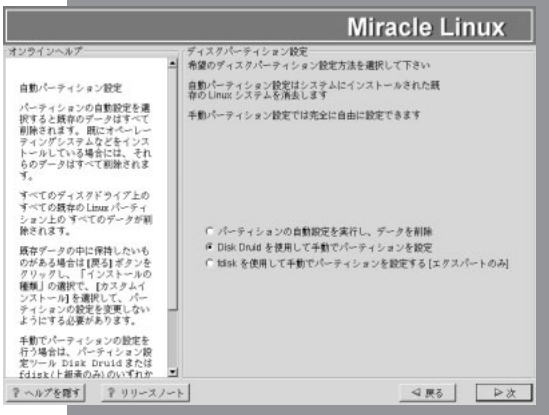
［fdisk］は柔軟にパーティションを設定できますが、コマンドで操作するので初心者にはお勧めできません。

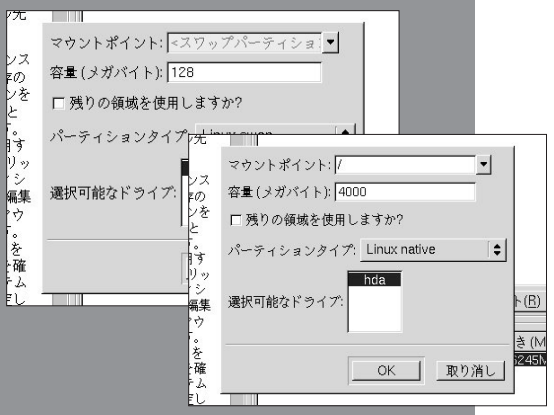
［Disk Druid］はやや柔軟性に欠けるものの、マウスを使ってパーティション設定できるので初心者にお勧めです。

［パーティションの自動設定を実行し、データを削除］を選択すると、ハードディスクのデータやパーティション情報がすべて削除され、自動でLinux用のパーティションが作成されます。

ここでは［Disk Druid］を選択したものとして解説を進めます。

7





パーティションの作成

Linuxのインストールには、最低限Linuxシステム用とSwap用の2つのパーティションが必要です。

まずはLinuxシステム用のパーティションを作成します。［追加］を押すと、小さいウィンドウが別に開かれるので、［マウントポイント］に［/］を、［パーティションタイプ］に［Linux native］を選択してます。MIRACLE LINUXをフルインストールすると、ディスクを約1.4Gバイト消費します。運用開始後のことも考えて、Linuxシステム領域を3～4Gバイト程度作成して［OK］を押します。

次にSwap用パーティションを作成します。［追加］を押して開かれるウィンドウで、［マウントポイント］にはなにも指定せずに、［パーティションタイプ］に［Linux swap］を選択して、マシンが搭載するメモリの1～2倍程度のサイズを指定して［OK］を押します。

2種類のパーティションを作成したら、［次］を押します。

パーティションのフォーマットとLILOの設定

Linux用にフォーマットするパーティションを選択します。［フォーマット中に不良ブロックをチェック］をチェックすると、ハードディスクに不良箇所がないかどうかを調べながら、パーティションをフォーマットします。この欄は特にチェックする必要はないでしょう。選択したら［次］を押します。

次にブートローダLILOのインストール先を設定します。まず、IDEタイプのハードディスクを使う場合は、［リニアモードを使用する（一部のSCSIドライブで必要）］のチェックをはずします。マシンをLinux専用に使う場合と、Windows 9x／MeとLinuxをデュアルブートする場合は、LILOのインストール先に［/dev/hda マスターブートレコード（MBR）］を指定します。Windows NT／2000とLinuxを共存させる場合は、LILOをインストールせずにフロッピーディスクからLinuxを起動します。

［ブートディスクを作成］は必ずチェックしておきましょう。

ネットワークとファイアウォールの設定

MIRACLE LINUXをDHCPサーバからネットワーク情報を取得する環境で使う場合は、［DHCPを使用して設定］を、DHCPサーバのない環境で使う場合は［DHCPを使用して設定］のチェックをはずして、IPアドレスなどのネットワーク情報を入力します。なお、この画面ではダイヤルアップ接続の設定はできません。

次にファイアウォールの設定です。セキュリティの設定に詳しくないユーザーは、［セキュリティレベルを選択して下さい］で［中］を選択してから、［次］を押してください。なお、「lokkit」コマンドを使えば、インストールしたあとでもファイアウォールを再設定できます。

使用言語とユーザー設定

MIRACLE LINUXで使用する言語を選択します。MIRACLE LINUXを日本語環境として使う場合は、デフォルトのまま［次］を押します。

次の［タイムゾーンの選択］画面もそのまま［次］を押します。

さて、今度はユーザー設定です。Linuxには大きく分けて、ソフトウェアのインストールなどシステム管理を行う特権ユーザー［root］と、Webブラウズやメールの読み書きなど一般的な作業を行う一般ユーザーの2種類が存在します。

まず、Linux管理者のパスワードとして、［root パスワード］と［確認］に同じものを入力します。この管理者のログイン名は［root］です。

次に一般ユーザーのアカウントを設定します。［アカウント名］（＝ログイン名）、パスワードとして［パスワード］と［パスワード（確認）］に同じものを、［フルネーム］にユーザーのフルネームを入力して、［追加］を押します。ユーザー設定を終えたら［次］を押します。［認証設定］もそのまま［次］を押します。

パッケージグループの選択

パッケージがいくつかのグループに分類されています。どのグループを選択してよいのかわからないユーザーは、最下段の［すべて］を選択して、すべてのパッケージをインストールしておくとよいでしょう。なお、［すべて］を選択した場合は、ハードディスクを1.4Gバイト程度消費します。

ここでは［すべて］を選択したものとして解説を進めます。

ビデオカードとモニタの設定

多くのビデオカードは自動で認識されます。インストーラが自動認識に失敗した場合は、リストの中から使用するビデオカードを選択します。

次はモニタの設定です。モニタの自動認識に失敗した場合は、モニタのマニュアルを参考にしながら、［Generic］の中からモニタのスペックを超えない［Generic Monitor］を選択します。なお、液晶モニタを使用する場合は、［Generic Laptop Display Panel］から無理のない解像度のものを選択します。

Xの設定

X Window Systemを綺麗に表示させるには、［色深度］で［High Color（16 Bit）］以上を選択する必要があります。［画面の解像度］には、モニタに合わせて適当な解像度を指定します。

［色深度］と［画面の解像度］を選択したら、［設定のテスト］を押してXの表示テストを行います。GNOMEのデスクトップ画面が表示されればXの設定は成功です。GNOMEのデスクトップ画面が表示されない場合は、［色深度］や［画面の解像度］の数字を低くしてテストしてください。

［ログインの種類を選択して下さい］で、［テキスト］を選択するとテキストベースのログイン画面が、［グラフィカル］を選択するとグラフィカルなログイン画面が表示されます。ただし、Xの表示テストに成功してないユーザーは［グラフィカル］を選択してはいけません。

［次］を押すとパッケージのインストールが始まります。

ブートディスクの作成

ハードディスクからOSを起動できなくなった場合に備えて、緊急時用の起動ディスクを作成します。空のフロッピーディスクをセットして［次］を押すと、ディスクの作成が始まります。

起動ディスクの作成が終わるとインストール作業は終わりです。フロッピーディスクを抜いて、［終了］を押してください。CD-ROMは自動で排出されます。

お疲れさまでした。