コンパックがLinuxベースのアプライアンスサーバ市場に参入 Compaq Linuxソリューションパッケージシリーズ

コンパックコンピュータは、Linuxを搭載したアプライアンスサーバ「Turbolinux Web/Mail Serverモデル」（以下Web/Mailモデル）と「Miracle Linux File/Print Serverモデル」（以下File/Printモデル）の2モデルを、3月26日に発売した。

価格は、Web/Mailモデルが79万8000円、File/Printモデルが75万8000円。両機種ともコンパックのLinuxシステム管理者向けサポートサービスである「CarePaqソフトウェアサポートLinux用Tier3 1年間回数無制限」をバンドルしている。

両モデルともProLiant ML350をベースにしており、PentiumIII 1GHz、256MバイトECC SDRAM、ホットプラグ対応18.2Gバイト10000rpm SCSIハードディスク×3台、Smartアレイ431コントローラ、20/40 DDS-4 DATドライブなどを搭載している。

Web/MailモデルのOSは、Turbolinux Advanced Server 6で、Webブラウザからサーバの設定／管理が可能なLinux Controller Turbolinux Editionが搭載されている。

File/PrintモデルのOSは、Miracle Linux for Samba Version 1.0で、LinuxをWindowsのファイル／プリントサーバとして利用可能にするSambaを組み込んでいる。ファイル共有／プリンタ共有の設定は、SambaのWeb管理ツールであるSWATを通して容易に行える。

発売日	2001年3月26日
発売	コンパックコンピュータ株式会社
TEL	0120-101589
価格	75万8000円〜
URL	http://www.compaq.co.jp/

Hardware　　発売日　2001年3月6日

Linux用オフィスパッケージ「Do Office」をバンドルしたノートPC はじめてみようLinuxパック

URL　http://value-shop.hitachi.co.jp/

発売　株式会社日立製作所 日立Internet Shop
TEL　0120-64-2073
価格　21万4000円〜

日立製作所は、レッドハットのLinux版オフィスパッケージ「Do Office」を、B5ファイルサイズノートPC「FLORA 220TX」および「FLORA 220FX」にバンドルした「はじめてみようLinuxパック」を3月6日より発売した。

FLORA 220TXは、トランスメタのCrusoeプロセッサ（533／600MHz）、64Mバイトメモリ（システムで16Mバイトを使用）、10Gバイトハードディスク、12.1インチTFT液晶（1024×768ピクセル、65万色）、LAN、モデムを搭載。同220FXは、CPUにモバイルCeleron 600MHzまたはモバイルPentiumIII 600MHzを搭載している。

Do Officeは、Vine Linux 2.1とHancomLinuxのHancom Office 1.2を1つのパッケージにした製品で、ワープロ／表計算／プレゼンテーション／ペイントなどのソフトが含まれる。

プレインストールされているのはWindows 98SEで、Do Officeを利用してWindowsとLinuxのデュアルOS環境を簡単に構築できる「Linuxインストール手順書」が添付される。

Hardware　　発売日　2001年3月12日

グループウェアを搭載したLinuxサーバの特別モデル
PowerEdge 1400 サイボウズOffice 4プリインストール・モデル〜ボウズマンサーバ〜

URL　http://www.dell.com/jp/

発売　デルコンピュータ株式会社
TEL　044-556-6190
価格　20万6000円〜

デルコンピュータは3月12日より、同社のタワー型エントリーサーバ「PowerEdge 1400」に、サイボウズのグループウェアの最新版「サイボウズOffice 4」をプレインストールした特別モデル「PowerEdge 1400 サイボウズOffice 4プリインストール・モデル〜ボウズマンサーバ〜」を発売した。

推奨構成は、CPUにPentiumIII800MHz、メモリ128Mバイト、9GバイトSCSIハードディスクを搭載する。デュアルプロセッサにも対応し、CPU／メモリ／各種I/Oコントローラはアップグレード可能。OSはRed Hat Linux 7Jがプレインストールされる。

価格はプレインストールされる「サイボウズOffice」製品によって、サイボウズ Office SOHO4（10ユーザー）の場合20万6000円から、サイボウズOffice パックEX4（50ユーザー）の場合30万8000円から、同100ユーザーの場合46万4000円からとなっている。

Hardware　　発売日　2001年3月20日

Webサーバの負荷を減らすネットワークキャッシュ専用装置 ネットワーク・キャッシュ・サーバ

URL　http://www.tocc.inetcan.com/

発売　株式会社トラフィック・ワン・コミュニケーションズ
TEL　03-5276-8178
価格　98万円〜

三菱電機とトラフィック・ワン・コミュニケーションズは、日本IBMのPCサーバにInktomiのキャッシュサーバソフトを搭載した「ネットワーク・キャッシュ・サーバ」を3月20日より発売した。

キャッシュサーバを利用することで、Webサーバへのトラフィック量を減らし、Webサーバの負荷を低減できる。プロキシサーバの機能も含んでおり、今後音声や動画などのマルチメディアのキャッシュ機能をオプションで提供する予定。

ベースとなるPCサーバは、日本IBMのeServer xSeriesで、タワー型の200シリーズと1Uラックマウントの300シリーズが用意されている。OSにRed Hat Linux 6.2Jを採用し、キャッシュサーバソフトにはInktomi Traffic Server 4.0を搭載している。

Hardware　　発売日　2001年3月8日

HPのテープバックアップ装置とNetVault 6をバンドルした Linux用バックアップ・ソリューション

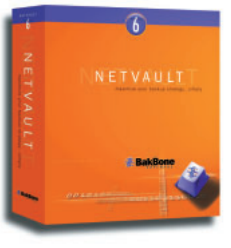
URL　http://www.nissho-ele.co.jp/

発売　日商エレクトロニクス株式会社
TEL　03-3544-8235
価格　25万6000円〜（キャンペーン価格）

日商エレクトロニクスは、バックボーン・ソフトウエアの「NetVault 6」バックアップ／リストアソフトウェアを、日本ヒューレット・パッカードの「hp surestore」テープバックアップ装置にバンドルした「Linux用バックアップ・ソリューション」を、3月8日より発売した。

NetVault 6は、テープデバイスの転送速度に最適なメモリバッファを割り当て、データ転送ブロックサイズを調整することで高いパフォーマンスを発揮するバックアップ／リストアソフト。ネットワーク経由で利用することが可能。LinuxのほかUNIX、Windows NT／2000に対応する。

hp surestoreシリーズのDDS-3、DDS-4、DLT1、UltriumドライブにNetVault 6をバンドルした10種類のパッケージが用意され、3月8日から5月末日までの期間は、発売記念キャンペーン価格で提供する。



Hardware 発売日 2001年3月12日

セキュリティホール対策を施したWebアプリケーションサーバ Web Application Optimized Server

URL http://www.plathome.co.jp/

発売 ぷらっとホーム株式会社
TEL 03-3251-2600
価格 46万円〜

ぷらっとホームは、インターネット・プラグイン・サーバシリーズの第1段として、Webアプリケーションサーバ「Web Application Optimized Server」を3月12日より発売した。

Webサーバ用ソフトがプレインストールされ、セキュリティホール対策が行われているアプライアンスサーバ。レッドハットの協力を得て、納入後、新たに発生したセキュリティホール対策を行うソフト保守サービス（1年目無償、2年目以降有償）が提供される。

2UラックマウントタイプにはPentiumIII 850MHz（最大1GHz、デュアルプロセッサ対応可能）、メモリ256Mバイト（最大2Gバイト）、18.2GバイトSCSIハードディスク（最大4台）を搭載した標準モデルと、RAIDハードディスク搭載モデルが用意される。

1UラックマウントモデルはPentiumIII 750MHz（最大850MHz、デュアルプロセッサ対応可能）、メモリ128Mバイト（最大1Gバイト）、18.2GバイトSCSIハードディスク（最大2台）を搭載する。

Software 発売日 2001年3月23日

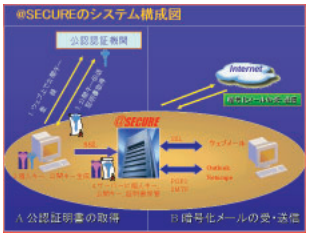
Webベースの暗号化メールサーバソフト @SECURE

URL http://www.3rsoft.com/jp/

発売 スリーアールソフト株式会社
TEL 03-5330-8851
価格 25万円（20ユーザー）〜

スリーアールソフトは、メールの暗号化や電子認証をサポートしたWebベースの暗号化メールサーバソフト「@SECURE」を3月23日より発売した。SolarisなどのUNIXおよびTurbolinux、Red Hat、Miracleなど主要なLinuxディストリビューションに対応する。

@SECUREは、米RSAのPKIベースの暗号化アルゴリズムを採用し、個人キー、公開キー、データの保管や処理などすべてをサーバ側で処理するため、ユーザーはWebブラウザだけで暗号化メールの送受信が可能である。@SECUREユーザー間だけでなく、Outlook Express／Netscape Messenger／Notesなどとも暗号化メールを送受信できる。また、公認認証機関発行のデジタル証明書を使えるほか、内部のmini CAで私設認証書を発行することも可能。このほか、アドレス帳やスケジューラ、掲示板、個人ファイルマネージャなどの機能を加えている。



Software 発売日 2001年3月27日

iDCの専用サーバホスティングサービス向けサーバマネージメントソフト HDE Management Suite for iDC

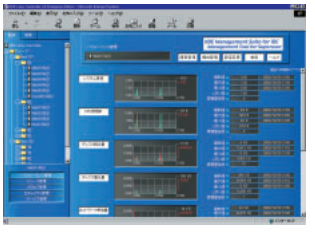
URL http://www.hde.co.jp/

発売 株式会社ホライズン・デジタル・エンタープライズ
TEL 03-5456-3260
価格 オープンプライス

ホライズン・デジタル・エンタープライズ（HDE）は、iDC（インターネットデータセンター）の運営に最適化したLinuxサーバ統合マネージメントソリューション「HDE Management Suite for iDC」を、3月27日より発売した。

iDCが顧客ごとに専用サーバをレンタルする「専用サーバホスティングサービス」を行う場合のサーバ運用管理に適した製品。iDCのシステム管理者が複数のLinuxサーバを一元管理する機能と、ユーザー管理者がレンタルしたマシンを管理する機能の両方を備えていて、Webブラウザから操作が行える。iDC管理者は、ユーザーが管理できる権限の範囲を詳細に設定することが可能。

Red Hat Linux 6.2JとTurbolinux Server日本語版6.1に対応し、価格はオープンで、1台あたりの管理コストとして5万円からとなっている。



Software 発売日 2001年4月5日

PC NFSクライアントアプリケーションの最新版 NFS Maestroシリーズ v7.0J

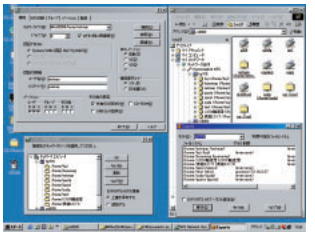
URL http://www.macnica.co.jp/

発売 株式会社マクニカ
TEL 045-476-1960
価格 2万9800円〜

マクニカは、加HummingbirdのPC NFSクライアントアプリケーションソフトの最新バージョン「NFS Maestroシリーズ v7.0J」を4月5日より発売した。

Windows 95／98／NT 4.0／2000上からUNIX（Linux）のファイルシステムにアクセスすることを可能にするソフトで、NFS基本機能を備えた「NFS Maestro Solo v7.0J」が2万9800円。各種ユーティリティ機能を付加した「NFS Maestro Client v7.0J」が5万4000円。

新バージョンでは、ファイル名／ファイル属性のキャッシュ機能を強化して高速なファイルアクセスを実現、NIS＋をサポート、Microsoft Windows Installer対応などが強化された。



News Headline from 日刊アスキーLinux

www.Linux24.com

シャープ、携帯情報端末にJavaを採用――海外向けZaurusにはLinuxを搭載！

2001年3月23日

シャープは3月23日、PDA／携帯電話機などの情報端末機器に、米サン・マイクロシステムズの開発したJava技術を採用すると発表した。あわせて、同社のPDA「Zaurus（ザウルス）」用のJavaソフト実行環境を開発者向けにオンライン販売すると発表した。

海外向けのZaurusには、現行のZaurus専用OSではなくLinuxを搭載。ただし、開発されたJavaプログラムは、国内／海外両方の機種で動作可能にするという。そのほか、情報端末機器以外にも、ファクシミリ／コードレス電話機などの家庭用通信端末や、テレビ／ビデオ／ステレオなどのAV機器にもJava実行環境を搭載することで、互いに情報を共有し、アプリケーションの共通化を図り、GUIを統一するSDE（Seamless Digital Environment）が実現できるように製品開発を行っていくという。

Zaurus向けのJava実行環境は、「Zaurus用ソフト実行環境（for PersonalJava）」という製品名で、同社のWebサイト“シャープスペースタウン”を通じて、4月4日から開発者向けに3000円でオンライン販売する。英Tao社（ダオ）が開発した技術“intent”を採用し、Javaプログラムの実行では高速化と省メモリ化を図った。intentは、コンパイル型エンジンのため、従来のインタープリタ型に比べ、10倍以上高速にJavaプログラムを実行できるという。

同製品の動作環境は「Zaurus MI-E1」で、16Mバイト以上のコンパクトフラッシュカードと、インストール用にWindows 95／98／2000搭載のパソコンが必要になる。動作可能なJavaプログラムは、PersonalJava 1.1.3準拠のもの。

国内外のZaurus開発者向けに、Webサイトなどを通じてハード／ソフトの情報提供を行い、ユーザーコミュニティサイトの運営を行う。すでに国内では、開発者向けのフォーラムをWeb上に開設。一方、海外では、6月にサンフランシスコで開催する“JavaOne”への同社出展に合わせて、ユーザーコミュニティサイトを開設する。

シャープ（http://www.sharp.co.jp/）

Zaurus開発者向けフォーラム

（http://more.sbc.co.jp/pjzaurus/pjzaurus.asp）

レーザーファイブとトラストガード、インターネットサーバ分野で提携

2001年3月22日

レーザーファイブは、サーバ向けハードウェアベンダーのトラストガードとインターネットサーバ分野において業務提携を行ったと発表した。

提携第1弾の製品として、キャッシュアプライアンスサーバを開発中だという。同製品は、リバースキャッシュプロキシサーバとしてWebサイト側の負荷を軽減するもの。LASER5 Linux 6.4ベースのキャッシュサーバ構築用ディストリビューション「LASER5 Linux Cache Server Edition」を、トラストガードの「TurstGuard/iSV」に実装し、「TurstGuard/iSV-CacheServer」としてリリースする。

レーザーファイブは、昨年後半から、特定用途に絞り込んだ高付加価値の“カスタムディストリビューション”を中心としたソリューション事業を重視しており、今回、従来からLinuxを中心としたデータセンター向けサーバで定評のあったトラストガードと思惑が一致したという。両社は今後、ファイアウォールサーバ、負荷分散サーバ、レスアドミニストレーションシステムなどの共同開発を予定している。

レーザーファイブ

（http://www.laser5.co.jp/）

トラストガード

（http://www.trustguard.co.jp/）

ネオ・ロジスの運送業者支援システムにTurbolinuxを採用

2001年3月15日

ターボリナックス ジャパンは、ネオ・ロジスの物流業界向けASPシステム「運送業者支援システム」に、サーバ用OSとして「Turbolinux Server 日本語版 6.1」、同システム専用クライアントOSとして「Turbolinux Workstation 日本語版 6.0」が採用されたと発表した。

クライアント環境には、今回システム構築を担当したブレイン・ハーツのLinux用GUI設計ツール「iface」を採用している。

ifaceでは、ソリューションごとにソフトウェアや画面構成などを自由に設計できるため、クライアントに必要な機能に限定して設計することで、Linuxベースでも簡単な操作が可能になるという。

本格的なサービス開始は2001年7月になる予定。ネオ・ロジスでは3年間で1000件以上の導入を目指しているという。

ターボリナックス ジャパン

（http://www.turbolinux.co.jp/）

ブレイン・ハーツ

（http://www.brainhearts.co.jp/）

東芝エンジニアリングがHard Hat Linuxを技術支援

2001年3月14日

モンタビスタソフトウエアジャパンは3月13日、東芝エンジニアリングとソリューション提供に関して業務提携したことを発表した。

東芝エンジニアリングは、モンタビスタソフトウエアジャパンが提供する組み込みシステム向けのLinuxであるHard Hat Linuxに対して、ミドルウェア／デバイスドライバやカスタムハードウェア設計の面での技術的支援を行う。

具体的には、Bluetoothを始めとする各種通信プロトコル用のドライバやブラウ

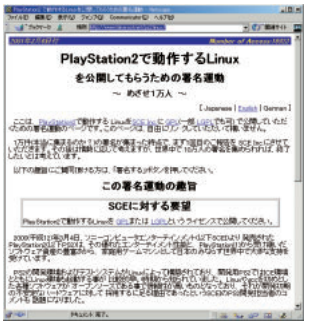
ザなどのインターネット・ミドルウエア開発、およびカスタムボードの統合を行い、Hard Hat Linuxのソリューションの拡大を図る方針であるという。

モンタビスタソフトウエアジャパン
(http://www.montavista.co.jp/)
東芝エンジニアリング
(http://www.toshiba-eng.co.jp/)

プレイステーション2用Linuxの公開を嘆願するWebページ登場
2001年3月5日

プレイステーション2で動くLinuxをGPLまたはLGPLで公開してもらうための嘆願書を、ソニー・コンピュータエンターテインメントに提出するための署名運動が開始された。署名の目標は1万人で、この人数に達したらソニー・コンピュータエンターテインメントに提出するという。

PlayStation2で動作するLinuxを公開してもらうための署名運動
(http://www.fakeroot.net/ps2linux/)



ミラクル・リナックス、SRAとPostgreSQLサポートで業務提携
2001年3月1日

ミラクル・リナックスは、「Miracle Linux for PostgreSQL Version 1.0」の出荷開始に伴い、PostgreSQLのサポートなどでも知られるSI企業SRAとPostgreSQLに関する包括的な業務提携を行った。

また、Miracle Linux for PostgreSQL出荷に伴い、3月8日（木）～4月19日（木）の期間中毎週木曜日の夜6時30分から開催する、ミラクル・リナックス主催のLinux関連講座「MIRACLE de Night 2001」において、Miracle Linux for PostgreSQL製品の詳細情報や導入事例などを紹介する予定だという。さらに、サンブリッジテクノロジーズ主催、ミラクル・リナックス協賛の「スパイラルセミナーin SVH」では、PostgreSQLからOracleへの移行に関する情報を提供するという。

ミラクル・リナックス
(http://www.miraclelinux.com/)
SRA（http://www.sra.co.jp/）

レーザーファイブ、IT企業のためのLinux教育「LPI準拠トレーニング」をスタート
2001年3月1日

レーザーファイブは、Linuxおよびネットワーク技術者養成教育を行うドットネットと協力し、LPI（Linux Professional Institute Japan）の教育内容に準拠したLinux技術者トレーニングプログラム「LPI準拠トレーニング」を、2001年4月より開始する。受講費用は各コースいずれも19万8000円で、初年度は約900名の受講者を見込んでいる。

LPI準拠トレーニングでは、米Caldera Systemsの英文テキストを採用し、日本語で授業を行う。コースは、Linux System Administration（5日間）、Linux Network Administration（5日間）、Enterprise Implementation with Linux（5日間）の3種類。

1コースだけでも受講可能で、3コース終了後には、LPIのレベル1の試験、LPIC1-101とLPIC1-102に合格できるレベルに到達するように想定されているという。しかし合格を保証するものではない。また、LPI試験自体はコースに含まれていない。定員は各コース毎月25名。

レーザーファイブ
(http://www.laser5.co.jp/)
ドットネット
(http://www.kabusikigaisha.net/)
日本語LPI情報サイト
(http://www.yesitis.co.jp/LPI/)

日本IBMと山田洋行、Linuxの Webサーバ高速化ソフトの販売で協業
2001年2月28日

日本IBMと山田洋行は27日、山田洋行が取り扱うLinux版のWeb高速化サーバソフトウェア「BoostWeb OPTIMIZER」と日本IBMのPCサーバ「eServer xSeries（旧Netfinity）」を活用したLinuxシステムの販売で協業すると発表した。主に企業におけるイントラネット、エクストラネットなどに向けたWebサーバパッケージとして販売していく予定だという。

BoostWeb OPTIMIZERは、米Boost Worksの製品で、コンテンツに含まれるデータタイプを分析／圧縮し、最適化されたコンテンツデータを回線上に送り出すことによって高速化を図っているという。

今回の協業にあたり、山田洋行は、日本IBMの本社内に設置されているLinuxサポート・センターのパートナー企業に登録。同センターの施設において動作検証、稼動確認などを行う。日本IBMは、山田洋行に対してLinuxやeServer xSeriesに関する技術情報、技術支援を提供する。

日本IBM（http://www.ibm.co.jp/）
山田洋行（http://www.yamada.co.jp/）

ターボリナックス ジャパンと日本SGIがIA-64やハイエンド分野を中心に包括提携
2001年2月27日

日本SGIと、ターボリナックス ジャパンは、Linux分野での開発、サポート、販売／マーケティングにおいて、ハイエンド分野／IA-64分野を中心とした包括的な提携を結んだ。

今回の提携第1弾の活動として、ターボリナックス ジャパンのハードウェア認定プログラムである「Turbolinux Tested Program」において、「SGI 1450 Server」、「SGI 1200 Server」の2サーバ機を認定した。さらにワークステーション3機種「Silicon Graphics 230／330／550」も認定中。

日本SGIはこれらの製品にTurbolinuxを搭載し、ハイエンド市場に対して積極的に販売していく。さらにターボリナックス ジャパンは日本SGIに対してサポートを行う。今回の包括的な提携以前にも、昨年8月にSGIのコンパイラ「SGI Pro64」をターボリナックスのIA-64対応Linuxに組み込むことで契約を締結している。

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

日刊アスキー Linux Business Report

http://www.linux24.com/

このところ、以前にもまして組み込み機器用Linuxの動きが出ている。この分野も多岐にわたってきており、小型ルータなどのサーバ系アプライアンス製品、リアルタイム系機能を持ったLinux、PDAなどの携帯端末用などが存在している。最近のトピックとしては、レーザーファイブの名刺サイズLinuxマシンや米国向けZaurusのLinux採用、Linux搭載PDA「Agenda VR3」の国内販売受け付け開始などが挙げられる。そして、TransmetaのMidori Linuxもついに登場した。

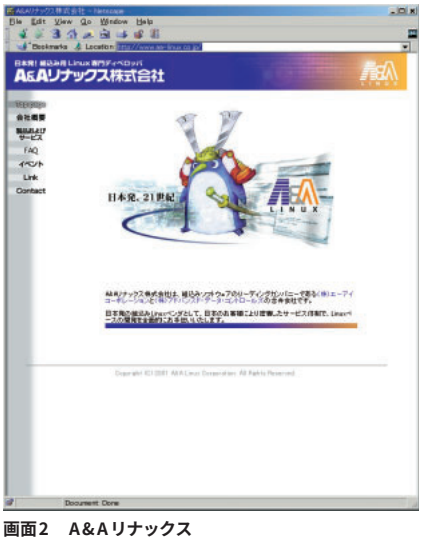
今回のBusiness Reportでは、百花繚乱の組み込み機器用Linux状況を整理し、その現状を探ってみたい。

大手との提携を進めるモンタビスタ

3月に動きがあったのが、モンタビスタソフトウェアジャパン（**画面1**）である。同社の組み込み用Linuxである「Hard Hat Linux」は、Linuxカーネルのチューニングによりリアルタイム性能を向上させたディストリビューションだ。今年に入ってから、IBMのHard Hat Linux用のJava開発ツール「Visual Age Micro Edition」が登場しJava環境の実装に道を開いたほか、ソリューション関連では東芝エンジニアリングと提携してBluetoothをはじめとするプロトコル用ドライバやWebブラウザの開発や、カスタムボードへのインテグレーションを行っていくと発表している。

さらに同社は、NEC、日本電気マイコンテクノロジーと提携してNECのMIPS系CPU「VRシリーズ」への対応も明らかにしている。同社の動きでおもしろいのが、無線に関するトピックだ。3月の発表によれば、同社は米ITT Industoriesとの技術協力により、StrongARMベースの兵士用ハンドヘルド通信装置の開発を行っているという。モンタビスタソフトウェアジャパンによると、この製品は今年後半にリリースされるそうで、「ジャングルや都市部といった、通信の信頼性に問題が生じるような環境において無線通信が機能するように設計される」という。同製品は「兵士用の無線通信を構築するDARPA（Defense Advanced Research Projects Agency）との契約に取り組んでいる」とのことだから、まだ採用されたわけではないようだが、軍事製品への採用ともなれば、Hard Hat Linuxの信頼性をよりアピールすることになるだろう。

画面1　モンタビスタソフトウェアジャパン
(http://www.montavista.co.jp/)



画面2　A&Aリナックス
(http://www.aa-linux.co.jp/)

国内でも組み込みLinux企業が誕生

一方、国内でも組み込みLinux開発の専門会社が昨年11月に設立された。Linuxだけではなく、広く組み込み系の製品を扱うエーアイコーポレーションと、組み込み用ソフトウェア開発ツールのアドバンスド・データ・コントロールズが合弁で設立したA&Aリナックス（**画面2**）だ。A&AリナックスはVine Linuxをベースとした同社のディストリビューション名でもあり、親会社のエーアイコーポレーションが持っているBluetoothプロトコルスタックやUSBホストスタックなどを利用し、組み込み機器用のミドルウェアや開発環境を充実させていく予定である。2001年夏までにはiTRONとインテグレートされ、「リナックス on iTRON」としてリリースされる。

組み込み系市場でシェアを持つ

iTRONと、ネットワークなどこれからの機能を備えたLinuxが融合されることの意義は大きい。家電製品にネットワーク機能搭載が必須になっていく中で注目のディストリビューションである。

小型のLinuxサーバ

小型のLinuxサーバは、ぷらっとホーム「OpenBlockS」（**写真1**）が昨年7月にリリースされたのをはじめ、ワイルドラボの「子羊ルータ LAMB」（**写真2**）、レーザーファイブの名刺サイズLinuxマシン「L-Card＋」（**写真3**）などがある。

アミュレットでは、子羊ルータや、リアルタイムLinuxである米TimeSysの「TimeSys Linux/RT 1.2」、サイバネテックのx86アーキテクチャを採用した小型サーバ「PathNavigator」を販売している。また、サーバ製品ではないが、モトローラのDragonBallと30ピンSIMMをセットにし、米Lineoの組み込みLinux「Embedix uClinux」を採用した開発キット「LINEO uCsimm」の販売も行っている。「LINEO uCsimm」についてアミュレットでは、DragonBall採用ということで、Palmとの親和性も高いとしている。

レーザーファイブのL-Card＋の場合は、MIPS系のVR4181を搭載し、カーネル2.4 test9をカスタマイズしたカーネルを採用している。VR4181はLCDコントローラやオーディオインターフェイスも内蔵しているため、同社ではサーバ分野だけではなくPDAなどへの開発用途も視野に入れて同製品をアピールしている。

一方、ぷらっとホームの「OpenBlockS」は、LinuxPPCで動作する。CPUにはPowerPC 860Tを採用、ストレージにはオプションとしてコンパクトフラッシュを接続できるが、開発者向けには2.5インチのハードディスクを追加することも可能だ。

こうした小型サーバはその低価格と手軽さから人気を博していくだろう。

組み込み用環境も整ってきた

組み込みLinuxの開発環境に関しては、レッドハット（日本シグナスソリューションズ）や前述したHard Hat Linux用Visual Ageのほか、ベンダー各社から提供されているが、それ以外にもGUIやミドルウェアなどが整いつつある。コンシューマー寄りで注目したいのが、アックスの「式神」だ。アックスは、Zaurusで動作するLinux「zxLinux」でも有名だが、昨年11月、レッドハット、テンアートニと共同でGNOME互換のGUIプログラミングAPIやアニメーションフレームワーク、人工知能APIを装備したLinux用GUI環境「式神」事業を推進すると発表した。式神は、手書き文字認識システムである「布目」やPIMを装備するなど、携帯端末としての特徴を持つ。アックスでは、インターネット端末などの用途によっては、式神の各モジュールごとの提供も考えているという。

通信関係では、昨年7月に発表されたIBMのBlueDrekarも挙げておく必要があるだろう。これはLinuxにおけるBluetoothによる通信を実現するためのミドルウェアだ。

Javaに関しては、IBMのVisual Ageのほか、ボーランドがJBuilderの組み込み開発用のバージョン「JBuilder Handheld Express」をリリースしている。ただ、IBMの場合はHard Hat Linux上でのJavaランタイム環境を提供するなど、完全に一ディストリビューション向けの製品であるのに対し、JBuilderのほうはあくまでもJava2 Micro Edition用の開発環境という違いがある。

●

昨年後半の「日本エンベデッド リナックス コンソーシアム」設立など、組み込みLinuxに関するビジネス環境も整ってきている。ローソンのロッピーの例を出すまでもなく、大規模なキオスク端末からPDAに至るまで、その採用事例もこと欠かない。先発のiTORONやVxWorks、コンシューマー向けではPalmやWindows CEなどのライバルがあるが、オープンソースとネットワーク機能を武器に、今後Linuxも浸透していくことだろう。

写真1　ぷらっとホーム「OpenBlockS」
(http://www.plathome.co.jp/)

写真2　ワイルドラボ「子羊ルータ LAMB」
(http://www.wildlab.com)

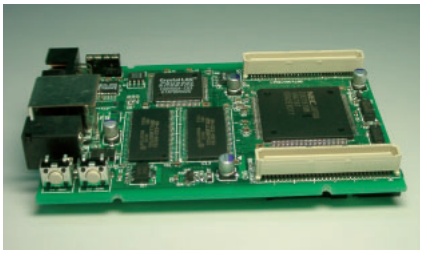


写真3　レーザーファイブ「L-Card＋」
(http://www.laser5.co.jp/)

Distribution

新着ディストリビューション

Turbolinux Server 6.5

グラフィカルインストーラMongoose（マングース）の登場だ。新しいTurbolinuxのインストーラは、使いやすさが向上しているだけでなく、インストール中にReiserFSやExt3のパーティションを作成できるなど、機能面も大幅に刷新されている。Mongoose搭載の一番手は、サーバ用のTurbolinux Server 6.5である。

SuSE Linux 7.1

Linuxのインストールもついに DVD へ突入だ。ドイツで人気ナンバーワンを誇るディストリビューションSuSE Linuxでは、パッケージの多さから、DVD-ROMからのインストールがサポートされている。SuSE Linux 7.1はXFree86 4.0.2など最新の基本システムと、ユーザーがそれらを簡単に設定できるようなGUI設定ツールを収録している。このあたりのバランスの良さは、SuSEの高い技術力をうかがわせてくれる。

Turbolinux Server 6.5

ターボリナックス ジャパンはサーバ用のディストリビューションTurbolinux Server 6.5（以下、Turbo Server）を発表した。Turbo Serverは、カーネルがその上位版にあたるTurbolinux Advanced Server 6と同等にチューニングされ、Turbolinuxとしては初めてのグラフィカルインストーラMongoose（**画面1**）が採用されているのが特徴だ。

Turbo Serverは、Webブラウザから Linuxを設定できるHDE Linux Controller 2.0 Expressや、高機能なバックアップツールNetVault6 Turbolinux OEM版がバンドルされ、4月20日に発売開始予定である。

ユーザーは、インストールからサーバ設定に関するサポートを、電話、FAX、メール、Webページを利用して90日間で5件受けられる。また、ソフトウェアにセキュリティホールが発見された場合は、メールでアナウンスを受け取れる。

高度にチューニングされたカーネル

Turbo Serverの基本システムには、動作実績の豊富なライブラリglibc 2.1.3とカーネル2.2.18が採用されている。

このカーネルは、複数のディスクやパーティションを束ねて、1つの大きなパーティションとして扱えるLVMを始め多くの機能が2.4カーネルからのバックポートによって拡張されているほか、2.4カーネルにもまだ実装されていないジャーナリングファイルシステムExt3が組み込まれている（**表1**）。

2月にリリースされたTurbolinux Advanced Server 6も ReiserFSとExt3に対応していたが、このTurbo Serverではインストール中にExt3などを選択できるのが特徴だ（**画面1**）。

Mongoose登場

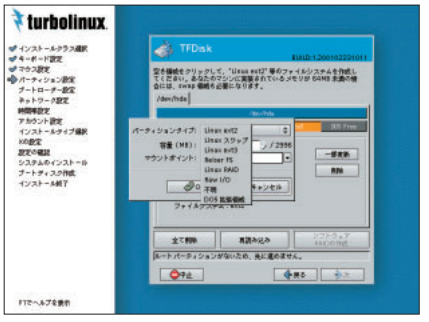
Turbolinuxでは長らくテキストベースのインストーラが採用されていたが、Turbo Serverにはグラフィカルなインストーラ Mongoose（マングース）が新たに採用された。

Mongooseは、Red Hat Linuxのインストーラ Anacondaと、Linux MandrakeのDrakXを折衷したような構成で、これまでユーザーがわかりにくいと感じていた部分が改良されている。

たとえば、パーティションの設定画面では、ハードディスクの領域を棒グラフ状に表示して、直感的にパーティションを作成できるように工夫されているほか、Linuxの起動にまつわるトラブルを減らすために、Linuxカーネルが格納される「/boot」を別パーティションに作成するようユーザーへ促す作りになっている。

新機能	概要
LFS	サイズが2Gバイトを超えるファイルをサポート
LVM	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
Ext3	Linux標準のext2にジャーナリング機能を追加した新しいファイルシステム
ReiserFS	Linuxで最も実績のあるジャーナリングファイルシステム
kparam	いくつかのカーネルパラメータを動的に変化させる
Raw I/O	DBMSから直接ハードディスク上のファイルにアクセスする
Big Memory	最大4Gバイトのメモリを搭載可能にする

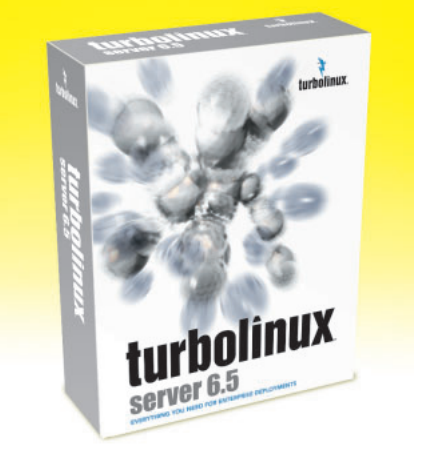
表1　サーバ用にチューニングされたカーネル
動作実績の豊富な2.2系カーネルに、2.4カーネルの先進機能が多数追加されている。



画面1　グラフィカルなインストーラMongoose
新しく採用されたインストーラは、操作性が向上しただけでなく、機能面も拡張されている。パーティション設定画面では、他のディストリビューションに先駆けてExt3ファイルシステムを選択できるようになっている。

また、インストールの進捗状況が画面の右側に表示され、パッケージのインストールを始める前に設定内容全体をチェックできるので、インストール作業全体を見渡せるようになっている。

このほかにも、英語、日本語、韓国語、中国語の表示に対応するなど、2バイト語圏に強いTurbolinuxの特徴が現れている。これから開かれるTurbolinuxの新しい局面を期待できる内容だ。



製品名　Turbolinux Server 6.5
価格　3万9800円
問い合わせ先　ターボリナックス ジャパン株式会社
03-5766-1660
http://www.turbolinux.co.jp/

SuSE Linux 7.1

ドイツでもっとも人気の高いディストリビューションSuSE Linuxの最新版SuSE Linux 7.1（以下、SuSE 7.1）がリリースされた。

SuSE Linuxとは？

SuSE Linuxはカーネルなどの基本システムに、いち早く最新版を組み込むのが特徴のディストリビューションで、SuSE 7.1でもカーネル2.4.0、XFree86 4.0.2、glibc2.2が採用されている。

カーネルは2.2.18と2.4.0を選択可能で、どちらを使用する場合でもジャーナリングファイルシステムReiserFSを利用できる。カーネル2.4.0を選択した場合は、IPv6や最大64Gバイトのメモリなどの機能がサポートされる。

デスクトップ環境に標準でKDE 2.0.1を採用しているSuSE 7.1にはProfessionalとPersonalの2つのパッケージが用意されている。Personalはおもにデスクトップユーザーをターゲットにした構成で、Professionalはこれに LDAPなどのサーバアプリケーションや、KDevelopなどの開発ツールが追加されている。

ユーザーは、電話、FAX、メール、Webページを利用して、Personalは60日間、Professionalは90日間のインストールサポートを受けられる。ProfessionalはCD-ROMが7枚と多いせいか、これらをまとめて収録したDVD-ROMからのインストールも可能だ。

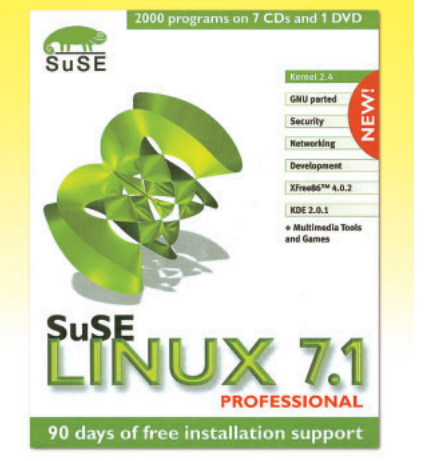
完成度の高い設定ツールYaST2

SuSE Linuxは、最新の基本システムを採用しているほかにも、GUIの統合設定ツールYaST2（Yet another Setup Tool version 2）が付属するのが特徴だ（**画面1**）。

YaST2では、PPPoEを利用したADSLクライアントの設定、ネットワーク経由でのRPMパッケージのアップグレード、サウンドカードやプリンタといったハードウェアの各種設定が可能だ。各設定項目はウィザード形式になっているので、Windowsユーザーも違和感なくシステムを設定できるだろう。

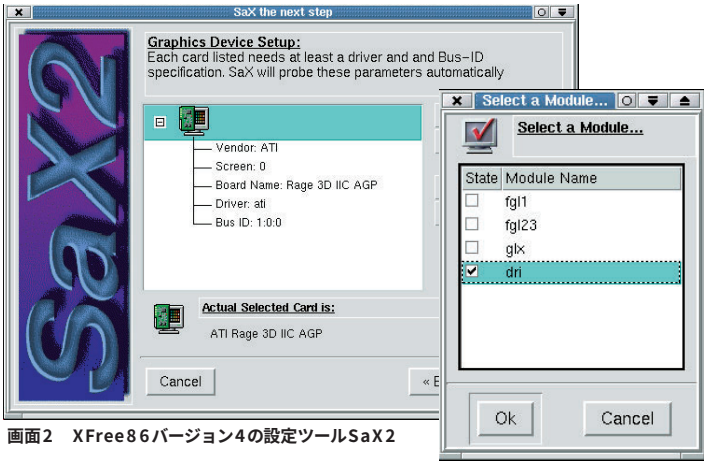
また、YaST2とともに付属するSaX2（SuSE Advanced X Configuration Tool）というツールを使えば、高速に3Dを描画する機能DRIモジュールの利用なども含めて、XFree86 4.0.2をGUIで設定可能だ（**画面2**）。

このように、最新パッケージを採用するだけでなく、ユーザーがそれらを簡単に利用できる設定ツールを収録するあたりは、Linuxカーネル開発者が多数在籍するSuSE Linux Inc.の高い技術力が現れている。日本語版が待ち遠しいディストリビューションである。



製品名	SuSE Linux 7.1	
価格	Professional	69.95USドル
	Personal	29.95USドル
発売元	SuSE Inc.	
	http://www.suse.com/	

画面1　統合設定ツールYaST2



画面2　XFree86バージョン4の設定ツールSaX2

Distribution ▶▶▶

Vine Linux 2.1.5リリース

人気の高い日本語ディストリビューションVine Linuxの最新版Vine Linux 2.1.5が3月24日にリリースされた。

Vine Linux 2.1.5では、バージョン2.1でバグやセキュリティホールの見つかったパッケージがアップデートされ、ProFTPDやPostfixなど、いくつかのサーバ用アプリケーションがバージョンアップされている。

また、VinePlusに収録されていた高機能パッケージ管理ツールaptが正式に採用され、Emacs系のエディタはEmacsに1本化された。Vine Linux 2.1.5は旧バージョンからのアップグレードインストールも可能である。

Project Vine（http://www.vinelinux.org/）

Mandrake 8.0ベータ版公開

フランスのLinuxディストリビューションMandrake Linuxの最新版となる、バージョン8.0のベータ版が公開された。

Mandrakeのベータ版は、カーネル2.4.2、XFree86 4.0.2、RPM 4.0という基本構成で開発中だ。デスクトップ環境には最新のKDE 2.1と、新しいファイルマネージャ「Nautilus」を採用したGNOMEが収録される。

MandrakeSoft SA（http://www.linux-mandrake.com/）

Midori Linuxリリース

省電力CPUのCrusoeや、Linuxの生みの親Linus Torvaldsが在籍していることでお馴染みのTransmetaが、これまでMobile Linuxとして知られていた組み込み用Linuxを、Midori Linuxと名づけてリリースした。

Midori Linuxは、組み込み機器など省電力と低発熱が要求される環境に向けて開発されている。

Midori Linuxの「Midori」は、日本語の「緑」からとったもので、省電力が環境に優しいことと「緑」のイメージとを関連づけて採用されたようだ。Midori LinuxはGPLで配布されおり、（http://midori.transmeta.com/pub/）からダウンロードできる。

Midori Linuxのキャラクター（？）は髪の色が緑色だ。名前もやはりみどりちゃんなのだろうか。

Transmeta Corporation （http://www.transmeta.com/）

Midori Linux（http://midori.transmeta.com/）

Caldera Project42公開

Caldera SystemsはOpenLinux 3.1のベータ版（コードネームProject42）を公開した。Project42は、現在カーネル2.4.1を採用して開発が進められている。

OpenLinux 3.1は、セキュリティを強化したWebサーバ、SambaやNFSを使ったファイル・プリントサーバ、一般的なネットワークサーバ、ユーザーが好みのサーバアプリケーションを動作させるために基本システムだけを収録した合計4つにパッケージ化される予定だ。

Project42は（http://www.calderasystems.com/products/beta/）からダウンロード可能だ。

Caldera Systems, Inc.（http://www.calderasystems.com/）

HOLON Linux 2.0 Serverリリース

初心者向けに特化したディストリビューションHOLON Linuxのサーバ版、HOLON Linux 2.0 Serverが5月11日に発売される。

サーバ版もHOLON Linuxらしく、RealStreaming Server、QuickTime Streaming Server、MP3 Streaming Serverといったマルチメディアサーバを目指したパッケージが収録される。また設定ツールとして、HOLON Linuxが独自にカスタマイズしたWebminが収録される予定だ。

株式会社ホロン（http://www.holonlinux.com/）

Products

国内初のLinuxとWindowsを両方インストールしたノートPC

IBM ThinkPad i Series 1620 Turbolinux＆WindowsデュアルOSキットモデル

日本IBMのB5ファイルサイズのノートPC「ThinkPad i Series 1620」をベースに、TurbolinuxとWindows MeのデュアルOSを搭載している本機は、両OS用の統合オフィスソフトもバンドルされており、すぐに使える本格的デスクトップ用途のLinuxマシンだ。

製品名	IBM ThinkPad i Series 1620 Turbolinux＆WindowsデュアルOSキットモデル
価格	24万4800円（オンラインショップ標準価格）
問い合わせ先	ターボリナックス ジャパン株式会社 TEL 03-5766-1660 http://www.turbolinux.co.jp/

ターボリナックス ジャパンから、Windows MeとTurbolinuxの2つのOSをインストールした「IBM ThinkPad i Series 1620 Turbolinux＆WindowsデュアルOSキットモデル」（以下ThinkPadターボモデル）が発売されている。

B5サイズThinkPad 1620がベース

ハードウェアは「IBM ThinkPad i Series 1620（2661-2CJ）」で、CPUに低電圧モバイルCeleron 500MHzを採用し、64Mバイトメモリと20Gバイトハードディスクを標準搭載している。ディスプレイは、12.1インチTFT液晶で1024×768ピクセル（1677万色）の表示が可能である。

本体には、PCカードスロット（TypeII）、USBポート×2、IEEE1394、ウルトラポート、コンパクトフラッシュ（CF＋TypeII）、モデム、サウンド入出力などのインターフェイスを装備している。

標準でバンドルされるドッキングステーション「Ultra Base X2」に、シリアル、パラレル、PS/2（マウス／キーボード用）といった、いわゆるレガシーインターフェイスと、24倍速CD-ROMドライブおよび3.5インチフロッピードライブ、ステレオスピーカーなどを内蔵している。本体とUltra Base X2をドッキングさせることで、オールインワンノートPCになるわけだ。

両OS用のオフィスソフトをバンドル

ThinkPadターボモデルには、Turbolinux Workstation日本語版6.0 LEとWindows Meでの2つのOSがインストールされている。そして、ブ

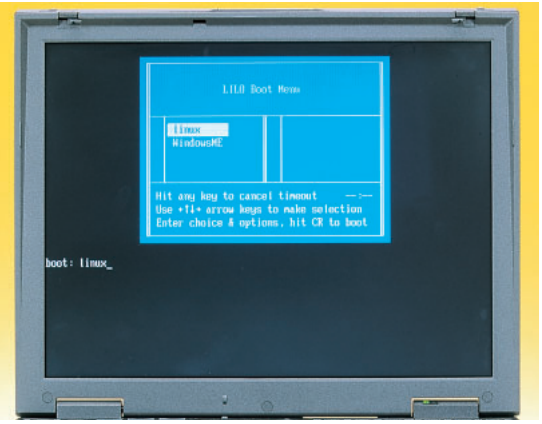


写真1　ブート画面のLILOメニュー表示
購入した時点でLinuxとWindows Meがインストール済み。ブート時に表示されるメニューでどちらにするか選ぶ。

写真2　ウルトラベースをドッキングしたところ
標準添付されるウルトラベースX2には、24倍速CD-ROM、フロッピー、ステレオスピーカーI/Oポートが付いている。

ート時にOSを選択可能なデュアルブート環境になっている（**写真1**）。

Linuxには「Applixware Office for Linux 5.0日本語版」（**画面1**）、Windowsには「Microsoft Office 2000 Personal」がインストールされているので、すぐにワープロ／表計算／プレゼンテーションなどのオフィスアプリケーションを利用できる。そのほかにも多数のインターネット／マルチメディア／ゲーム／実用ソフトがバンドルされている。

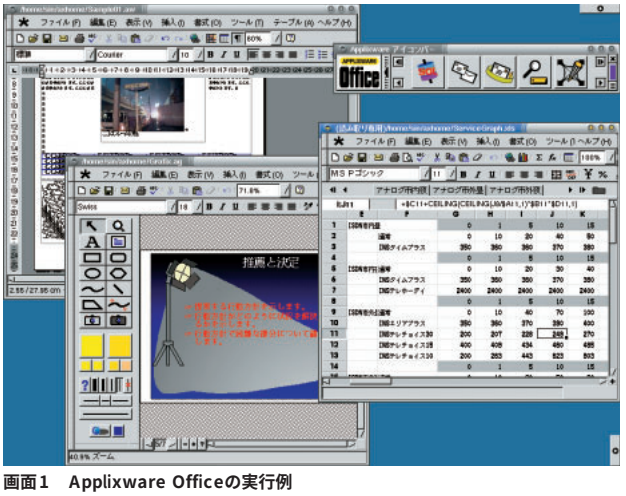
試用したマシンのハードディスクは、3つのパーティションに分けられていて、Windows用に約9.3Gバイト、Linux用に約8.8Gバイト、Linux Swap用に約480Mバイトが割り当てられていた。どちらのOSもシステム関係のファイルは1～2Gバイトであり、ユーザー用のディスクエリアは十分といえよう。

実際にLinuxを起動すると、X Window Systemのグラフィカルログインが表示されるまでの時間も結構速く、Xのレスポンスも悪くない。Applixware Officeのワードプロセッサ（Applix Words）や表計算（Applix Spreadsheets）のボタンを押して起動されるまでも数秒で、Windows上でMicrosoft Officeを使う場合となんら変わらない印象だった。

モバイルCeleron 500MHz、64Mバイトメモリというスペックは、最近ではちょっと物足りないと感じるかもしれないが、内蔵されているハードディスクが高速なのと、グラフィックアクセラレータに使われているATI Rage Mobility Mによって、必要十分な性能が得られていると思われる。ちなみにhdparmコマンドで、ハードディスクのデータ転送速度を計ったところ、16～17Mバイト／秒だった。

ハードウェアの保守サービスとして、日本IBMより1年間標準保証とIBM PC Careサービス（災害・事故もサポート）が提供される。

ThinkPadターボモデルは、面倒なインストールの手間も不要で、持ち運べるLinuxマシンとして最適なノートPCだろう。

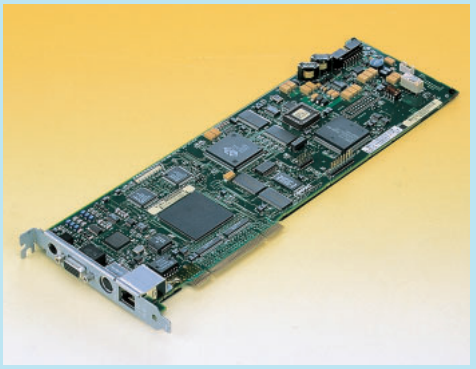


画面1　Applixware Officeの実行例
Applixware Office for Linux 5.0日本語版は、Windows用ソフトに劣らぬ機能を備えたLinux用のオフィススィートだ。

CPU	低電圧版モバイル Celeron 500MHz
RAM	64MバイトSDRAM（PC100、最大320Mバイト）
ハードディスク	20GバイトIDE
ディスプレイ	12.1インチTFT液晶（1024×768ピクセル、1677万色）
グラフィックス	ATI Rage Mobility M（AGP接続、4Mバイト）
内蔵モデム	56Kbps（V.90 対応）／FAX 14.4Kbps
インターフェイス	USB×2、ウルトラポート、ディスプレイ、ライン入力、RJ11(モデム)、マイクロフォン、ヘッドフォン、拡張コネクター、IEEE1394
PCカード	Type II×1（CardBus対応）
コンパクトフラッシュ	CF+Type II×1
ウルトラベースX2	24倍速CD-ROM、3.5インチ1.44Mバイト、シリアル（9ピン）、パラレル、マウス／キーボード共通ポート、マイクロフォン／ステレオスピーカー（Sound Blaster Pro互換）
本体サイズ（mm）、重量	279.4（W）×226.8（D）×24.9～28.7（H）、1.58kg
電源	ACアダプタ（AC100-240V、50/60Hz）、消費電力72W（最大）
バッテリー	リチウムイオン2次電池、使用時間3.5時間

表1　ThinkPad i Series 1620（2661-2CJ）の主な仕様

ハードウェアやOSに依存せずWebブラウザからサーバを監視／管理するPCIカード



リモートInsightボードLights-Out Edition

離れたところにあるLinuxサーバの管理は、telnet（SSH）することが多い。しかし、マシンがハングアップしたり、サーバが落ちていたらどうしようもない。電源のON／OFFやリセット、BIOSの画面からOSの起動まで、Webブラウザで制御できるサーバ内蔵用カードだ。

製品名	リモートInsightボードLights-Out Edition
価格	6万2000円
問い合わせ先	コンパックコンピュータ株式会社 TEL 0120-101589 http://www.compaq.co.jp/

コンパックコンピュータから、ハードウェアやOSの状態に依存せずグラフィカルな画面でサーバを効率的に監視／管理する「Compaq リモートInsightボードLights-Out Edition」（以下リモートInsight）が発売されている。

Lights-Outとは照明を消すという意味で、コンピュータルームの照明を消したままの状態でもサーバが操作できるということを表している。

電源ボタンやリセットも遠隔操作可能

リモートInsightは、Intelのi960 CPUとディスプレイ／LANコントローラを搭載したPCIカードで、コンパックのPCサーバProLiant ML／DLシリーズに装着して利用する。装着されたサーバは、リモートInsightのLANポートに接続した別のマシンのWebブラウザから、仮想的に画面／キーボード／マウスが利用できるほか、電源ボタンやリセットまでコントロールすることができる。ハードウェアで実現しているため、サーバやOSの状態、ソフトウェアに依存することなく、BIOSの画面から、WindowsやX Window Systemのグラフィック画面まで見ることができる。すなわち遠隔地にあるサーバが、あたかも自分の手元にあるかのように監視／管理を行うことが可能になる。

今回試用したマシン本体は、ProLiant DL320（**写真1**）という1Uサイズ（高さ44.5mm）のラックマウントサーバで、PentiumIII800MHz、128Mバイトメモリ、20GバイトIDEハードディスクを搭載している。

リモートInsightをPCIスロットに装着し、16ピンのフラットケーブルでマシン本体にも接続する。このケーブルによって、キーボード／マウスの制御と仮想電源ボタンをLAN経由でコントロールすることが可能になる。そのため、リモートInsightを使うには、このケーブルが接続できるコンパックのProLiant／Prosigniaに限られる。

リモートInsightには、VGAとキーボード／マウス用のPS/2、LANコネクタが付いている（**写真2**）。ディスプレイケーブルは、DL320本体ではなく、こちらに接続する。キーボード／マウスは、付属する専用ケーブルを使って両方に接続する。LANケーブルは、それぞれに接続する。

リモートInsightの初期設定は、専用ユーティリティも用意されているが、サーバ起動時に表示されるBIOSの途中で、F8キーを押してセットアップメニューから行った。LANのIPアドレスやルーティング情報などは、固定IPアドレスを指定してもよいが、今回はDHCPで取得するようにした。また、管理者パスワードを設定しておく。

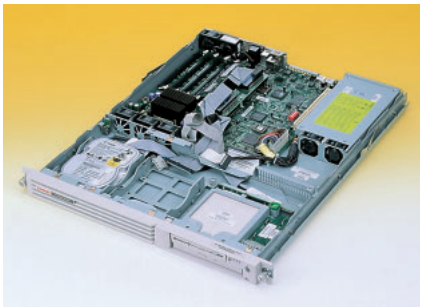
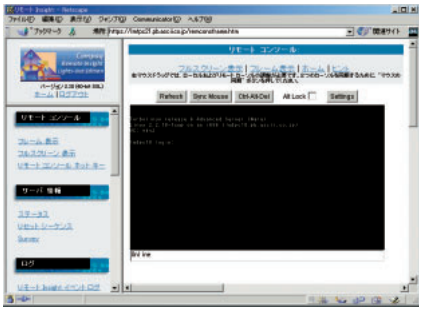


写真1　ProLiant DL320
1UラックマウントサーバProLiant DL320に、リモートInsightボードLights-Out Editionを装着して試用した。

画面1　アカウントログイン
リモートInsightボードにWebブラウザで接続すると、SSLによる暗号化が行われ安全に通信できる。



画面2　リモートコンソール
telnetとは違い、このリモートコンソールの画面は、本体のディスプレイ画面をそのまま表示している。

写真2　リモートInsightボードの外部出力
左から、外部電源入力、VGA出力、PS/2、LANコネクタ。他のPCからこのLAN経由でサーバを監視／管理できる。

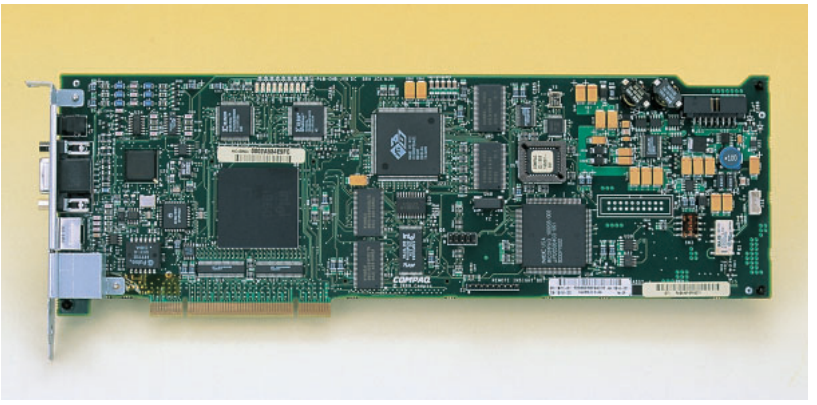


写真3　リモートInsightボードLights-Out Edition
i960 CPU、ATI RAGE IIcグラフィックス、i82559 LANコントローラを搭載している。右上の16ピンコネクタからケーブルで本体に接続し、内蔵仮想電源ボタン、キーボード／マウスをコントロールする。

画面3　リモートコンソールのフルスクリーン表示
Webブラウザから、サーバ機のX Window Systemの表示も見ることができる。マウスも動くし、キー入力も可能。

画面4　仮想電源ボタン
システムがハングアップした場合でも、遠隔地にあるサーバの電源をON／OFFしたり、リセットすることが可能。

画面5　リモートコンソールのホットキーの設定
CtrlとAltとファンクションキーのように複数のキーを同時に入力したい場合には、ホットキーを利用する。

X Window Systemの画面もWebブラウザで表示できる

それでは実際にどのように使うのか見てみよう。Webブラウザで、リモートInsightのIPアドレスまたはDNS名を指定すると、SSLの認証のためセキュリティ警告のダイアログが表示される。それを承認すると、**画面1**のようなアカウントログインの表示になるので、先ほど設定したユーザー名とパスワードを入力する。

左側のメニューの「フレーム表示」ボタンを押すと、**画面2**のように右側の「リモートコンソール」に、サーバの画面と同じ内容が表示される。Webブラウザを起動しているマシンのキーボードからログインし、コマンドを入力／実行することができる。Xが立ち上がっていても同様に利用できる（**画面3**）。ただしXでの画面の色は、正しく表示されないこともあるようだ。

Webブラウザへの表示とキーボード／マウスのコントロール機能は、Javaによって実現しているので、Java 1.1をサポートするIE 4.01または、Netscape Navigator 4.05以上を使用する必要がある。

メニューにある「サーバリセット」と「仮想電源ボタン」（**画面4**）からは、システムがハングアップした場合に、サーバのリセットや、電源のON／OFFが行える。付属するACアダプタからリモートInsightに電源を供給しておけるため、サーバの電源に障害が発生したとしても、リモートInsightに接続することが可能で、サーバ機の状態を知ることができる。リセット時の前後の画面表示テキストをキャプチャしているので、そのシーケンスを再生して見ることによって障害の原因を調べることの手助けにもなる。

サーバマシンのコンソールをネットワーク経由で制御できるということで、セキュリティが心配になると思う。しかし、サーバ自体のネットワークポートをまったく利用しないため、メンテナンス用にtelnetやSSH、FTPなどのポートを開ける必要がない。サーバルームなどでは、リモートInsight用のLANを別系統にして接続すると完璧だろう。また、インターネットから接続する場合でも、SSLを使用した暗号化が行われるので、アカウントとパスワードをしっかりと管理しておけば安心だ。

BASIC

UPGRADE

基本設定から使いこなしまで

Vineで極めるLinux

インターネット、MP3、プリンタなど、入門に最適のVine Linuxをベースにノウハウを伝授!

文:編集部　Text : *Linux magazine*

NETWORK

MUSIC

GRAPHIC

STORAGE

クライアント活用術

CUSTOMIZE

春ですね。
春は始まりの季節です。ここは心機一転、あなたのLinuxライフを見直してみませんか？
この特集では、Vine Linux 2.1をベースにLinuxをさまざまな場面で使いこなす方法を紹介していきます。Vine Linuxは、ベーシックなパッケージ構成と優れた日本語環境を備えた、入門に適したディストリビューションです。Linuxは初めてという方は、Vineのインストールと基本的な操作から試して、さらに一歩進んだ利用法にもチャレンジしてみてください。すでにLinuxを使いこなしている方にとっても、有効な情報があるはずです。ぜひ、新たなLinux活用術を見つけてください。

Linuxの基本操作

～知っておきたい知識とコマンド～

LinuxにはGNOMEやKDEといった統合デスクトップ環境がある。これらを使えばWindowsのようにマウス中心に操作できるが、WindowsマシンからLinuxにリモートログインすれば、必然的にコマンドで操作することになる。

まずはLinuxの基本的なコマンドからマスターしていこう。

ユーザーを管理する

Linuxには、大きく分けてシステム内の特権ユーザー「root」と、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が存在する（**図1**）。

全能のrootにはシステム内でできないことがなにもないので、ちょっとした操作ミスでシステムを壊してしまう可能性がある。rootで作業するときは慎重な操作が必要だ。

■su

作業中のユーザーから、別のユーザーに切り替えるには「su」コマンドを使う。「su」コマンドの書式は、

```
$ su - <ログイン名>
```

で、＜ログイン名＞になにも入力しない場合は、rootに切り替わる。

```
$ su -
Password:（rootのパスワード）
```

rootになると、コマンドプロンプトが「$」から「#」に変わる。「$」が表示されているときは一般ユーザー、「#」が表示されているときはrootで操作しているというわけだ。

元のユーザーに戻る場合は「exit」か「Ctrl＋D」をタイプする。

■adduser

Linuxに新しくユーザーを追加するには、「su －」でrootに切り替わって「adduser」コマンドを使う（**画面1**）。これでユーザーアカウントは作成できたが、パスワードがまだ設定されていないので、「passwd」コマンドを使ってパスワードを設定する。設定ミスを防ぐために、パスワードは二度入力する必要がある。「/etc/passwd」というファイルにユーザーの一覧があるので、「sakura」があるかどうかをチェックしよう（**画面2**）。

ファイルを上手に扱う

Linuxにはファイルのコピー（cp）、移動（mv）、削除（rm）のほかにも、多くのファイル操作コマンドがある。

■find

システム内にあるファイル名を検索するにはfindを以下のようにして使う。

```
$ find <ディレクトリ名> -name <検索ファイル名>
```

Linuxの設定ファイルが含まれている「/etc」ディレクトリから、「conf」

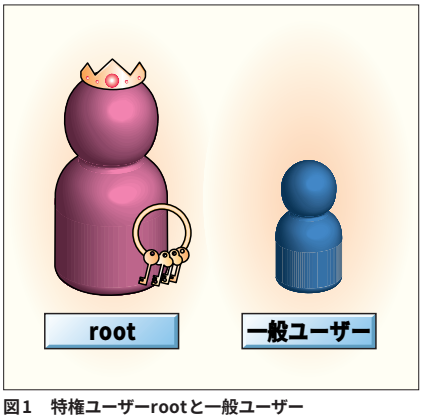


図1　特権ユーザーrootと一般ユーザー

```
# adduser sakura
```
追加するユーザーのログイン名

```
# passwd sakura
Changing password for user sakura
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
```
パスワードを設定・変更するユーザーのログイン名

ユーザー「sakura」のパスワードに同じものを2度入力する

画面1　ユーザーの追加

```
# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
          :
          :
squid:x:23:23::/var/spool/squid:/dev/null
haru:x:500:500::/home/haru:/bin/bash
sakura:x:501:501::/home/sakura:/bin/bash
```
「adduser」コマンドでユーザー「sakura」が追加された

画面2　ユーザーの一覧

という文字列を含むファイル名を検索する（**画面3**）。

```
ファイル検索の対象となるディレクトリ
# find /etc/ -name "*conf*"
                検索する文字列「conf」を「"*」と「*"」ではさむ
/etc/host.conf
/etc/X11/applnk/System/conf.desktop
/etc/X11/wmconfig
/etc/X11/fs/config
/etc/X11/xdm/xdm-config
/etc/X11/wdm/wdm-config
/etc/X11/wdm/wdm-config.in
/etc/X11/wdm/wdmReconfig
        ⋮
```

画面3　findを使ったファイルの検索

■less

「less」はファイルを読むときに使うコマンドで、

```
$ less <ファイル名>
```

として使う。たとえば、

```
$ cd /usr/src/linux
$ less MAINTAINERS
```

とすると、Linuxカーネルの開発者一覧を読むことができる。

このファイルは1画面に表示しきれないので、Spaceキーを押して1画面ずつスクロールダウンして読む。1行ずつスクロールする場合は「↓」か「J」を押す。「Q」をタイプすれば「less」は終了する。

■grep

「grep」はファイルの中の文字列を検索するコマンドで、

```
$ grep  <検索文字列> <ファイル名>
```

として使う。

たとえば、さきほど「less」で読んだMAINTAINERSの中に「Linus」という文字列があるかどうかを確認するには、

```
$ grep -i linus MAINTAINERS
```

とする。「－i」はファイルの中の大文字と小文字を区別しないためのオプションだ。

Linuxでは、複数のコマンドを組み合わせて作業することが多い。このときに重要となるパイプやリダイレクトという便利な仕組みを理解しよう。

■パイプ「｜」

findを使ってファイルを検索すると、画面に表示しきれないほど多くのファイルが表示されることがある。

Linuxには「｜（パイプ）」という機能があり、コマンドを実行して得られる結果に対して、さらにコマンドを作用させられる（**図2**）。

たとえば、**画面3**で得られる出力を、初めから終わりまで画面をスクロールさせながらじっくり見たいとする。こんなときは、findの出力をパイプを通してlessに渡せばよい（**画面4**）。表示結果はやはりSpaceキーなどで読み進めていく。

また、findの出力をさらに絞り込んで検索するときは、パイプとgrepコマンドを組み合わせて使う（**画面4**）。この例では、「/etc」以下にあるファイルで、「conf」と「net」の両方を含む名前のファイルを検索している。

■リダイレクト「>」

コマンドを実行したときの出力をファイルに書き込むにはリダイレクト「>」を使う。たとえば、Linux起動時のメッセージをファイルに書き出したいときは、

```
$ dmesg > bootlog
```

とする。「less bootlog」でLinux起動時のメッセージがbootlogに書き込まれているのがわかるだろう。

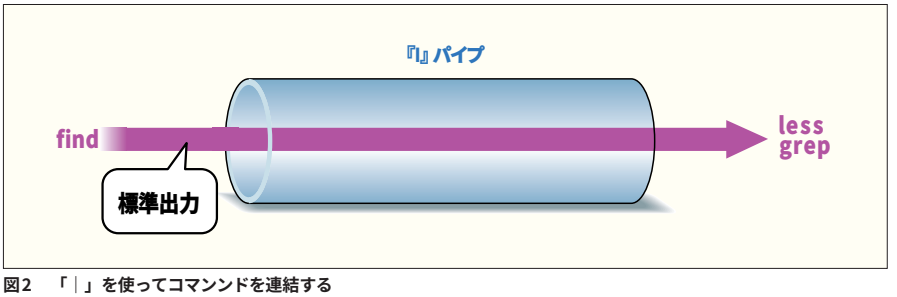


図2　「｜」を使ってコマンドを連結する

```
# find /etc/ -name "*conf*" | less
# find /etc/ -name "*conf*" | grep -i net
/etc/inetd.conf
/etc/inetd.conf.rpmsave
/etc/inetd.conf.rpmnew
```

画面4　パイプ「｜」を使ってコマンドを連結

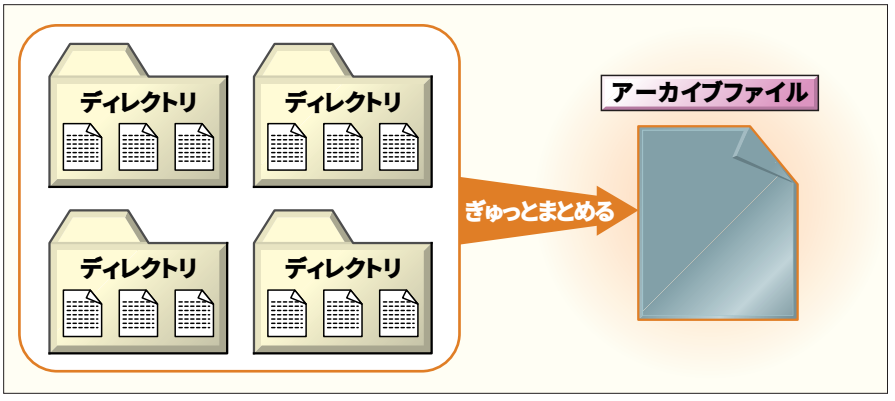


図3　複数のファイルを1つにアーカイブ

さて、同じファイルに対してリダイレクトを2度以上使うと、それまでファイルに書かれていた内容が上書きされてしまう。

これを避けるためには「＞」の代わりに「＞＞」を使うとよい。「＞＞」だと、リダイレクト先のファイルの末尾に内容が追加されていくからだ。

さきほどと同じbootlogを使って次のようにコマンドを実行してみよう。

```
$ cat /proc/pci >> bootlog
```

「less bootlog」でファイルを見ると、Linux起動時のメッセージにLinuxがBIOSから得たハードウェア情報が書き足されていることがわかるだろう。

■man

manはプログラムや設定ファイルのマニュアルを表示するコマンドだ。たとえば、ファイル名を一覧表示する「ls」コマンドのマニュアルは、

```
$ man ls
```

として表示する。manの表示が1ページに収まりきらない場合はSpaceキーを押して読み進めていく。ただ、manのすべてを読むのは大変なので、「／」を使って特定の文字列へ移動すると便利だ。

「ls」コマンドのマニュアルの中から「file」という文字列を検索する場合は、「man ls」—「／」—「file」と順番にタイプする。

こうすれば、検索文字列「file」の箇所が反転表示されるのがわかるだろう。ここで「N」をタイプすれば、次候補が表示される。

Vine Linuxには日本語に翻訳されたマニュアルもあるので、「jman ls」を実行してみよう。翻訳されたマニュアルがあれば日本語で表示されるし、なければ英語で表示される。

複数のプログラムを同時に実行する

Linuxはマルチタスクをサポートしている。マルチタスクとは、同時に複数のプログラムを走らせる機能だ。

■&（アンパサンド）

ユーザーが実行するプログラムには、フォアグランドで動いているものと、バックグラウンドで動いているものの2種類がある。

たとえば、ktermなどのターミナル上で「kterm」とタイプすると、別のktermが1つ起動する。元のktermでは次のコマンドを実行できないのだが、これはあとで起動したktermがフォアグラウンドで実行されているためだ。元のkterm上でCtrl＋Cキーを押すと、あとから起動したktermは終了する。

今度は「kterm」の代わりに「kterm &」と「&」をコマンドのうしろに付けてみよう。さっきと違い元のktermでさらにコマンドを実行できることに気がつくだろう。これは「&」を付けて実行されたプログラムがバックグラウンドで実行されているためだ。

簡単にいうと、「&」を付けて実行するプログラムはバックグラウンドで、「&」を付けずに実行するとフォアグラウンドで動作する。

このように「&」を付ければいくらでもプログラムを実行できるのは、Linuxがマルチタスクをサポートしているからなのだ。

アーカイブファイルを上手に使う

アーカイブファイルとは、複数のファイルを1つにまとめたファイルのことだ。Linuxカーネルのソースは数千個のファイルからなるので、アーカイブファイルにしたうえで、さらに圧縮して配布されている（**図3**）。

■tar（基本）

複数のファイルをまとめるには「tar」コマンドを使う。たとえば、「/usr/src/linux」ディレクトリ以下にある数千のLinuxカーネルのソースファイルを1つにまとめるには、

```
$ cd /usr/src/linux
$ tar cvf /tmp/linux.tar ./
```

とする。tarが終了すると「/tmp」ディレクトリにLinuxカーネルのソースファイルをまとめた「linux.tar」というファイルが作成される。

■gzipとbzip2

次に「gzip」や「bzip2」コマンドを使って「linux.tar」を圧縮する。

```
$ gzip linux.tar
$ bzip2 linux.tar
```

gzipで圧縮した場合は「linux.tar.gz」が、bzip2で圧縮した場合は「linux.tar.bz2」が作成される。

圧縮されたファイルを伸展するには「-d」オプションをつけて実行する。

```
$ gzip -d linux.tar.gz
$ bzip2 -d linux.tar.bz2
```

■tar（応用）

実は「tar」コマンドにはアーカイブと圧縮を同時に行うためのオプションがある。tarとgzipを同時に使ってファイルをまとめる場合は「z」オプションを、tarとbzip2を同時に使う場合は「I」オプションを使う。

「/tmp」ディレクトリにLinuxカーネルのソースファイルをまとめつつ圧縮するには、

```
$ cd /usr/src/linux
$ tar zcvf /tmp/linux.tar.gz ./
$ tar Icvf /tmp/linux.tar.bz2 ./
```

と実行する。

逆にこれらアーカイブファイルの圧縮を解きつつ展開する場合は、

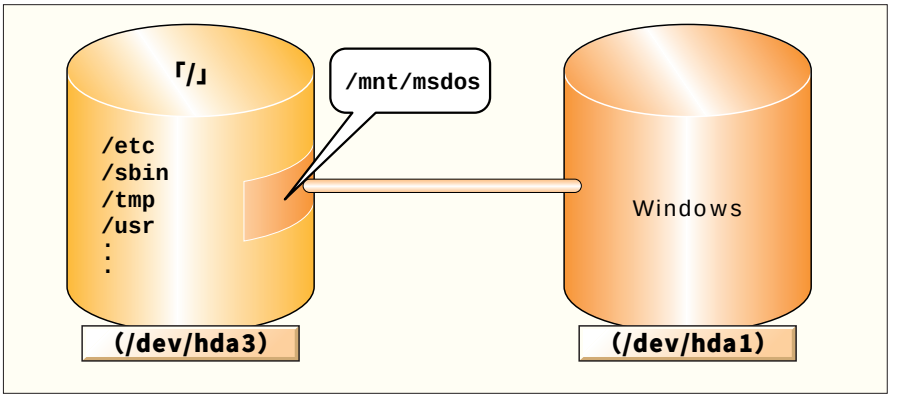


図4　「/mnt/msdos」ディレクトリにWindows領域を連結する

```
$ cd /tmp
$ tar zxvf linux.tar.gz
$ tar Ixvf linux.tar.bz2
```

とする。圧縮されていないファイル（ファイル名が.tarで終わる）を展開するときは、「z」や「I」をつけずに「xvf」オプションをつけてtarを実行すればよい（**表1**）。

Windowsのファイルを操作する

Windowsとのマルチブート環境で、Linuxを使用中にWindows領域のファイルを操作したいことがある。こんなときに使えるのが「mount」コマンドだ（**図4**）。

■mount

「mount」コマンドの基本的な書式は、

```
# mount -t <ファイルシステム> <デバイス名> <マウントポイント>
```

だ。＜ファイルシステム＞には、Windows 9x／Meの領域を使う場合は「msdos」を、Windows NT／2000の領域を使う場合は「ntfs」か「msdos」を指定する。

たとえば、Winodws 9x／Meがインストールされている「C:」ドライブのファイルを扱う場合は、

```
$ su -
# mkdir -p /mnt/msdos
# mount -t msdos /dev/hda1 /mnt/msdos
```

とする。これでWindows領域のファイルが「/mnt/msdos」以下に現れるから、あとはLinuxのファイルと同様に操作すればよい。

「mount」はCD-ROMやフロッピーディスクを読み書きするときにも使う。

操作	操作の対象になるもの	操作方法
ファイルを圧縮する	すべてのファイル	gzip ＜ファイル＞ bzip2 ＜ファイル＞
圧縮されたファイルを解凍する	拡張子が「.gz」と「.bz2」のファイル	gzip -d ＜ファイル.gz＞ bzip2 -d ＜ファイル.bz2＞
ディレクトリ内にある複数のファイルをアーカイブする	ディレクトリまたは複数のファイル	tar cvf ＜ファイル.tar＞ ＜ディレクトリ＞
アーカイブファイルを展開する	拡張子が「.tar」のファイル	tar xvf ＜ファイル.tar＞
複数のファイルを圧縮しつつアーカイブする	ディレクトリまたは複数のファイル	tar zcvf ＜ファイル.tar.gz＞ ＜ディレクトリ＞ tar Icvf ＜ファイル.tar.bz2＞ ＜ディレクトリ＞
圧縮アーカイブされたファイルを解凍する	拡張子が「.tar.gz」と「.tar.bz2」のファイル	tar zxvf ＜ファイル.tar.gz＞ tar Ixvf ＜ファイル.tar.bz2＞

表1　「tar」コマンドの使い方一覧

Vineのアップグレード方法

～GnoPRMとaptを使う～

Windowsではインストーラを使って新しいソフトをインストールする。そして、必要のなくなったソフトウェアを「アプリケーションの追加と削除」を使ってアンインストールする。

このときユーザーはアプリーケションやヘルプファイルの場所を知らなくてもかまわない。

さて、Linuxをインストールしたはよいが、気になるフリーソフトのインストール方法がわからない。Linuxではどうやってソフトをインストールするのだろう？

パッケージとは？ RPMとは？

Vine LinuxではRPM（Red Hat Package Manager）でソフトを管理している。RPMはRed Hat Linuxでお馴染みのRed Hat Inc.が開発したパッケージ管理方式で、Red Hat系ディストリビューションと呼ばれるVine LinuxやKondara MNU/Linuxなど多くのディストリビューションで採用されている。

RPMパッケージとはファイル名が「.rpm」で終わるファイルのことで、その中には、プログラムの実行ファイルやドキュメント、設定ファイルなどが含まれている。RPMを採用するLinuxでは、rpmコマンドやGUIツールを使ってRPMパッケージをインストールしていくのだ。

Vine Linuxには多くのRPMパッケージがインストールされているが、ソフトウェアにはバグがつきものであり、不具合の修正されたRPMパッケージが日々FTPサイトで更新されている。

付録CD-ROMにVine Linux用のアップデートパッケージを収録しているので、これらをインストールしながらRPMパッケージの使い方をマスターしよう。

GnoRPMの起動

GnoRPMは、ほとんどマウスだけでRPMパッケージのインストールやアンインストールを行えるツールだ。

まずVine Linuxに収録されているGnoRPMというGUIツールを、GNOMEメニューから［プログラム］—［システム］—［GnoRPM］と選択して起動しよう（**画面1**）。

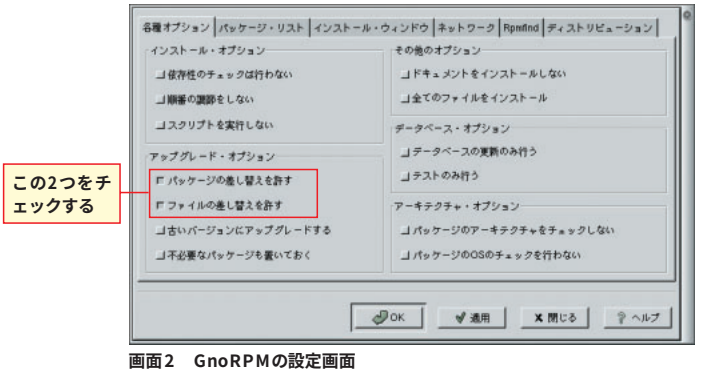
このとき、一般ユーザーでログインしているとrootのパスワードを要求されるので、ダイアログにrootのパスワードを入力する。

GnoRPMの設定

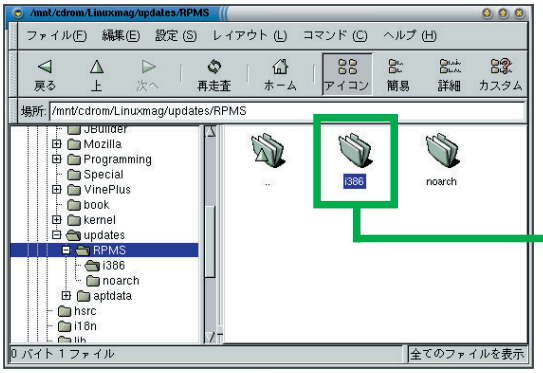
GnoRPMを起動したら［操作］—［設定］として、設定画面を起動する（**画面2**）。デフォルトの状態ではアップデート用のRPMパッケージをインストールできないので、［各種オプション］を選択して、［パッケージの差し替えを許す］と［ファイルの差し替えを許す］をチェックして［OK］を押す。

次に、GnoRPMの［インストール］ボタン（**画面1**）を押して［インストール］を開く（**画面4**）。ここで付録CD-ROM Disc 2をマシンにセットすると、自動的にファイルマネージャが開かれるので、［Linuxmag］—［updates］—［RPMS］と開き、［i386］フォルダを［インストール］へドラッグ&ドロップしよう。

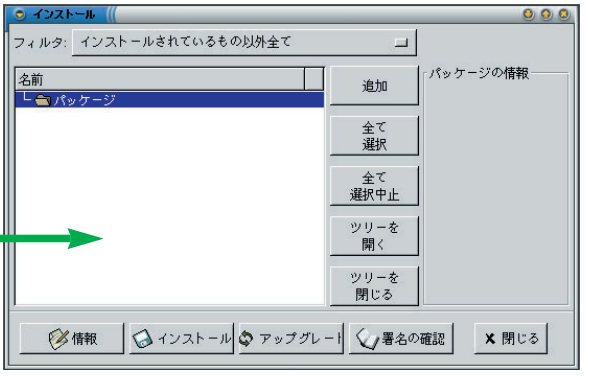
画面1　GnoRPMのメイン画面
GnoRPMのメイン画面から設定画面を開く。



画面2　GnoRPMの設定画面
GnoRPMでパッケージをアップグレードするために、［パッケージの差し替えを許す］と［ファイルの差し替えを許す］をチェックする。



「i386」フォルダをドロップする



画面3　アップデートパッケージが収録されているフォルダ
付録CD-ROM Disc 2の「Linuxmag/updates/RPMS/i386」にアップデート用のパッケージが収録されている。

画面4　GnoRPMの「インストール」画面
インストールするパッケージや、パッケージを含むフォルダをファイルマネージャからドロップする。

アップデート開始

［インストール］で［ツリーを開く］を押すとアップデートパッケージが一覧表示される（**画面5**）。

一覧を見るとパッケージが青と黒で色分けされている。青はインストールされているパッケージよりもバージョンが新しいことを表し、黒はパッケージがインストールされていないことを表している。

また、ドロップした「i386」フォルダには存在しないが、インストール済みのパッケージは緑で表示される。

アップデートの対象になるのは青いパッケージなので、［インストール］画面のフィルタで［新しいパッケージのみ］を選択して一覧表示を更新する。

一覧表示されるパッケージのひとつひとつにチェックを入れてインストールの対象にするのは大変なので、［全て選択］を押して、すべてのパッケージをチェックしよう。

■エラー発生

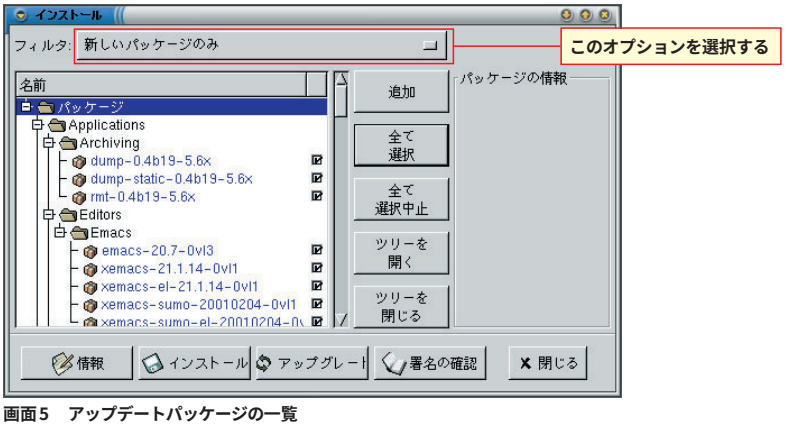
［アップグレード］を押してアップデート用RPMパッケージのインストールが始まるが、環境によってはエラーメッセージが表示されてインストール作業が停止することもある。たとえば、Vine Linuxにすべてのパッケージをインストールした環境では、「xemacs」と「emacs」というパッケージについてのエラーが表示される（**画面6**）。

■パッケージ間の依存関係とは？

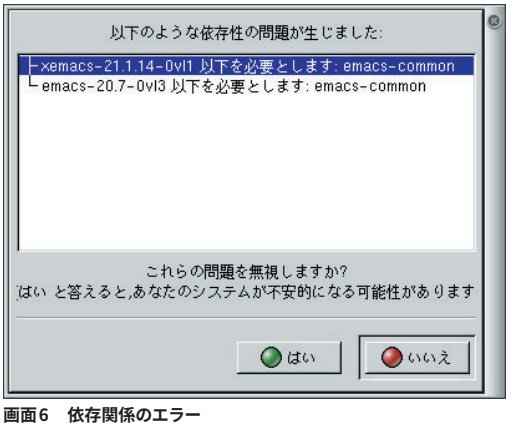
エラーを見ると、これからインストールするパッケージ「xemacs-21.1.14-0vl1」と「emacs-20.7-0vl3」に必要な「emacs-common」がインストールされていないようだ。

RPMは、インストールするのに必要なほかのパッケージの情報や、競合するパッケージの情報を管理している。これらパッケージの依存情報をもとにして、共存できないソフトがインストールされたり、あるプログラムに必要なパッケージがアンインストールされないようにシステムを管理している。

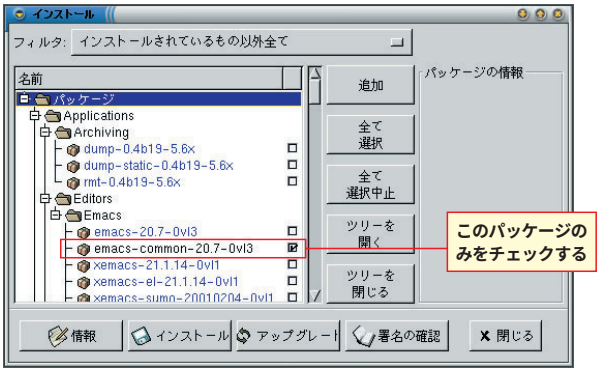
パッケージのインストールやアンインストール中に表示されるエラーメッセージは、パッケージを無理にインストールするとパッケージ間の依存関係を乱すことを意味しているので、すべてのエラーを解決する必要がある。



画面5　アップデートパッケージの一覧
青で表示されるパッケージは、現在インストールされているパッケージよりもバージョンが新しいものだ。



画面6　依存関係のエラー
アップデートインストールする「emacs」と「xemacs」には「emacs-common」が必要だ。



画面7　アップデートパッケージの一覧
黒く表示されるパッケージは、まだインストールされていないものだ。［emacs-common］をチェックして［情報］を押す。

画面8　パッケージの情報
依存関係の原因となった「emacs-common」のパッケージ情報。emacsとxemacsの両方に必要とされるパッケージなので、［インストール］を押してパッケージをインストールする。

■依存関係を解決する

エラー画面では［いいえ］を押していったんアップデート作業を中止する。そして、［インストール］画面に戻り、フィルタを［インストールされているもの以外全て］に変更して、［全て選択中止］を押す。

更新された一覧（**画面7**）にエラーの原因となった「emacs-common-20.7-0vl3」がある。まずはこのパッケージがどんなものなのかを確認しよう。パッケージ［emacs-common-20.7-0vl3］をチェックして［情報］を押すと、パッケージの情報が表示される（**画面8**）。

この画面を見ると、「emacs-common-20.7-0vl3」がまだインストールされていないことや、EmacsとXEmacsという2つのエディタのインストールに必要とされていること、「/usr/bin/b2m」を始めとするファイルがパッケージに含まれていることがわかる。「emacs-common-20.7-0vl3」がどういうパッケージがわかったので、さっそく［インストール］を押す。インストールを終えたら、［閉じる］を押して［インストール］画面（**画面9**）へ戻る。

■再度アップデート開始

エラーの原因となっていた「emacs-common-20.7-0vl3」をインストールしたので、**画面5**から始まる手順をくり返す。［インストール］画面のフィルタで［新しいパッケージのみ］を選択する。ツリーを開いたあとに［全て選択］を押して、アップデートパッケージをチェックする。

さきほどのエラーは解消されているので、［アップグレード］を押してインストールを始める（**画面10**）。

パッケージのインストール中に警告メッセージが表示された場合は、メッセージのメモを取っておこう。

インストールが終わったら、再度CD-ROMの「i386」フォルダを［インストール］画面にドロップしてみる。フィルタで［新しいパッケージのみ］を選択して、青いパッケージが表示されなければ、パッケージはすべてアップデートされている。

さらに進んだツールapt

パッケージをインストールする際に、どうしても依存関係を解消できないときはaptの出番だ。

apt（Advanced Packege Tool）は、Debian GNU/Linuxに実装されている高機能なパッケージ管理ツールで、ハーディスク、CD-ROM、FTPサーバな

画面9　アップデートパッケージの一覧
「emacs-common」をインストールしたら、再度青く表示されるパッケージをチェックして、［アップグレード］を押す。

画面10　パッケージのアップグレード
パッケージをアップグレードインストールする際にいくつかの警告ダイアログが表示されるかもしれない。警告内容をメモしておこう。

どにあるパッケージを検索したり、インストールできるのが特徴だ。インストールに必要なほかのパッケージも一緒にインストールしてくれるので、依存関係のエラーに悩まなくてもすむ。

■aptのインストール

まずはCD-ROMに収録したaptパッケージをインストールしよう。

ファイルマネージャで「Linuxmag」—「VinePlus」—「2.1」—「RPMS」—「i386」と開き、「apt-0.3.19.cnc.27-6vl15.i386.rpm」をGnoRPMの［インストール］にドロップする。次にaptをチェックして［インストール］を押す。

■aptを使ってアップデート

まずはマシンをインターネットに接続してroot権限で、

```
# apt-get update
```

としてaptのデータベースを更新する。次に、

```
# apt-get upgrade
```

とすれば、aptが自動的にネットワーク経由でアップデートパッケージをインストールしてくれる。

もし、aptを実行する前にパッケージの依存関係が壊れているとaptは途中で停止するので、その場合は、

```
# apt-get -f upgrade
```

と「—f」オプションを付けて実行する。

■もっともっとapt

aptの設定ファイル「/etc/apt/sources.list」を編集すれば、さらに進んだ使い方が可能だ。たとえば、VinePlusのパッケージをaptでインストールする場合は、このファイルを**リスト1**のように編集する。そして、

```
# apt-get update
```

と実行すれば、VinePlusのパッケージを利用可能になる。

たとえば、「KDE」という文字列を含むパッケージがあるかどうかを確認したいとする。こんなときは、

```
# apt-cache search --names-only kde
```

としてパッケージを検索する。

探しているパッケージの候補が検索にかかれば、

```
# apt-cache show kdebase
```

でパッケージ内容を確認する（**リスト2**）。この「kdebase」はどうやらデスクトップ環境KDEの核となるパッケージのようだ。もしこのパッケージをインストールするのならば、

```
# apt-get -f install kdebase
```

と実行する。こうすれば、FTPサーバから必要になるパッケージも一緒にダウンロードしてインストールしてくれる。使いこなせればとても便利なツールなので、aptのコマンド一覧（**表1**）を参考にしてぜひマスターしよう。

▼リスト1　aptの設定ファイル「/etc/apt/sources.list」

```
# VinePlus 2.1
# for i386
rpm ftp://ftp.ring.gr.jp/pub/linux/Vine/VinePlus/2.1/ aptdata/i386 0
rpm ftp://ftp.ring.gr.jp/pub/linux/Vine/VinePlus/2.1/ aptdata/noarch 4
#rpm-src ftp://ftp.ring.gr.jp/pub/linux/Vine/VinePlus/2.1/ aptdata/i386 0
```

行頭の「#」を削除する

▼リスト2　aptを使って取得したパッケージ情報

```
パッケージ名:    kdebase
セクション:      User Interface/Desktops
インストールサイズ:      30983
メンテナ:        net_hal <net_hal@cwa.bai.ne.jp>
          :
          :
ファイル名: kdebase-2.1-3vl2.i386.rpm
要約: K Desktop Environment - core files
          :
          :
```

aptの主なコマンド	コマンドの意味
apt-get update	パッケージデータベースを更新する
apt-get upgrade	アップデートパッケージをインストールする
apt-get dist-upgrade	ディストリビューションをバージョンアップする
apt-get install ＜パッケージ名＞	パッケージをインストールする
apt-cache search ＜文字列＞	文字列が含まれているパッケージをデータベースから検索する
apt-cache search --names-only ＜文字列＞	名前にある文字列を含むパッケージを検索する
apt-cache show ＜パッケージ名＞	パッケージ情報を表示する

表1　aptのおもなコマンド一覧
コマンドを実行して依存関係のエラーが表示されたら、「-f」オプションをつけて実行しよう。たとえば、「apt-get install」でエラーになったら「apt-get -f install」とする。

いまやデフォルト!? ネットにつなごう

～インターネット接続、メール、Webブラウザ～

メールのやり取りから、Webサイトでの情報収集、ハードウェアのドライバやフリーソフトのダウンロード、さらにはチャットに掲示板、はたまたネットワークゲームに至るまで、日々の生活の中で、あるいはホビーとしてPCを活用するには、もはやインターネットへの接続はあたりまえのものになっている。

Linuxマシンをクライアントとして使いこなすためにも、やはりネットワーク接続は欠かせない。ここでは、ダイヤルアップPPP接続とLAN接続の設定手順、Linuxで利用できるメールソフト、Webブラウザを紹介する。

インターネットに接続する

クライアントマシンのインターネットへの接続には、モデムまたはターミナルアダプタ（TA）によるダイヤルアップ接続と、ルータを介してインターネットにつながっているLANに接続するという2つの形態がある。後者の場合はLANカード（ネットワークインターフェイスカード、略してNICと言ったりもする）を使って接続する。

つまり、モデム／TAまたはLANカードを装着して、きちんとネットワークを設定してあげなければならない。実は、先月号の本誌特集において、このあたりの解説をかなり詳しくしている。ネットワークの仕組みやサーバ側での設定は、そちらを参照していただくとして、今回はクライアント側での設定手順をひと通り説明していくことにしよう。

ダイヤルアップ接続の設定手順

モデム／TAを接続したら、まずはPC側で認識されているかを確認する。rootユーザーに移行して**画面1**の通りにコマンドを実行してみよう。

外付け、内蔵を問わずモデム／TAは通常、シリアルポートデバイスとして認識される。Linuxでのシリアルポートのデバイス名は、**画面1**のcuコマンドで指定しているように「/dev/ttyS0」のようになる。末尾の数字は、システム上の最初のシリアルポートが「0」、以下は順番に各ポートに割り当てられる。

もし、「Connected」と表示されたあとコマンド入力を受け付けない場合は、指定したポート名が違っている可能性がある。いったん「~ .」と入力してcuコマンドのプロンプトから抜け、/dev/ttyS1以下を試してみてほしい。

画面1のように「OK」と表示されれば、まさにOKだ。接続ポートのデバイスファイルに「/dev/modem」というシンボリックリンクを作成し、インターネット接続時に使うユーザーアカウントを「uucp」グループに登録して、そのアカウントでモデム／TAを使用できるようにしておこう。リンクを作成するコマンドラインは以下のとおり。

```
# ln -sf /dev/ttys0 /dev/modem
```

```
$ su -
<rootのパスワードを入力>
# cu -l ttyS0 -s 38400
Connected.
at
OK
~.
Disconnected.
#
```

画面1　cuコマンドによるポートのテスト

ここに注意！

ソフトウェアモデムの罠

PCIスロットに挿す内蔵モデムやマザーボードにビルドオンされたモデムチップの多くは、本来モデムが持つべき機能の一部をソフトウェア（デバイスドライバ）による処理で代替している。これらのモデムは「ソフトウェアモデム」と呼ばれる。あるいは、そのデバイスドライバの多くがWindows用であることから「Winmodem」と呼ばれることもある。

ほとんどのソフトウェアモデムはLinuxでは動作しない。一部には動作する機種もあるが、そのために特別な作業が必要になる。Linux用にモデムを新規に購入する際には注意してほしい。

基本的に内蔵タイプは避けたほうがよいのだが、シリアルポート接続の外付けタイプの製品は店頭にほとんど見られないのが現状だ。何かないかと思って探していたら、ラトックシステムの「REX-PCI56」という機種のパッケージにペンギンマークと「Linux Ready」の文字が！

残念ながら今回はテストできなかったのだが、調査したところ、同社のWebサイトにTurbolinux、Red Hat Linux、Kondara MNU/Linuxについては動作確認しているとあった。しかし、Vineの場合はナゼだか、別途ドライバのダウンロードが必要とある。

やはり、なんとかしてシリアルポート接続の外付けモデムを探すしかないようだ。なお、PCカードタイプのモデムは意外に動作するものが多いようだ。今回は詳しくお伝えできないが、機会をみていずれ取り上げてみたい。

画面2　Quick DialupのPPP設定画面

[標準設定] タブ	
デバイス名	シリアルポートのデバイス名
ダイアル形式	普通のアナログ回線は「Pulse」、プッシュホン回線は「Tone」
電話番号	プロバイダのアクセスポイントの電話番号
ログイン名	プロバイダから割り当てられたユーザーアカウント名
パスワード	ユーザーアカウントのパスワード
[詳細設定] タブ	
認証プロトコル	CHAPまたはPAPが一般的
PPPモード	通常は「Active」
モデムタイプ	一覧に使用機種があるとき意外は「generic」でよい
アイドル時間	無通信状態が続いた際に切断されるまでの待機時間
IPマスカレード	ネットワーク（LAN）の設定に合わせる
VL圧縮	通常は「yes」
[DNS設定] タブ	
rsolv.confの書き換え	DHCPが有効な場合は「Yes」。それ以外は「Fix」
ドメイン名	特にプロバイダの指定がない限りブランク
サーバ1〜3	DHCPが有効な場合はブランク。それ以外はプロバイダの指定した値

表1　Quick Dialupの主な設定項目

グループへの登録については「man usermod」を参照のこと。

続いてPPP（Point to Point Protocol）の設定に移ろう。PPPはダイヤルアップ接続時にインターネットサービスプロバイダ（ISP）のアクセスポイントとの通信に使われるプロトコルだ。利用するには、専用の通信ソフトウェアと「トンネルデバイス」と呼ばれる特殊なデバイスが必要となる。

「専用」に「特殊」とくれば、何やら大掛かりな作業が必要な感じがするかもしれないが、ほとんどのディストリビューションには、必要なパッケージが標準で用意されている。Vine 2.1では、「PPxP」というプログラムと「tap0」というデバイスがそれにあたる。特にtap0については、あらかじめ必要な設定がされているので、あらためて設定を行わなくて済む。念のため確認しておくと、/etc/conf.modulesに以下の記述があるはずだ。

```
alias tap0 ethertap
```

「ethertap」とあるのは、tap0のデバイスドライバで、モジュール（ethertap.o）は/lib/modules/2.2.17-0vl10/net/に置かれている。ほかのディストリビューションの場合も、ethertap.oが見つかれば（パスの「2.2.17-0vl10」の部分はディストリビューションによって異なる）/etc/conf.modules（またはmodules.conf）に上の記述を追加することで設定できる。

次にPPxP。Vineには、GUIフロントエンドも用意されている。設定にはこれを使おう。GNOMEならメインメニューから［プログラム］－［Vine Linuxメニュー］－［PPPの設定（qdial）］を選ぶ。Window Makerの場合は、メインメニューの［ユーティリティ］－［ネットワーク］に同じメニューアイテムがあるはずだ。

起動すると**画面2**が表示される。Linux以外の環境で、PPPの設定を行ったことがある方には、見覚えのある項目が並んでいる。それぞれの設定値は、プロバイダとの契約時に通知されているはず。**表1**に各項目の説明を載せておくので、プロバイダから送られてきた書類と突き合わせながら設定してみてほしい。

各項目が入力できたら［保存］をクリックし、プロバイダ名などのわかりやすい名前を付けて設定を保存する。

実際の接続には「tkPPxP」というツールを使う。設定ツールがあったメニューに［PPP接続（tkPPxP）］というメニューアイテムがある。選択すると**画面3**のようなGUIツールが起動する。［File］メニューから［Load Configuration］を選び、先ほど保存した設定を指定して［OK］ボタンをクリックする。

接続は、オーディオプレーヤの「Play」ボタンにあたるボタンで行う。クリックすると、接続が開始され［Dial］→［Chat］→［Auth］→［Network］の順に点灯していく。すべて点灯すれば接続は成功だ。［Auto］とあるのは「ダイヤルオンデマンド」と呼ばれる機能のオン／オフを切り替えるスイッチで、オンにしておくとアプリケーションからの接続要求（Webブラウザで外部のURLを指定した場合など）に応じて自動的にダイヤルするようになる。

うまくいかない場合は、設定を見直してみよう。［Operation］メニューの［Quick Dialup］で、tkPPxPから設定ツールを呼び出すことができる。

画面3　tkPPxP
PPxPのGUIフロントエンド。PPP接続の状態を視覚的に捉えることができる。

```
eth0      リンク方法:イーサーネット  ハードウェアアドレス 00:90:99:16:5F:A8
          BROADCAST  MTU:1500  Metric:1
          RXパケット:0 エラー:0 損失:16 オーバラン:0 フレーム:16
          TXパケット:20 エラー:0 損失:0 オーバラン:0 キャリア:0
          衝突(Collisions):0 TXキュー長:100
          割り込み:11 ベースアドレス:0xdc00
```

画面4　ifconfigの実行例

LANカードの設定手順

LAN経由でインターネットに接続するケースには、ISDN＋ダイヤルアップルータ、CATVやxDSLなどの常時接続サービスといったパターンがある。xDSLの場合は特別な設定が必要になるが、ダイヤルアップルータとケーブルモデム（CATV）の場合は、基本的なTCP/IPの設定を行うだけで接続できる。ルータやケーブルモデムなどの設定は完了しているとして、クライアントでのLANカードの設定について説明しよう。

Linuxでは、LANカードに搭載されているネットワークコントロールチップの種類ごとにカーネルモジュール（デバイスドライバ）が用意されている。チップに合ったモジュールがロードされていれば、LANカード自体はシステムで認識されている状態だ。最近のLANカードであれば、スロットに装着してマシンをブートすると、ほとんどの場合は自動的にハードウェアが認識され、適切なモジュールがロードされる。

```
# ifconfig eth0
```

コンソールから上記のコマンドを実行してみよう。**画面4**のようなメッセージが表示されるはずだ。続いてネットワークの設定を行う。

Xを起動して、メインメニューから［プログラム］－［システム］－［Control Panel］とクリックする（GNOMEの場合。Window Makerの場合は［ユーティリティ］－［管理ツール］－［コントロールパネル］）。表示されるメニューパネルから［Network Configuration］とツールチップが表示されるアイコンをクリック。**画面5**のネットワーク設定ツールが起動する。LANカードの設定は［インターフェイス］タブで行う。［eth0］を選択して［編集］ボタンをクリックし、**画面6**で適切なオプションを設定すればよい。

通常は、プロバイダがDHCPサービスを提供しているだろうから、［インターフェイス設定プロトコル］を「DHCP」にしておけばよい。一部のCATVのインターネット接続サービスでは、静的なグローバルIPアドレスをユーザーに割り当てていることがある。

画面5　［インターフェイス］タブ
この画面で停止／活動を手動で切り替えることができる。

その場合、［IP］と［ネットマスク］にプロバイダから提供されたIPアドレスとサブネットマスクを設定する。また、［名前］タブにある［ネームサーバ］、［ルーティング］タブの［デフォルトゲートウェイ］などの項目も、プロバイダから提供される情報をもとに設定する必要がある。

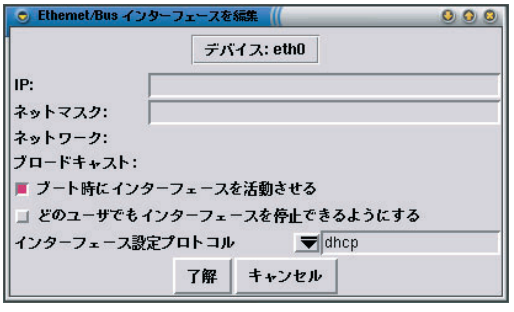
インターネットへの接続が完了したら、友達にメールを送ったり、ネットサーフィン（死語？……ですね）を満喫したり、とLinuxでのネットライフの始まりだ。以下では、Linuxで使えるメールソフトとWebブラウザを紹介していこう。

GUIメールソフト「Sylpheed」

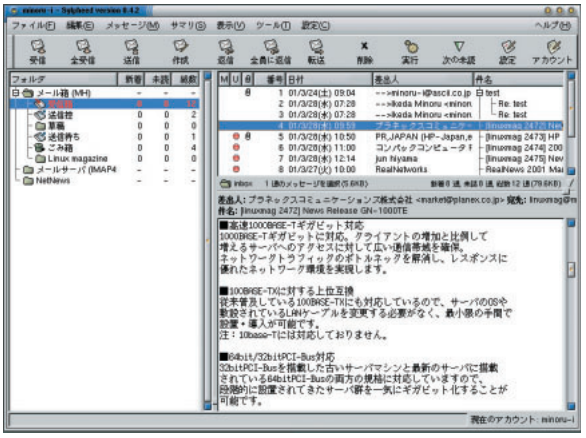
Vine 2.1では、優れたGUIインターフェイスを備えた「Sylpheed」が標準

画面8　WindowMakerのメニューパネル
一番下にあるのがSylpheedのメニューボタン。左端はNetscape Communicator。

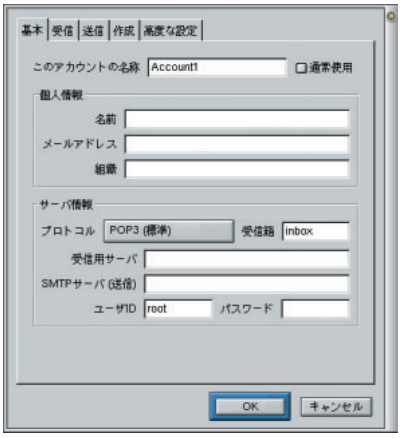
画面7　GNOMEのメニューパネル
右端がSylpheedのメニューボタン。左端にあるのがNetscape Communicator。



画面6　デバイス「eth0」の設定
DHCPが有効な場合は画面のような設定になる。



画面9　Sylpheedのメインウィンドウ
この画面はデフォルトの表示形式。フォルダツリーとメッセージ内容の「ビュー」を切り離して別表示することも可能。



画面10　アカウントの作成
特に難しい項目はないので、プロバイダからの情報をもとに正確に設定しよう。

メールソフトとなっている（ちなみに「しるふぃーど」と発音します）。標準というだけあって、GNOME、Window Makerのどちらの場合も、メニューパネルにアイコンがデフォルトで登録されている（**画面7**、**画面8**）。

Sylpheedのメインウィンドウは**画面9**からもわかるように、Windowsのメールソフトに近い外観を持っている。Windowsのメールソフトに慣れたユーザーであれば、すぐにでも使い始められるはずだ。

機能面も以下のように、とても充実している。

●複数アカウントの管理
●メールの振り分け機能
●アドレス帳
●外部エディタによるメールの作成
●組み込みの画像ビューア
●メール送信時の自動改行機能
●IMAP4、APOPに対応

では、Sylpheedの基本的な設定と使い方を見ていこう。

■新規アカウントの作成

アカウントの登録は［設定］メニューの［新規アカウントの作成］で行う（**画面10**）。「基本」タブでは、メールの送受信に必要な、メールアドレス、プロトコル、メールサーバ、メールサーバでのユーザーアカウントとパスワードなどを設定する。このタブでの設定は、プロバイダから割り当てられる、ごく一般的なものなので、特に説明の必要はないだろう。

［基本］タブの一番上にある［このアカウントの名称］は、Sylpheedでの複数アカウントの管理に使われるものだ。複数のメールアカウントを持っている場合は、プロバイダ名やメールサービス名を設定しておくとわかりやすいかもしれない。その横にあるチェックボックスは、作成するアカウントを起動時の初期アカウントにするかどうかの設定だ。

このほかのタブにある項目も、特にわかりずらいものはない。自分の環境に合わせて、適宜設定してほしい。

■使用環境のカスタマイズ

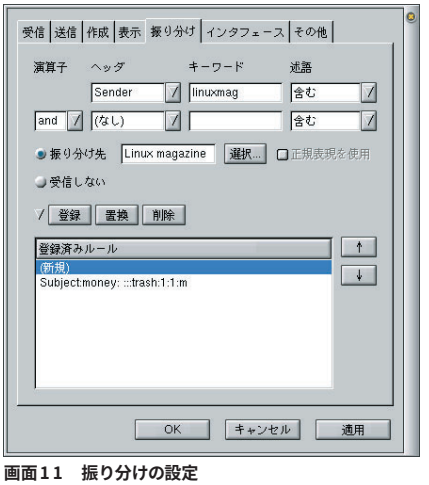
［設定］メニューの［全般の設定］では、送受信時のオプションや、返信を作成するときに元のメールを引用するかどうか、あるいは、画面表示フォント、サマリーに表示する項目（差出し人や日付など）、外部エディタとして使用するプログラム、といった使用時の細かな環境設定を行える。

メールの振り分けも、このメニューから設定できる。**画面11**の［振り分け］タブがその設定ダイアログだ。

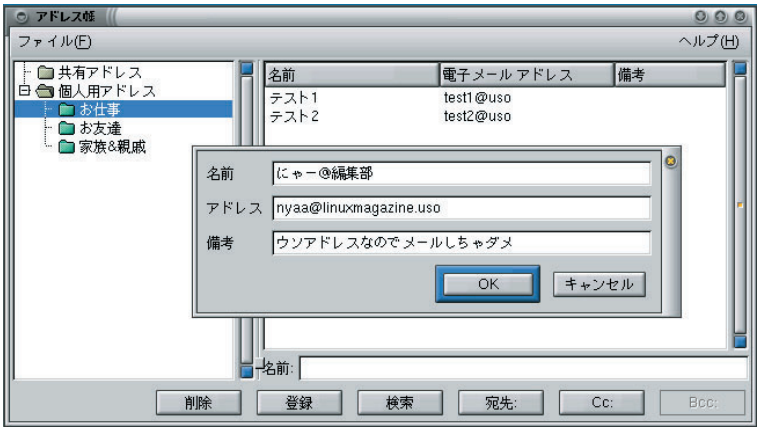
たとえば、件名に「Money」という文字列を含む、たいていの場合はあやしい内容であろうメールをごみ箱に振り分けるには、［ヘッダ］のドロップダウンリストから「Subject」を選んで［キーワード］に「Money」と入力して、［振り分け先］の［選択］ボタンをクリックして［trash］を振り分け先に指定すればよい。

■メールの送信

送信メールを作成するには、ツールバーの［作成］ボタンをクリック。表示される内蔵のエディタで作成してもいいし、［エディタ］ボタンをクリックして使い慣れた外部エディタを使ってもいい。紹介するのが遅れたが、



画面11　振り分けの設定
上部にある項目を組み合わせて振り分けの条件を設定し、［登録］ボタンで下のリストに追加する。

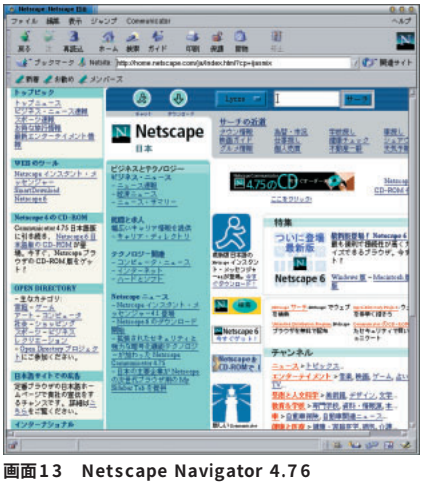


画面12　アドレス帳へのエントリの追加　フォルダを作成してグループごとにメールアドレスを管理できる。

Sylpheedは日本生まれのフリーソフトだ（作者は山本博之さん）。当然のことながら。日本語の入力や表示は何の問題もなく行える。

ここで、アドレス帳も紹介しておこう。メール編集ウィンドウのツールバーにある［アドレス］ボタンをクリックすると、アドレス帳が起動する。アドレス帳の操作インターフェイスも直感的でわかりやすいものだ。また**画面12**に示すとおり、アドレスをグループ分けして管理する機能もある。

作成したメールをすぐに送る場合は、［送信］ボタンをクリック。作成したメールを「送信待ち」にしておき、あとからまとめておくりたい場合は［後で送信］ボタンをクリックすればいい。



画面13　Netscape Navigator 4.76　Vineでは、表示、フォームなどへの入力といった日本語処理が正しく行えるように、あらかじめ設定されている。

■そのほかの機能や操作

最後に、知っていると便利なSylpheedの機能や使い方をいくつか紹介しておく。

●メールをまとめて受信する

複数のアカウントを使い分けている場合にも、ツールバーの［全受信］を使って各アカウント宛のメールを一度にすべて受信できる。

●クリッカブルURL

メールメッセージ中のURLをダブルクリックするとWebブラウザが起動し、URLの内容が表示される。起動するブラウザは［全般の設定］の［その他］タブで変更することもできる（デフォルトはNetscape Navigator）。

●表示方法を変更する

人によってはデフォルトの3ペイン表示（フォルダ、メールのサマリ、メールメッセージ）が、好みでないこともあるだろう（実は私がそう）。［表示］メニューの［フォルダービューを分離］と［メッセージビューを分離］を使えば、別々のウィンドウに表示を分けることが可能だ。

定番Webブラウザ

Windowsの世界では、Internet Exprolerに覇権を奪われた形の「Netscape Navigator」だが、Linuxの世界では、いまだ健在である。Linuxの標準Webブラウザといっていいだろう。

VineでもNetscape Navigator 4.76が標準でインストールされる（正確にはNetscape Communicator。CmmunicatorのWebブラウザコンポーネントがNavigatorで、そのほかにメールコンポーネントのMessengerなどが含まれる）。また、Sylpheedと同様にGNOMEとWindow Makerの双方でメニューパネルにデフォルト登録済みである。ボタンアイコンをクリックするだけで起動できる。

本誌の読者層を考えると、「Linuxは、まだまだこれから勉強です」という方の中にも、「WindowsでMozilla使ってます」とか「以前はNetscapeユーザーでした」という「Mozilla派」が多いかもしれない（Mozillaについては、このページのコラムを参照）。Linux版のNavigatorは、フォントサイズの動的な変更ができない点や、デフォルトでインストールされるプラグインなどの違いを除けば、同バージョンのWindows版とほぼ同じ機能と操作性を備えている。

また、Webブラウザなので操作方法が難解ということもない。とはいえ、Navigatorは初めてという方もやはり多いと思うので、基本的な設定方法について簡単に説明しておこう。

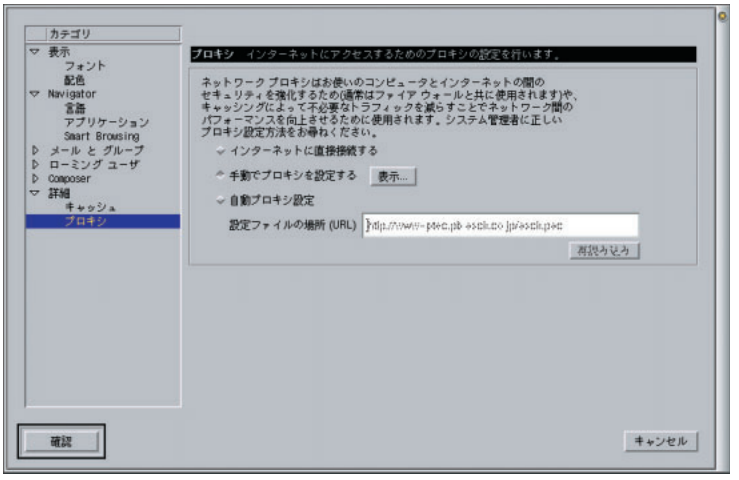
■プロキシサーバの設定

Navigatorの基本的な設定ダイアログを呼び出すメニューアイテムは、伝統的に［編集］メニューに置かれている。一般的なメニュー構成からすると、ちょっと変わった位置にあるのだが、「古くからのしきたり」のようなものだと思ってもらいたい。

設定ダイアログの左側のツリーを［詳細］－［プロキシ］と、たどっていくと右側に設定パネルが現われる（**画面14**）。通常のダイヤルアップ接続の場合は、一番上の［インターネットに直接接続する］をチェックしておけばよい。

プロバイダがプロキシサービスを提供している場合や、自宅やオフィスのLANにプロキシサーバが設置されている場合は、まん中の［手動でプロキシを設定する］をチェックしてから［表示］ボタンをクリックする。次に表示されるダイアログにプロキシサーバのURLを設定しよう。

自動プロキシ設定が有効な環境では、［自動プロキシ設定］をチェックしてURLを指定する。



画面14　Navigatorのプロキシ設定ダイアログ
フォントの変更、JavaScriptのオン／オフ、キャッシュのクリアなど、設定はすべてこのダイアログから行う。

■プラグインの追加

プラグインの設定はWindows版と大きく異なる点だ。ここでは一例として、最近のWebサイトで多く見られる「Flash」用のプラグインの設定方法を紹介しておこう。

開発元であるMacromediaのダウンロードサイト（http://www.macromedia.com/jp/downloads/）からLinux版Netscape用のFlash Playerをダウンロードする。tar＋gzip形式になっているので、以下のコマンドでユーザーのホームディレクトリなどに展開。

```
$ tar -xzvf flash_linux.tar.gz
```

展開先のディレクトリに移動し、suコマンドでスーパーユーザーに移ってから「ShockwaveFlash.class」と「libflashplayer.so」の2つのファイルを/usr/local/netscape/plugins/にコピーする。これでNavigatorを起動すれば、プラグインとしてFlash Playerが自動的に認識されるはず。

テストはMacromediaのWebサイトにある「ギャラリー」に行ってみるとよい。Flash5の機能をフル活用した作品がきちんとブラウズできるか試してみよう。

ほかのプラグインについては、基本的な作業内容は共通しているもののプラグインごとに独自の設定が必要な場合があるので、付属のドキュメントを必ず確認するようにしてほしい。

ちょこっとコラム

MozillaとNetscape 6

「Mozilla」というWebブラウザがある。もともとは、1998年にNetscape（社。開発元の企業のほう）がNetscape Communicator 5.0（NC5）のソースコードを公開し、オープンソース方式での開発を呼びかけたことに始まる。このとき発足したソースコードを管理する組織が「mozilla.org」で、以後リリースされるブラウザにMozillaという名前が冠された（説明は省くが、MozillaというのはNetscapeにとって歴史的な意味を持つ、ある意味で栄光の呼称なのである）。

その後、紆余曲折を経てベースであるNC5とは、まったく別の新しいWebブラウザが開発されることになり、やはりMozillaの名前で呼ばれた。これが現在のMozillaである。

■Netscape 6

2000年11月にNetscapeからリリースされた「Netscape 6」は、Mozilla 0.6とほぼ同じバージョンのソースコードをベースとしたもので、同社によって独自の変更とチューニングが施されたものである。

Communicatorの文字がないとはいえ、時を経てMozillaは再び、もとの「Netscape」に姿を変えたわけだ。別の言い方とすると、MozillaにとってのNetscape 6は、自身がベースになったという意味で「息子」であり、NC5の後継であることを考えると「義理の弟」のような存在ということになる。

実際、両者の外観はほぼ同じで、違いは左肩のロゴ部分くらいだ。

■インストール方法と注意点

今月号の付録CD-ROMに、Mozilla 0.8とNetscape 6のバイナリパッケージを収録している（Disc 2の/Linuxmag/Mozilla/）。どちらも、インストーラが付いているのでセットアップは簡単だ。tarコマンドでアーカイブを展開してインストーラを起動し、あとは指示に従っていけばよい。詳しくは、それぞれのREADMEファイルを参照してほしい。

気をつけたいのは、インストーラはXで動作するので必ずXのターミナルエミュレータから実行することと、MozillaとNetscape 6は共存させないほうがよいということ。Netscape Navigatorとの共存は、少なくとも編集部で確認した限り、どちらも問題なかった。

MP3で音楽三昧

～リッピングとMP3エンコーディング～

MP3はデジタルオーディオデータのフォーマットとして広く普及している。専用のプレーヤも小型で携帯に適していることから、CDやMDの携帯型プレーヤを凌ぐ勢いだ。

ネット上での音楽配信やPCでの音楽鑑賞には（Napsterに代表されるような著作権関連の諸問題はあるものの）もはや欠かせない存在になっている。ここはひとつ、LinuxでもMP3を楽しむしかない！

リッパーとエンコーダのインストール

MP3と著作権、特許に関する話題はコラムに譲っておいて、オーディオCDからデータを切り出してMP3形式にエンコードする手順から説明していくことにしよう。

CDからのデータの切り出しは「吸い出す」、「リップする」、あるいは婉曲に「（ハードディスクに）録音する」と表現されることもある。ここでは、「リップする」または「リッピング」と記述する。リッピング用のソフトウェアは「リッパー」と呼ぶ。

で、そのリッパー。Linuxで使えるリッパーには、代表的なものとして「cdda2wav」と「cdparanoia」がある。今回は、このあと**58ページ**から紹介するCDライティングソフト「CD Recorder」のパッケージの一部であるcdda2wavを取り上げる。

付録CD-ROMにもVine Plusに収録されているCDRecord 1.9のRPMパッケージを収めてあるので、それを使ってまずはインストールしてみよう。CD-ROMをマウントしてRPMパッケージが置かれているディレクトリに移動し、以下のコマンドを実行する。

```
# rpm -Uvh cdrecord-1.9-0vl3.i386.rpm
```

MP3エンコーダも、やはりVine Plusに収録されている「午後のこーだ」を使おう。インストール手順はCDRecordと同様で、ファイル名が異なるだけだ。コマンドは次のとおり。

```
# rpm -Uvh gogo-2.39-0vl2.i386.rpm
```

最後にもう1つ、GUIインターフェイスでリッパーとエンコーダを操作できるプログラム「Grip」をインストールしておこう。これまた手順は同様で、コマンドは次のとおりだ。

```
# rpm -Uvh grip-0.95-0vl1.i386.rpm
```

まずはコマンドラインで確認

インストール作業が終わったところで、それぞれのツールが正しく動作するかコマンドラインで確認してみよう。あわせて各ツールの機能や役割も理解できるはずだ。

cdda2wavは、ATAIP接続とSCSI接続されたCD-ROMドライブをサポートしている。もちろんドライブ自体がオーディオCDの規格であるCD-DAメディアに対応している必要があるが、CD-DAをサポートしていないドライブを探すほうが難しいので、この点は気にしなくてかまわない。

デフォルトではrootユーザー以外はCDRecordのコマンドを実行できない。一般ユーザーで使用するには、CDRecordのインストール時に作成される「cdwrite」グループにユーザーを追加する必要がある。グループへのユーザーの追加については、**34ページ**で紹介しているので参考にしてほしい。

準備が整ったら、手近にあるCDをドライブにセットして次のコマンドを実行してみよう。

```
$ cdda2wav -D /dev/hdc -H -J
```

「－D /dev/hdc」はCD-ROMドライブの指定で、この部分は環境によって変わってくる。たとえば、プライマリIDEポートのスレーブとしてCD-ROMドライブが接続されていれば「/dev/hdb」になる。SCSI接続の場合は少し複雑で、「SCSIバス，ターゲットID，lun」の形式で指定する（詳しくはmanページを参照）。いずれにしろ、TOC（Table Of Contents）の内容が読み込まれて画面にトラック数や曲の長さが表示されればOKだ。

午後のこーだの実行コマンドは「gogo」だ。パラメータなしで実行してヘルプ情報が表示されていれば正しくインストールされている。

では、実際にリッピングとMP3エンコーディングができるかどうか試してみよう。CDの1曲目だけをリップするには、次のように指定する。

```
$ cdda2wav -D /dev/hdc -H -t 1+1
```

前のコマンドラインにも登場した「−H」は、トラックごとの情報ファイル（拡張子は.inf）を作成しないためのオプション。「−t」には「開始トラック＋終了トラック」という形式でリップするトラックを指定できる。終了トラックの指定を省略すると最終トラックまでのすべてのトラックが処理される。−tオプション自体を省略した場合にはCDの全トラックがリッピングされる。また、デフォルトでは「audio.wav」ファイルにリッピング結果が出力されるが、コマンドラインの末尾にファイル名を付けて、出力ファイル名を指定することも可能だ。

続いてエンコード。コマンドラインは次のようになる。

```
$ gogo audio.wav track01.mp3
```

出力ファイル名は任意なので、曲名などをもとにわかりやすい名前を付けられる。上記のようにオプションなしの場合、ビットレート128kbps、ジョイントステレオモード、心理音響解析（MP3のエンコードモードのひとつ。音質に大きく影響する）が有効な状態でエンコードされる。

やっぱりGUIがラクだわ

コマンドラインで実行する場合でも、cdda2wavではオプションを指定すれば、一度にCDの全トラックをリッピングできるが、MP3エンコーディングは、ファイル名などを考えつつ1曲ずつ行わないと、どのファイルがどの曲だったかわからなくなったりで大変である。

Gripを使えば、簡単な操作でリッピング＆エンコーディングできる。しかも、CDDBサーバでアーティスト名、アルバムタイトル、曲名を検索し、それを利用してMP3ファイルをアルバムごとのディレクトリに自動的に振り分けて保存することも可能なのだ。

Gripの実行コマンドは、そのまんま「grip」である。GUIツールなので、もちろんXのコンソールエミュレータから起動する。無事起動したら、リッパーとMP3エンコーダを設定しよう。

初期画面の［Config］タブをクリックして、さらに次の画面で［Rip］タブをクリックすると**画面1**が表示される。［Ripper］はデフォルトでは［Grip（cdparanoia)］となっているので、これを［cdda2wav］に変更する。そのほかのオプションは、そのままで特に問題はないはずだ。

［MP3］タブでも同様に、デフォルトの［lame］を［gogo］に変更すればOK。やはりオプションはそのままに

ちょこっとコラム

MP3と著作権と特許

本文でも少し触れたNapsterの登場以前から、MP3については、著作権問題に関して激しい議論と法廷闘争が行われてきた。高圧縮率・高音質、しかもデジタルデータであるためアナログメディアのように複製のための特別な機器も必要なく、複製による音質の劣化も発生しない。このため、無断複製によって著作権が大きく損なわれる可能性があると危惧されたのだ。

当初はMP3の規格に著作権保護機能がないことや、MP3ファイルを作成したり専用のプレーヤに記録する行為自体すら問題視するという傾向もあったが、MP3が急速に普及した段階で、このような議論は意味を失った。その後、MP3をはじめとするデジタルデータを（特にネットワーク上で）利用するうえでどのように著作権を保護していけばよいのか、また、ベンダー側とユーザー側双方の利益を保護するためにはどうすべきか、といった議論がなされ、記録メディアへの著作権保護機能の搭載、各国での現状に即した著作権関連法の整備、国際的なルール作りなどが進められてきた。現在も議論は続いているが、一部では成果もあがっている。

常識的には、自分が購入したCDからデータをリッピングしてMP3にエンコードする行為は、カセットテープやMDに録音するのと同様に、作成したMP3ファイルを個人で楽しむ範囲内であれば、著作権を侵害しないと考えていいはずだ。ただし、法的に微妙な問題を含んでいるということは常に気にかけておこう。

MP3には、もうひとつ権利がらみの大きな問題点がある。それは「特許」に関するものだ。

■MP3の特許問題

MP3は「MPEG 1 Layer-3 Audio」の略で、この標準規格はISOによって策定・管理されている。問題となるのは、MP3の規格に含まれているエンコードやデータ圧縮に関する技術的な特許である。これらの特許はISOではなく、技術を開発した団体または個人が取得したものだ。MP3規格に則ってデータをエンコードするには、これらの技術をどうしても利用しなければならない。

MP3エンコーダを作成し、配布する場合、関連する特許技術を利用しているとみなされる。つまり配布が有償であろうと無償であろうと、特許の保有者が特許権を行使すれば、使用料を支払う義務が生じる。実際に、いくつかの企業はMP3に関連する特許権を行使している。

商用ソフトウェアであれば開発・販売元の企業が使用料を支払うだろうし、納得できなければ法的手段をとることも可能だ。だが、基本的にボランティアをベースとしているフリーソフトウェアの場合はそうはいかない。このため、バイナリの配布を中止したエンコーダもある（一般的な解釈ではソースコードには特許権は認められていない）。

本文中で紹介している「午後のこーだ」も、、実際に特許問題が原因かどうかの正確な経緯はわからないが、現在は公式にはバイナリは配布されておらず、開発も終了している（付録CD-ROMに収録したRPMは、バイナリの配布がストップする前にVine Plusに加えられたもの）。

●

著作権と特許権というのは、MP3に限らずフリーソフトウェアに常につきまとう難問だ。国ごとに法制度が異なり、法解釈も立場によって異なる複雑な問題ではあるが、フリーソフトウェアの開発プロジェクトに影響が及ぶような事態はぜひ避けてほしいものだ。

しておいてかまわない。

［CDDB］タブではCDDBサーバの設定を行う。これも「freedb.org」で特に問題はないだろう。デフォルトでは、CDがドライブにセットされると自動的にCDDBサーバにクエリーを発行するようになっているため、インターネットに接続されていない状態では「クエリーに失敗しました」というメッセージが出力される。気になる場合には、［Perform CDDB lookup automatically］をオフにしておくといい。

［Proxy］タブはプロキシ経由でインターネットに接続している場合に設定が必要になる。［Get server from ‘ http-proxy ’ env. var］をオンにして、［Proxy server］にHTTPプロキシのURLを、［Proxy port］に「80」を指定しておけば、ほとんどのプロキシ環境でCDDBサーバが利用可能になるはずだ。

運がよければボタン一発！

必要な設定を行い、CDをドライブにセットしたら、インターネットに接続して操作パネルの下段の右から3番めにあるボタンをクリック、CDDBサーバからCDの情報をゲットしよう。無事発見できれば、［Tracks］タブに曲名などの情報が表示される。CD情報が見つからなかった場合は、残念ながらあきらめるしかない。ただし、リップとエンコードは問題なく行えるので安心してほしい。ただ、エンコード後のファイル名は「Track01.mp3」、「Track02.mp3」などになってしまう。

リッピングとエンコーディングの手順はとても簡単だ。［Tracks］タブで［Rip］をクリックするとリッピングの対象としてマークされる（**画面2**）。トラックを右ボタンクリックして、対象から外すこともできる。［Rip］タブに移って［Rip＋Encode］をクリックすれば、あとは自動的にGripが各設定オプションをパラメータとしてコマンドを呼び出して、マークしたトラックを処理してくれる。

MP3ファイルは、ユーザーのホームディレクトリの「mp3」ディレクトリ以下に保存されているはずだ。デフォルトではアーティストとアルバムごとにディレクトリが作成されて、アルバム名の付いたディレクトリにMP3ファイルが置かれる。リップしたWAVファイルは、自動的に削除される。

mp3ディレクトリにはプレイリストファイルも作成される。ファイル名は「アーティスト名－アルバムタイトル.m3u」となる。このプレイリストはXMMSなどのMP3再生ソフトで利用できる。プレイリストファイルは、演奏順にファイルパスが記録されている単純な書式のテキストファイルなので、エディタを使って自由に編集できる。好きな曲だけを集めたプレイリストを作ることも可能だ。

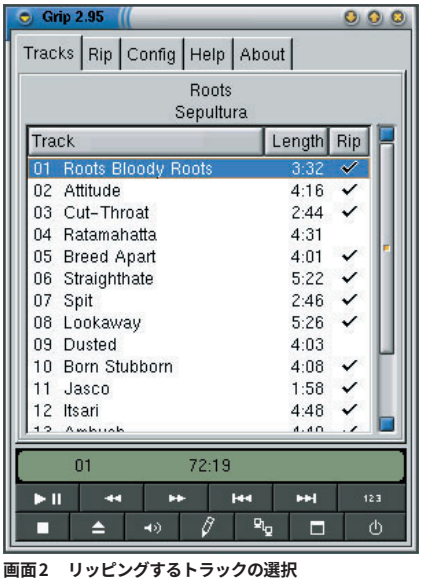
なお、CDDBサーバで検出できなかったCDについては、アーティスト名が「noaritist」、アルバムタイトルが「unknown_disc」となる。MP3ファイル名は、前にも触れたように「Track XX.mp3」になる。そのままでは、2枚目以降のCDのデータで上書きされてしまう可能性があるので、ディレクトリ名を変えるなどの処置が必要になる。せっかくの便利な機能をフル活用するためにも、CDDBで検出されることを祈ろう。

以上で、MP3へのエンコードは完了だ。お次は再生するソフトウェアを紹介していこう。

マルチメディアプレーヤ「XMMS」

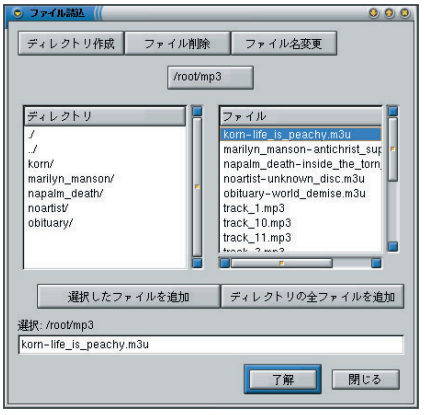
GNOMEがインストールされている環境なら、「XMMS」というソフトウェアが入っているはずだ（**画面3**）。XMMSは、MP3、CD-DA（オーディオCD）、WAVファイルなどを再生できるマルチメディアプレーヤ。プラグインによる拡張機能を備えており、各ファイルフォーマット用のプレーヤ機能はプラグインとして提供される。MP3用のプラグインは標準で用意されているので、そのままMP3プレーヤと

画面1　Gripのリッパー設定画面



画面2　リッピングするトラックの選択

画面3　XMMSのメインウィンドウ
ウィンドウ下部の操作ボタンはCDプレーヤなどでお馴染みのモノ。ランダム／リピート再生、イコライザなどの標準的な機能も備えている。



画面4　ファイル選択ダイアログ
この画面のようにプレイリストを選択して再生することもできる。

して使える。

Vine 2.1では、GNOME、Window Makerともに、デフォルトでメニューに登録されている。GNOMEの場合は［プログラム］－［マルチメディア］、Window Makerでは［アプリケーション］－［マルチメディア］にXMMSのメニューアイテムがある。ちなみに、コマンドラインから起動する場合のコマンドは、まま「xmms」でいい。

画面3からわかるように、通常のオーディオプレーヤと同じボタンインターフェイスなので再生や停止、頭出しについては説明の必要はないだろう。再生するMP3ファイルを指定するには、左上隅のボタンをクリックしてメニューから［ファイルを再生］を選択するか、オーディオ機器の「取り出し」にあたるボタンをクリックする。**画面4**のダイアログが表示されるので、あとはファイル名を選択して［了解］をクリックすれば自動的に再生が開始される。CtrlキーまたはShiftキーとマウスクリックのコンビネーションで複数のファイルを選択することも可能だ。

また、前述のプレイリストファイル（拡張子.m3u）を選択することもできる。この場合、プレイリストの指定に従って再生が行われる。プレイリストは、使いこなすと非常に便利な機能なので少し詳しく見てみよう。

マイ・フェイバリット・リスト

前にも触れたようにプレイリストファイルは、再生順にディレクトリパスとファイル名が記述されているテキストファイルだ。つまり、お気に入りの曲のMP3ファイルが置かれているパスを再生したい順に記述するだけで、手軽に「フェイバリット・アルバム」が作れるわけだ。

たとえば、「Artist-A/Album-01」と「Artist-B/Disc-01」ディレクトリにMP3ファイルがあるとして、両方のディレクトリから曲をピックアップして指定順に再生する場合、**リスト1**のように記述すればよい。

リスト1ではパスを相対指定している。Gripのデフォルト設定で作成されるプレイリストファイルでも、ディレクトリパスは相対パスで記述される。これは、ディレクトリツリーごと移動したり、CD-Rに保存したりする場合を考えるとこうしておくほうが便利だからだ。自分でプレイリストを作成するときにも、相対パスでディレクトリを

▼リスト1　プレイリストの例

```
artist-A/Album-01/Track-02
artist-B/Disc-01/Track-01
artist-B/Disc-01/Track-12
artist-A/Album-01/Track-08
        :
```

指定するようにしよう。

Vineでのサウンド設定

ちょっと順番が逆になってしまったが、最後にサウンドの設定について説明しておこう。各ディストリビューションには、OSS/FreeまたはALSAというフリーのサウンドドライバパッケージが含まれている。Vine 2.1では、OSS/Freeが採用されている。

サウンドカードの設定は専用ツール「sndconfig」で行う（コマンド名も同じ）。起動するとサウンドカードが自動検出され、指示にしたがっていけば、必要な設定とドライバのロードまで行ってくれる。サポートされているサウンドカードであれば、設定は非常に簡単だ。**表1**にサポートされている主なサウンドカードを掲載しておくので購入の際などに参考にしてほしい。

ボリューム調整は、GNOMEの場合は［プログラム］－［マルチメディア］－［オーディオミキサー］、Window Makerでは［ユーティリティ］－［サウンド］－［ミキサー］で行える。

これでLinuxマシンでのMP3環境はひと通り揃うはず。さっそく実行して、充実のPC音楽ライフを過ごそう！

ベンダー	サウンドカードまたはチップ	ドライバモジュール
Creative Media	Sound Blaster Live !	emu10k1.o
YAMAHA	YFM744／754	ymfpci.o
ESS Technology	ESS Solo-1	esssolo1.o
	ESS Maestro-1／2／3	maestro.o
C-Media Electronics	CMI8338／8738	cmipci.o

表1　Vine 2.1で標準サポートされている主なサウンドデバイス

フォトレタッチに挑戦しよう!

～デジカメデータの加工とプリント～

デジタルカメラが発売された当初は、PCの家庭への普及率が今ほど高くなく、TVモニタに接続するためのアダプタや専用プリンタが主に使われていた。

現在ではPCの普及が進み、データをPCに取り込んで加工・プリントすることが一般的になっている。「Linuxでは、はたしてどこまでできるのか」を試しながら、必要なソフトウェアや設定方法について解説していこう。

画像データの取り込み

デジタルカメラの画像データは、本体にセットされている記録メディアに保存される。現在主流になっているメディアは、コンパクトフラッシュカードとスマートメディア、それにソニー製品で使われるメモリースティックの3種類だ（写真1）。

これらのメディアに保存されたデータを取り込むには、カメラ本体に付属しているPCとの接続インターフェイスを使うか、各メディア専用のハードウェア（リーダ／ライタやFDドライブアダプタ）を利用する。いずれの場合も、Windowsマシンとの間でなら付属のドライバや専用ソフトを使って手軽にデータ交換できるが、Linuxでの対応状況は機種ごとにまちまちである。機種ごとに細かく見ていくことはできないので、それぞれのケースごとに情報の入手先やヒントを紹介しておこう。

デジカメ－PC間の接続インターフェイスは、シリアルまたはUSBが一般的。シリアル接続の場合、Linuxでは「gPhoto」というデジタルカメラの画像データを取り込んで、管理するためのツールが利用できる。サポートしている機種も多く、ユーザーの評価も高いツールである。USBに関しても準備が進められており、次期バージョンでは一部の機種がサポートされる模様だ。

残念ながらgPhotoは、Vineには収録されていない。試してみたい方は開発プロジェクトのWebサイト（http://www.gphoto.org/）でサポート機種や入手方法などの詳細な情報を収集してみよう。

専用のカードリーダでは、SCSI接続のものが動作するようだ。メーカーが情報を公開しているものとしては、メルコのPCカードリーダ（MCR-S／SFB／MCR-S2／S2FB）がある。正式なサポートではないが、同社のWebサイトにはLinuxでの設定に関する情報も掲載されている（http://buffalo.melcoinc.co.jp/taiou/os/linux/）。

USB接続の場合、デジカメ本体の接続インターフェイス、リーダとも状況はあまり芳しくない。Vine 2.1では一部のUSB機器をサポートしているが、動作する機種はかなり限られてしまう。2.4.X系のカーネルではUSBサポートが大幅に強化されているので、USB接続の機器を利用している場合はカーネルをバージョンアップしたほうがよい（Vine 2.1のカーネルはバージョン2.2.17）。最新のサポート状況は、Linux USB ProjectのWebサイト（http://www.linux-usb.org/）が参考になる。

今回のサンプルデータの撮影に使ったソニーの「Syber-Shot DSC-P1」（**写真1**）の場合も、カーネル2.4.2にバージョンアップすることで、付属のUSBインターフェイスケーブルを介してデータを取り込むことができた。カーネルのバージョンアップについては、この特集の範囲を超えるものなので、大変申し訳ないが、割愛させていただく。チャレンジされたい方は、本誌のバックナンバー（2001年3月号）や書籍、Linuxの情報サイトなどを参照してほしい。

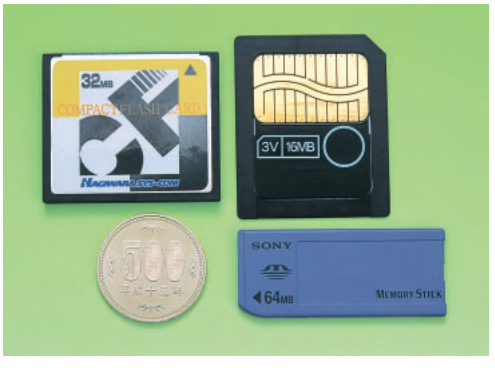


写真1　デジカメのデータ保存に使われる記録メディア
左上から時計回りに、コンパクトフラッシュ、スマートメディア、メモリースティック。一般に使われる範囲では、容量もほぼ同程度で、64Mバイトまたは128Mバイトのものが多い。

写真2　ソニー Syber-Shot DSC-P1
113.0（W）×53.9（H）×43.8（D）mmと非常にコンパクト、かつ重さもわずか250g。体格のいい人は逆に扱いづらいのではないかと、余計な心配をしてしまう。

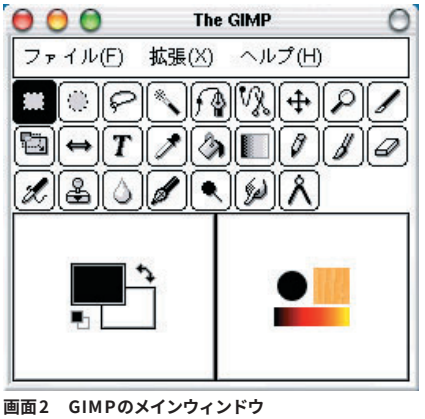
GIMPでフォトレタッチ

さて、あまり威勢のよくない話になってしまったので、画像データの加工に話を移そう。この分野にはオープンソースソフトウェアの優等生である「GIMP」があるのだ。

GIMPは非常に多機能な画像作成・フォトレタッチツールだ。今回紹介するフォトレタッチに関しても、さまざまなフィルタや効果を標準で備えていて、ほとんどの操作はマウスの右ボタンメニューで表示されるポップアップメニューから行うことができる。また、Vine 2.1をはじめ、多くのディストリビューションで標準インストールされるので、Linuxであればどの環境でも利用できる。

では、さっそくGIMPを立ち上げてみよう。Vine 2.1では、標準でXのメニューに登録される。GNOMEならメインメニューの［グラフィックス］－［GIMP］、Window Makerならメインメニューの［アプリケーション］－［グラフィックス］－［Gimp］で起動できる。

初めて起動した場合、**画面1**に示すようなインストール画面が表示される。



画面2　GIMPのメインウィンドウ
選択、スポイト、ブラシなど、ドローツールではお馴染みのアイコンが並ぶ。下段のパネルは、左側が前景／背景色、右側がブラシ／パターン／グラデーションの現在の設定を示している。

ここは指示に従っていけば、特に問題が起こることはないはずだ。

メインメニューは非常にシンプルな構成になっている（**画面2**）。［ファイル］－［開く］として、表示されるダイアログでファイル名を選択し、加工する画像データを開いてみよう。このあとの操作は画像の任意のポイントを右クリックして、ポップアップメニューを表示して行う。

写真3を見てほしい。先ほど紹介したソニーのCyber-Shot DSC-P1で撮影して（撮影技術が拙い点はご容赦を）PCに取り込んだ未加工の画像だ。これをサンプルとしてGIMPの持つ機能をいくつか（本当にほんの一部なのだが）試してみることにしよう。

まず、念のためオリジナルを残しておくために、加工する前に別名で保管しておく。右クリックでポップアップメニューを表示し、［ファイル］－［別名で保存］を選択する。［画像の保存］ダイアログでファイル名を指定して［了解］ボタンをクリックする。こ

画面1　GIMPのユーザーインストール画面
そのユーザーで初めて起動したときに表示される。データを保存するディレクトリパスやデフォルトの解像度などのユーザー設定を行う。

のようにGIMPのユーザーインターフェースは、WindowsやMacのアプリケーションと大きな違いはない。初めて使う場合でも、操作に戸惑うことは少ないだろう。

■主役を目立たせる

実際の誌面ではどう映るかわからないのだが、サンプルの画像は「主役」であるパンジー（ですよね？）がややくすんだ感じで背景に沈んでしまっている印象がある。このあたりを調整し

写真3　サンプル画像データ
DSC-P1の最大画像サイズ（2048×1536ピクセル）で撮影した。せっかくの高解像度も腕前が追いつかず……。

写真4　縁を強調するフィルタを適用したサンプル
違いを際立たせるためにフィルタの効果を大きめに設定したので、少し不自然な感じになっているかもしれない。

画面3　［NL Filter］の設定画面
プレビューを見ながらパラメータを調整できる。

てみよう。

ポップアップメニューから［フィルタ］－［強調］－［NL Filter］を選んぶと**画面3**のダイアログが表示される。［縁強調］をオンにしてプレビュー画面を見ながらパラメータを調整し、［了解］をクリックするとフィルタが適用される。違いをハッキリさせるために、少し大げさに強調したものが**写真4**だ。

このほかにも、［画像］－［色］で表示されるメニュー項目で、彩度や明るさ、コントラストなどを調整して鮮やかな感じにすることができる。

■特殊な効果で遊んでみる

画像をセピア調に一発変換するScript-Fuを試してみた（Script-FuはGIMPに新しい機能を追加するための仕組み）。ポップアップメニューから［Script-Fu］－［装飾］－［古い写真］を選び、ダイアログのデフォルトのままオリジナルを変換したものが**写真5**だ。ややクリアすぎる感じだが、さらに色調を変えていけば落ち着いた感じになりそうだ。

写真6は、［フィルタ］－［芸術効果］メニューにある［油絵］と［キャンバス効果］を実行した結果。こちらは何度かパラメータを調整して、それっぽいものに仕上げたつもりだ。

GIMPにはほかにも、さまざまな効果を持つフィルタが用意されている。あれこれ試してみて、自分なりのテクニックを編み出そう。

Linuxでのプリンタ設定

せっかく撮影した写真なので、家族や友人などに見せたいこともあるはずだ。デジカメの画像データはファイルサイズが大きいので、メールに添付して送るのは避けたいところ。最近ではカラーインクジェットプリンタが安くなってきているので、ここはひとつ試してみることにしよう。

今回、編集部でテストに使用したのはエプソンのPM-720Cという機種（**写真7**）。PM-720Cはエプソンの PMシリーズの中では最もローエンドなモデル

写真5　Script-Fuの［古い写真］適用後のサンプル

写真6　2つのフィルタ効果で油絵風にしたサンプル

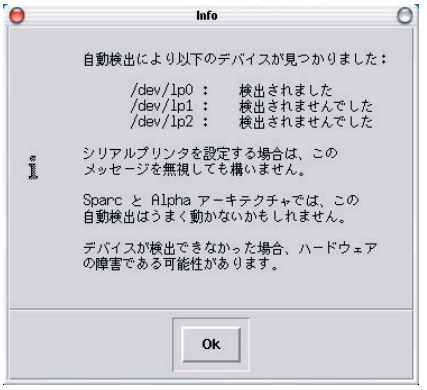
で、標準的な実売価格は2万円を切っている。

エプソンのプリンタについては、関連会社のエプソンコーワから、「Photo Image Print System」というLinux用のドライバとプリンタ設定ツールをまとめたパッケージが無償配布されている。Vine 2.1には、標準でこのパッケージが収録されており、PMシリーズの750C、760C、770C、800C、820C、2000C、3000C、3300Cがサポートされている。

実は購入してからPM-720Cがサポートされていないことに気づいたのだが、皆さんはこのようなことのないようにしてもらいたい。で、どうしたのかというと、720Cの「兄貴分」にあたるPM-780C用の新しいドライバをエプソンコーワのWebサイト（http://www.epkowa.co.jp/）からダウンロードしてきてテストすることにしたのであった。結果やいかに……。なお、PM-720Cはパラレルポートと USBポート接続に対応しているが（**写真8**）、今回は無難なところでパラレルポート接続でテストを行っている。

RPMで配布されているので、パッケージのインストールは簡単だ。

```
# rpm -Uvh pips780-1.3-1.i386.rpm
```



画面4　プリンタの自動検出
Print Toolがプリンタを検出すると、このようなメッセージパネルが表示される。どのデバイスファイルがプリンタに割り当てられているのか確認できる。

必要な設定は、/usr/doc/pips780-1.3にインストールされるreadme780ファイルに詳しく説明されている。ようは、システムのプリント設定ファイルである/etc/printcapに適切な記述を追加すればよいので、ここはひとひねりして「Print Tool」を使って雛型を作成し、それを修正する方法を紹介しておこう。Print ToolはRed Hat系ディストリビューション独自のプリンタ設定ツールで、プリントキューの作成とプリントフィルタの設定をGUI操作で行うことができる。

起動は、Xのターミナルエミュレータで「printtool &」とする。［追加］ボタンをクリックすると、プリンタの種類を指定するダイアログが表示される。パラレルポート接続なので［ローカルプリンタ］を選択して［了解］ボタンをクリックする。ここでプリンタの自動検出が実行され、**画面4**に示すウィンドウが表示されるはずだ。プリンタが検出されない場合は、ケーブルの接続やBIOSでのパラレルポートの設定を見直してみよう。

［OK］をクリックして次の画面に進み、デフォルトの設定のまま［了解］ボタンをクリックする。readme780には、「必ずデフォルトのプリンタ設定以外に新しくプリンタ設定を作成して下さい」と記述されているので、もう一度同じ手順を繰り返して、プリンタ設定を追加する。

ここまでの操作で「lp」と「lp0」という2つのプリンタの設定が、/etc/printcapに追加されてるはずだ。あとは780C用のプリントフィルタを使うように修正を加えれば設定完了。完成したprintcapは**リスト1**のようになる。

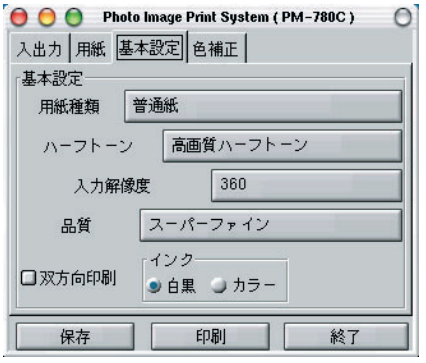
▼リスト1　printcapの例

```
lp:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:\
        :if=/usr/local/EPKowa/
         PM780C/filter780:\

lp0:\
        :sd=/var/spool/lpd/lp0:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:\
        :if=/usr/local/EPKowa/
         PM780C/filter780:\
```

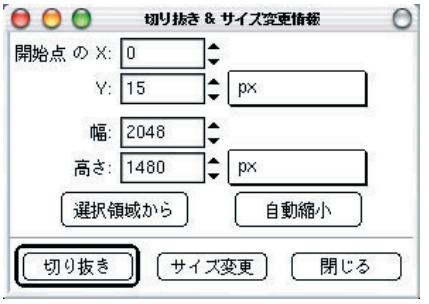
写真7　EPSON PM-720C
安さにつられてツイ買ってしまったが、なかなかのハイクオリティ。ハーフトーンなども、美しくプリントアウトできる。が、スピードは遅い。

写真8　EPSON PM720Cの背面

画面5　pips780の［用紙］タブ
用紙サイズをはじめ、拡大／縮小率、マージンを設定する。［給紙方法］はオートシートフィーダのみ選択可。



画面6　pips780の［基本設定］タブ
用紙の種類、解像度、プリント品質などを指定。720Cでは、［双方向］をオンにしないほうがよい。



画面7　GIMPの［切り抜き＆サイズ変更］ツール
開始位置と変更後のサイズをピクセル単位、インチ単位、パーセント指定などで設定する。

画像データの調整とプリント

Linux側の設定の次は、Photo Image Print Systemに付属のツール「pips780」を使ってプリンタの設定とプリントテストを行ってみよう。

pips780はPNG形式のファイルを標準プリンタに出力するので、テスト用のPNGファイルを作成する必要がある。GIMPで適当な画像ファイルをPNG形式で保存するといいだろう。準備ができたら、次のコマンドを実行してpips780を起動する。「-la JP」は日本語モードを指定するオプションだ。

```
$ pips780 -la JP
```

テストなので、用紙はA4サイズの普通紙を使おう。pips780でも、それに合わせて設定する。とりあえず、インクは黒で試すほうが無難だ。これらの設定は［用紙］タブ（**画面5**）と［基本設定］タブ（**画面6**）で行う。プリントするファイルの指定は［入出力］タブだ。

モノクロのプリントがうまくいったら、インクの設定を変更してカラープリントも試してみよう。編集部で行ったテストでは、どちらも無事成功した（誤って720Cを購入した担当編集がホっとしたことは言うまでもあるまい）。

テストがうまくいったら、いよいよデジカメで撮影した画像データのプリントだ。各プリンタごとに写真用紙が発売されているので、素直にメーカー純正のものを使うのがよい。テストでは、PMシリーズ用の「フォトカード2」を試してみた。PM-720C自体はTV-CMでもおなじみの「縁なし印刷」には対応していないが、この用紙は縁の部分にミシン目があって切り取ることができるようになっている。うまくプリントサイズを調整すれば、「擬似」縁なし印刷が可能なわけだ。

あれこれ設定を調整してうまくプリントできた結果が**写真9**である。pips780を使ってPM-720Cに実際にプリントしたものを高精細のスキャナで取り込んだ画像なので、誌面上でも実物に近い感じで再現されているはず。2万円以下のプリンタで、これだけのクオリティとは、ちょっと感動モノである。

ほかのプリンタで出力する場合の参考にもなるだろうから、プリントの調整で使ったGIMPのテクニック（というほどもないのだが）を紹介しておこう。ドライバやプリントツールで画像の向きを変えられないときは、ポップアップメニューの［画像］－［変換］－［回転］を使って画像データ自体の向きを変えるといい。また、縦横比が写真用紙にうまく合わないときは、［道

写真9　サンプルデータを720Cで出力

具］－［変換ツール］－［切り抜き&サイズ変更］を使って比率を微調整しよう（**画面7**）。

pips780には、プリント時のサイズを変更するオプションがあったので用紙サイズに合わせられた。ドライバ側で設定できない場合は、GIMPの印刷オプションで［PPI］を設定してサイズを変更し、プリント位置、マージンを調整するとよい。ただし、GIMPからの出力では画面の色調とはかなり異なるプリント結果になる（GIMPでのプリント設定については、下のコラムも参照のこと）。

そのほかのプリンタのサポート状況

PMシリーズのLinux用ドライバの開発元であるエプソンコーワでは、エプソンのレーザープリンタ「LPシリーズ」のLinux用ドライバも開発・無償配布している。同社のWebサイトによると、LP-8500C、LP-8300C、LP-8200C、LP-3000C、LP-8600FX、LP-1900などがサポートされている。

エプソン以外では、キヤノンがLinux用プリンタドライバを無償配布している。サポート対象は、BJ F850、BJ F860、BJ F870だ。キヤノンではWebサイトでの情報提供も行っている（http://www.canon-sales.co.jp/faq/linux/index-j.html）。市販されているドライバパッケージやフリーのプリントフィルタでのサポート状況がまとめられているので、キヤノンユーザーは一度訪れてみるといいだろう。

実際のところ、フリーのプリントフィルタによるサポートはかなり充実している。これは、Print Toolを起動してプリンタエントリの追加または編集を行えば確認できる。エントリの追加（編集）ダイアログにある［入力フィルタ］の［選択］ボタンをクリックすると、**画面8**が表示される。左側の［プリンタの種類］の一覧から、数多くのプリンタがサポートされていることがわかる。

画面8　Print Toolのフィルタ設定ダイアログ

「プリントアウトはWindowsから」なんて悲しいことは言わないで、ぜひLinuxマシンにもプリンタを接続してあげてほしい。

GIMPと手頃なカラープリンタがあれば、結構お気軽に楽しめる。あなたも始めてみてみませんか？

ステップアップ講座

Linuxのプリントシステム

Linuxのプリントシステムは、「lpd」というプログラムによって制御されている。lpdは常にバックグランドで待機して、プリントジョブが送られてくるとその要求どおりにプリントデータをスプールし、プリンタポートを介してプリンタにデータを送る。プリントジョブをlpdに送るためのコマンドは「lpr」という。以下のように実行すれば、どのプリンタ（この場合は「lp0」）に出力したいのかlpdに知らせることができる。

```
$ lpr -Plp0　<ファイル名>
```

特に設定が行われていなければ、lpdはファイルデータをそのままプリンタに受け渡すだけなので、たとえばテキストファイルであれば、テキストデータと改行コードなどが直接プリンタに送られる。テキストデータに対応するフォントをプリンタが持っていれば正常にプリントアウトされ、フォントを持っていなければ文字化けが発生する。

「/etc/printcap」は、lpdが参照する設定ファイルだ。lpdが制御するプリンタスプールやプリンタデバイス名などが設定されている。printcapに「if=」が設定されている場合、lpdはそこに指定されている「プリント（入力）フィルタ」にデータを通してからプリンタに送るという処理を行う。プリントフィルタの役割は、ファイルの種類を判断して、正常にプリントされるようにプリンタ制御コードなどを付加することである。これで、先ほどの文字化けの問題は回避される。データがテキストであると判断され、フォント情報を付加したプリントデータがプリンタへと送られるようになるからだ。

テキストファイル以外のケースを考えてみよう。伝統的にUNIX系OSの標準プリントフォーマットとなっているPostScriptではどうだろうか？　送り先がPostScriptプリンタであれば、もちろん正常に印刷されるが、それ以外のプリンタでは印刷できない。そこで登場するのが「Ghostscript」である。Ghostscriptは、PostScript（またはPDF）フォーマットのデータを、さまざまなプリンタで出力できるように変換するプログラムだ。

lprコマンドを使ってPostScriptファイルのデータがlpdに送られると、プリントフィルタがGhostscriptを呼び出す、Ghostscriptは正しくプリントされるようにフォーマット変換を行って最終的なプリントデータを生成してlpdに制御を戻す、という処理の流れになる。

そのほかのファーマットのファイルもPostScriptに変換すれば、この仕組みを使ってプリンタの種類を問わずLinux環境でプリント可能になる。GIMPやNetscape Navigatorなどは、PostScriptによるプリント機能を備えているので、これらのアプリケーションで扱えるファイルをプリントアウトすることができる。

最後に、プリントスプールを管理するためのコマンドを紹介しておこう。

●lpq
オプションなしで実行するとデフォルトプリンタのスプール情報が表示される。「－P」でプリンタを指定することもできる。

●lprm
「job　<ジョブ番号>」で指定した番号のジョブを削除する。ジョブ番号はlpqで参照可。

CD-Rを使いこなそう!

～データCDとオーディオCDの作成～

データをバックアップしたり、自分で撮ったデジカメのデータをCD-ROMにまとめて友人に見せたり、好きな曲を集めたオリジナルのオーディオCDを作成したりと、CD-Rが活躍する場面は多い。ここでは、LinuxでのCD-Rの活用法を紹介しよう。

ハードウェアを選ぶ

なにはともあれ、CD-Rを利用するには専用のドライブを用意しないと始まらない。DVD-ROMが必要なければ、CD-RとCD-RWのコンボドライブが値段もこなれてきていて、お買い得感がある。ただし、Linux環境で使うCD-R／RWドライブを購入する際には、接続インターフェイスに注意したい。

ATAPI（IDE）やSCSI接続のものは、ほとんどのドライブが動作すると思っていい。反対に、USBやパラレルポート接続のものは動作しないドライブのほうが多いようだ。ただし、ATAPIやSCSIのドライブでも、PCカードアダプタを介して接続する外付けタイプは動作しない可能性があるので要チェック。このあと説明するCDRecordのWebサイトにも、サポートハードウェアに関する情報（http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html）があるのでそちらも参考にしてほしい。

写真1　動作確認に使用したCD-R/RWドライブ
LITEON ITというあまり馴染みのないメーカーの製品。CD-R 12倍、CD-ROM 32倍、CD-RW 10倍で、BURN-Proofにも対応していながら、1万5980円という格安ドライブだ。

SCSIはアダプタが別途必要なケースが多いだろうから、ATAPI接続の内蔵ドライブが無難だし、お勧めだ。というわけで、以下の説明はATAPI接続のドライブを前提に進めていく。

IDE-SCSIエミュレーションを有効にする

Linuxで利用できるCD-R作成ソフトウェアの中で「標準」ともいえるのが「CDRecord」だ（VineではVine Plusに収録されている）。最新のバージョン1.9では、CD-Rへの書き込みの際のエラーを抑制する「BURN-Proof」機能にも対応している。インストールについては、**48ページ**の「MP3で音楽三昧」で紹介してあるので、そちらを参照のこと。

CDRecordのパッケージは、cdrecord（ライティング）、mkisofs（イメージファイル作成）、cdda2wav（リッパー）といった各機能ごとの専用プログラムで構成されている。このうちcdrecordは、「IDE－SCSIエミュレーション」と呼ばれる特殊な方法でATAPIドライブをサポートしている。

LinuxでIDE－SCSIエミュレーションを利用にするには、以下のサポートがカーネルで有効になっていなければならない。

①SCSIコアサポート
②SCSI CD-ROMサポート
③汎用SCSIサポート
④IDE－SCSIサポート

Vine 2.1の場合、標準で①と②がカーネルに組み込まれており、③と④がカーネルモジュール化されている。モジュールファイル名は、「sg.o」と「ide-scsi.o」である。

また、IDE接続されている機器がIDE－SCSIエミュレーションで動作することを明示的に指定する必要がある。これには、起動時に表示される「boot:」プロンプトで以下のカーネルパラメータを指定すればよい。

```
boot: linux hdc=ide-scsi max_scsi_luns=1
```

「hdc」の部分は環境に合わせてIDE機器のデバイス名を指定すること。たとえば、セカンダリIDEポートに2つのドライブが接続されていれば「hdc=ide-scsi」と「hdd=ide-scsi」の両方を指定する必要がある。

これらのパラメータを与えて再起動したら、sg.oとide-scsi.oを次のようにしてロードする。

```
# modprobe sg
# modprobe ide-scsi
```

ide-scsiモジュールをロードした際にIDE機器に関する構成情報が表示されればOK。最後にCD-ROMドライブのシンボリックリンク「/dev/cdrom」

を再作成すれば完了である。これも環境によって、デバイスファイル名が異なるので注意してほしい。

```
# ln -sf /dev/scd0 /dev/cdrom
```

IDE－SCSIエミュレーションについては、/usr/doc/cdrecord-1.9/README.ATAPIに詳しい説明がある。

X-CD-Roastをインストールする

CDRecordはコンソールで動作するコマンドラインツールで、指定しなければならないオプションも多い。GUI環境でお手軽に使用するためのプログラム（GUIラッパー）を別途用意したいところだ。今回は「X-CD-Roast」を紹介することにしよう。

ここで、お詫びを言っておくべきだろう。残念ながら編集部で行ったテストでは、付録CD-ROMに収録したバージョンのX-CD-RoastはVine 2.1＋update環境において動作確認できなかったのだ。お手数で申し訳ないが、最新のバージョン0.98alpha8のソースRPMパッケージ（ファイル名はxcdroast-0.98alpha8-1.src.rpm）をX-CD-RoastプロジェクトのWebサイト（http://www.xcdroast.org/）からダウンロードしてほしい。バージョンからわかるとおりアルファ版だが、テストした範囲内ではきちんと動作した。

ダウンロードが完了したら、次はインストール。まず、ソースRPMからバイナリRPMを作成する。

```
# rpm --rebuild xcdroast-0.98alpha8-1.src.rpm
```

続いて、できあがったバイナリRPMを使ってインストールを行う。

```
# rpm -Uvh xcdroast-0.98alpha8-1.i386.rpm
```

Xを起動して、ターミナルエミュレータから「xcdroast &」として起動できればインストールは完了だ。

X-CD-Roastの初回起動時には、「設定ファイルが見つからないので設定を行ってください」というメッセージが表示される。ここは素直にしたがって［Setup］ボタンをクリックしよう。

画面1に示すようにSetupダイアログには、各設定項目ごとにタブがある。最初に表示される［Device-scan］タブは、cdrecordコマンドで検出されたCDドライブの一覧だ。使用するドライブが検出されているか確認しよう。

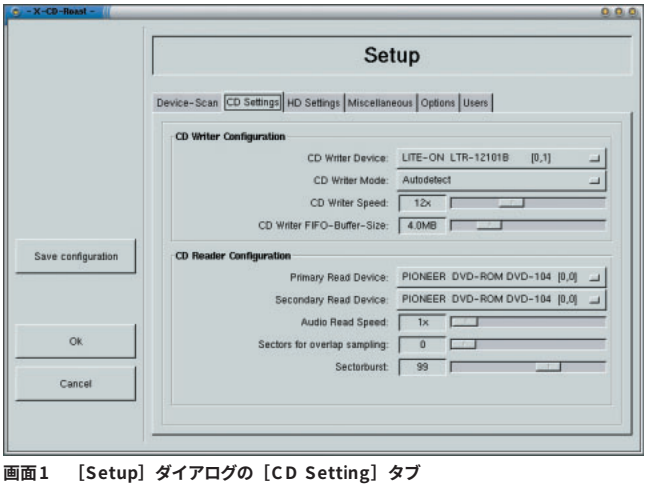
［CD Settings］タブでは、CDへの書き込みを行うドライブとCDからの読み込みを行うドライブを指定する。［HD Setting］タブは、ISO9660イメージやWAV形式のファイルを作成する作業用ディレクトリパスの設定だ。ほかのタブはデフォルトで特に問題はない。各項目にマウスポインタを近づけるとツールチップが表示されるので、必要に応じて設定しよう。

設定したら［Save configuration］ボタンをクリックして設定内容を保存するのを忘れずに。

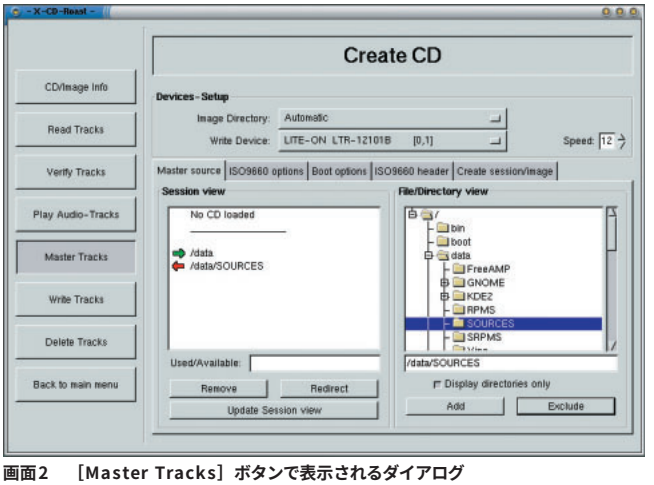
CD-ROMを作成しよう

準備が整ったところで、ハードディスク上のデータをCD-Rに記録する方法から見ていこう。

まずは対象となるデータの選択から。メインメニューで［Create CD］ボタンをクリックし、表示される［Create CD］ダイアログで［Master Tracks］をクリックする。**画面2**が表示されるので、右側のツリーからバックアップするディレクトリを選択して［Add］をクリックする。ツリーの下にある



画面1　［Setup］ダイアログの［CD Setting］タブ
システム上にCD-Rドライブと CD-ROMドライブがあれば、画面のように読み出しドライブと書き込みドライブを設定して、オンザフライでCDをコピーすることも可能。



画面2　［Master Tracks］ボタンで表示されるダイアログ
ディレクトリを選んで［Add］をクリックすると、画面のように左側のセッションリストに追加される。赤い矢印のものはマスタリングの除外対象。

［Display directry only］をオフすれば、個々のファイルを選択できる。

ディレクトリ（またはファイル）を選択して［Exclude］をクリックし、バックアップ対象から除外することも可能だ。［Add］または［Exclude］した結果は、右側の［Session view］に反映される。［Redirect］はCD上でのディレクトリ階層を変更するためにある。たとえば、/dataディレクトリをバックアップするとして、/dataをCDのルートディレクトリにリダイレクトすれば/data以下にあるファイルはCDのルートに記録される。

CD-ROMのフォーマット規格であるISO9660には、さまざまな拡張オプションがある。バックアップなどの目的でパーミッションや所有者といったファイル属性を保存する場合は、［ISO 9660 options］タブで［Rock Ridge only］を選択して［Rock Ridge（Buck Up）］をオンにしておこう。データ交換など、作成したCD-ROMをWindowsマシンでも使いたい場合は、［Rock Ridge＋Joliet］を選択するといい。［ISO9660 options］、［Boot options］、［ISO9660 header］の各タブで指定できるオプションの詳細については、表示されるツールチップが参考になる（**画面3**）。

データの選択とオプションの指定が終わったら［Create Session/image］タブに移動。このタブでは、イメージファイルの作成（マスタリング）のみ行うか、あるいはオンザフライでデータのマスタリングと書き込みを同時に行うことができる。確実性を求めるなら、［Master to image file］をクリックして一度イメージファイルを作成してから、［Write Tracks］ダイアログに移ってCD-Rへ書き込みを行うほうがよい。どちらの場合も、［Simulation write］オプションをオンにしておけば、実際にメディアには書き込まないで「テスト」することもできる。

オリジナルのオーディオCDを作ろう

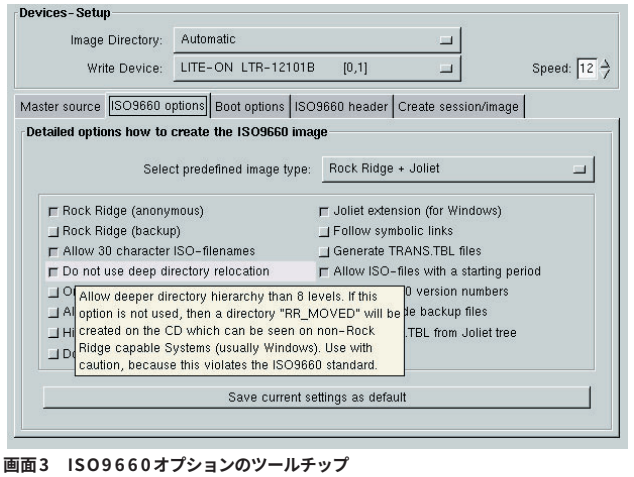
せっかくなので、オーディオCDの作成にもチャレンジしてみよう。

元になるオーディオCDを読み込み用に指定したドライブにセットし、メインメニューの［Create CD］をクリックする。［Create CD］ダイアログで［Read Tracks］をクリックすると**画面4**が表示される。ここで［CD information］でトラックを選択して［Read selected tracks］をクリックすればCDから曲データがリッピングされる。

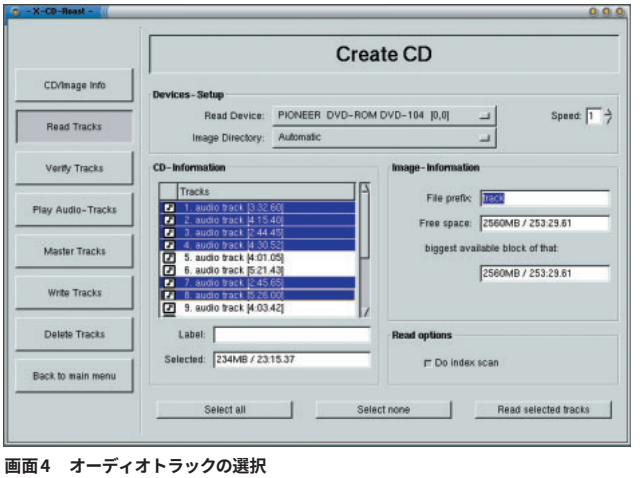
複数のCDからトラックを選択したい場合は、［CD/image info］に戻ってメディアを交換して［Update］をクリックし、もう一度［Read Tracks］ダイアログに戻ってトラックの選択とリッピングを行えばよい。ただし、ダイアログの右側にある［File prefix］を変更しないと同じ番号のトラックのデータが上書きされるので、CDごとに異なるプレフィックスを指定すること。たとえば、1枚目は「cd1-track」、2枚目は「cd2-track」などとする。冗長になるが、アーティスト名やアルバムタイトルを指定すれば、よりわかりやすいだろう。

すべてのリッピングが完了したら、［Write Tracks］ダイアログに移動する。このダイアログの［Layout tracks］タブでは、最終的な曲順を指定する。［Image information］リストからトラックを選択して［Add］ボタンで［Track to write］リストに追加していけばよい。複数選択した場合、クリックした順番に追加されることに注意しよう。

あとは［Accept track layout］で曲順を確定し、［Write tracks］タブの［Write tracks］ボタンでCD-Rに書き込めばオリジナルCDの完成だ。



画面3　ISO9660オプションのツールチップ
この画面だけでなく、すべてのダイアログでツールチップが表示される。ツールチップは、［Setup］の［Options］タブでオフすることができる。



画面4　オーディオトラックの選択
Shiftキーまたは Ctrlキーを押しながらクリックすれば、画面のように、複数のトラックを任意に選択できる。

ここに注意！

ハードウェア選びのコツ

「コツ」などと書いてみたものの、実際にはLinux magazine編集部でも結構失敗している。この特集のために購入した右の写真のカードリーダは、そのほんの一例である。

そんなわけで少々心もとないのだが、購入時のポイントをまとめてみたので参考にしてほしい。なお、編集部で動作を保証するものではない点は、どうかご了承願いたい。

■ビデオカード（XFree86での動作）

現在の主流であるAGP接続のものは、広範にサポートされている。情報はXFree86 ProjrctのWebサイト（http://www.xfree86.org/）に詳しい。バージョン3.3.6と4.0.xでのサポートの違いに注意しよう。

トップベンダーであるNVIDIAも自社開発のドライバを配布している（http://www.nvidia.com/Products/Drivers.nsf/Linux.html）。同社のカードで3Dグラフィックスを描画する場合は必須である（2Dの描画はXFree86で標準サポートされている機種もある）。

■LANカード（補足情報）

本文内では触れなかった具体的な機種（PCIもの）について。3ComのEtherLinkシリーズ、IntelのEtherExpressとPRO/100シリーズがドライバの開発歴も長く動作実績は高い。Intelは自社開発のドライバも提供している（http://support.intel.com/からダウンロードページを検索のこと）。

コレガやプラネックスコミュニケーションズなどの一部ベンダーは、動作確認／保証された製品のパッケージに「Linux Ready」または「Linux OK」と表記して販売している。探してみよう。

低価格な製品に搭載されていることの多い、RealtekのRTL8139B、VIAのVT86C100Aは、編集部でも何度か動作確認している。コスト最重視の方は一度試してみてもよいかもしれない。

■PC（PCMCIA）カード製品

LANカード、無線LAN、モデム、SCSIアダプタ、CD-ROMなどのATAPI機器など、数多くの機種がサポートされている。/etc/pcmcia/configファイルでサポート製品を確認可能。最新情報は、http://sourceforge.net/projects/pcmcia-cs/を参照のこと。

■USB機器

USB製品の中ではスピーカの汎用性が高い。基本的には、USBオーディオクラスドライバ（audio.o）で動作する可能性が高い。

残念ながら動作しなかったアイ・オー・データ機器のフラッシュカードリーダ「USB-DFRW」

ステップアップ講座

カーネルモジュールを知る

本文中にも何度か登場した「カーネルモジュール」について解説しておこう。

■基本的な知識

カーネルモジュールとは、カーネルの機能の一部を1つの部品として取り出した（モジュール化した）もので、必要に応じてメモリへのロード／アンロードが自由に行えることから「カーネルローダブルモジュール」とも呼ばれる。カーネルモジュールのほとんどはデバイスドライバとして動作する。また、サードパーティ製のLinux用デバイスドライバも、カーネルモジュールとして提供されることが多い。つまり、ハードウェアを利用するためには、それに合ったカーネルモジュールが必要になるわけだ。

システムにどのようなカーネルモジュールがあるのか調べるには、以下のパスを探ってみるとよい。

```
/lib/modules/<カーネルバージョン>/
```

■modutils

カーネルモジュールを管理するためのツール群をまとめたパッケージが「modutils」だ。modutilsに含まれるコマンドを使えば、カーネルモジュールのロード／アンロード、現在ロードされているモジュールの表示などができる。以下のコマンドを実行してみよう。

```
# lsmod
```

「Module」の下に何か表示されていれば、そのカーネルモジュールがロードされているということだ。「Used」が「0」以外の数字である場合は、ほかのプログラムから参照されていることを示している。

モジュールのロードには、「insmod」または「modprobe」コマンドを使う。modprobeのほうがオートマティカルで、指定したモジュールのロードに必要なほかのモジュールがあれば、そちらもいっしょにロードしてくれる。

insmodは指定したモジュールをロードするだけだが、そのぶんオプションが多く柔軟な使い方ができる。

コマンドラインの書式は、（特別なオプションを指定しなければ）どちらも同じで、コマンド名に続けてモジュール名を指定するだけでよい。以下の例は、modprobeで「cmpci」というモジュールをロードしている。

```
# modprobe cmpci
```

アンロード用のコマンドは「rmmod」。通常は、modprobe（insmod）と同じようにモジュール名を指定して実行する。モジュールを指定しない場合は、未使用のモジュールがすべてアンロードされる。

デバイスとモジュールの関連付け、ロード時のオプションを定義したファイルが/etcディレクトリにある「conf.module」である（ディストリビューションによってはmodules.conf）。このファイルは、modprobeとinsmodの実行時に参照される。

デスクトップをカスタマイズ

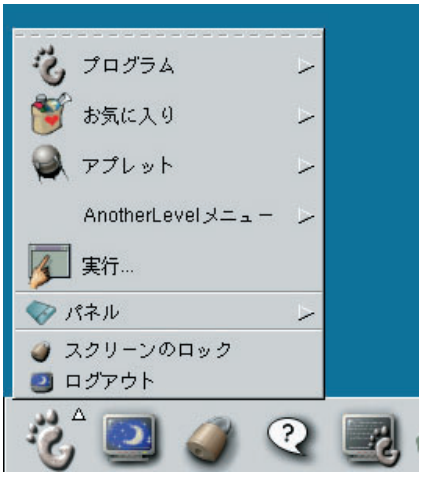
見た目も重要でしょ

インターネットに接続してメールを出したり、ネットサーフィンもした。音楽CDからMP3データも作ったし、GIMPで写真の加工もしたし、できあがったデータをCD-Rに焼いたりもした。この間ずっと同じデスクトップ画面を眺めていたわけで、そろそろ飽きてきたころだと思う。そこで最後に、メニューやデスクトップのカスタマイズをしてみよう。

メニューのカスタマイズ

GNOMEでプログラムを起動する際には、メインメニューから選択したり、パネル上に置いてあるアイコンをクリックしたりする（**画面1**）。

あらかじめ用意されているメニューは万人向けなので、それなりに使いやすくできている。だがユーザーによって使い方はさまざまだから、「こうなっていればいいのに」と思うことだってあるはずだ。そう感じたら、早速カスタマイズしてしまおう。



画面1　GNOMEメインメニュー
プログラムを起動するのは、ここから。

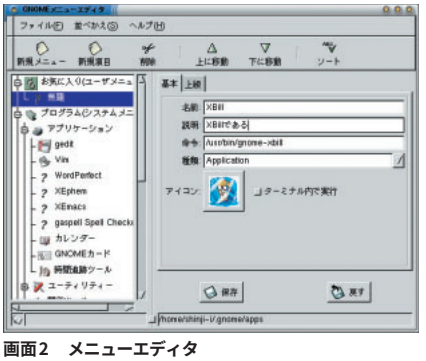
■メニューエディタ

メインメニューから［プログラム］－［デスクトップ設定］とたどっていき、「メニューエディタ」を起動する（**画面2**）。左側にメインメニューの中身がツリー状に表示されているのがわかるだろう。操作は非常に簡単で直感的に行える。追加したいフォルダを選択し、ツールバーで「新規項目」を選べばいい。あとは、ファイルの格納場所やプログラム名などを右側で指定して、保存すれば完了だ。

ところが、メニューに項目を追加しようとしても、

「あなたには権限がありません」

と怒られてしまうことがある。メインメニューの［プログラム］以下は、システムメニューであり、変更するにはroot権限が必要になるのだ。だから、一般ユーザーでは［プログラム］以下に項目を追加、または削除したりすることはできないのだ。自分だけで使っているマシンならば、root権限を得てから上記の操作をすればいいのだが、会社や学校のマシンではそうもいかないだろう。その場合は、［お気に入り］メニューに追加していこう。項目の追加だけでなく、フォルダ（メニュー）の追加もできるので、登録数が多くなったら分類しておくといいだろう。



画面2　メニューエディタ
メインメニューのカスタマイズを行う。

■たねあかし

メインメニューの［プログラム］の中身は、

```
/usr/share/gnome/apps
```

以下のファイルとディレクトリにある。root権限があるならばメニューの構成をまとめて変更したい場合などは、ここに置いてあるファイルをまとめて操作するほうが早いかもしれない。また、［お気に入り］メニューの中身は、ホームディレクトリの

```
~/.gnome/apps
```

以下にある。

■パネルにアイコンを追加

使用頻度が高いプログラムは、パネルに登録しておこう。［パネル］－［パネルに追加］内の「ランチャ」という項目を選択すると、メニューエディタの右側と同じような画面が出てくるので、必要な項目を設定して保存すればいい。

適当にパネルにアイコンを追加していくと、アイコンの間隔がまちまちになり、何となく落ち着かない感じになる。パネル上のアイコンは、マウスの中ボタン（2ボタンマウスを使っている

場合は、左＋右）でドラッグできるので、そうやって揃えてもいいのだが、実はSHIFTキー＋中ボタンでドラッグすると、複数のアイコンをまとめて「押す」ことができる。パネルが混雑してきたら、端のアイコンをこのやり方で押してやれば、きれいに整列させることができる。

デスクトップテーマを選ぼう

最近のウィンドウマネージャには、たいてい「テーマ」を選択して外観を変える機能が備わっている。せっかくだから、いろいろと試してみよう。

正確にはGNOMEはデスクトップ環境であり、ウィンドウマネージャではない。Red Hat系ディストリビューションでGNOMEを利用する場合、たいていはSawfishというウィンドウマネージャと組み合わせられている。

上記の組み合わせで使用する際に注意しなければいけないのは、GNOMEとSawfishそれぞれに「テーマ」機能があるということだ。

■Sawfish

ウィンドウフレームのデザインを変更するには、Sawfishのテーマ機能を利用する。デスクトップ上で中ボタンを押して出てくるメニューから、[カスタマイズ]－[外観]とたどると、Sawfishのテーマ設定画面が表示される(**画面3**)。GNOMEのコントロールセンターから、[Sawmillウィンドウマネージャ]を選択してもいい。なお、Sawmillというのは、Sawfishの昔の名前だ。商標の問題があり、Sawfishに変更されたらしい。

デフォルトでは、タイトルバーの左端が青い「micro GUI」になっている。これ以外には画面3で表示されている「brushed-metal」などが用意されている。

■GNOME（GTK＋）

「GNOME」のテーマと呼んでいるのは、実際にはツールキットである「GTK＋」のテーマである。ボタンやリスト、メニューやパネルに貼り付けられるビットマップなどをまとめて変更できる。コントロールセンター内の[デスクトップ]－[テーマセレクタ]を選ぶと、設定画面が表示される(**画面4**)。

こちらには、非常に多くのテーマが用意されている。しかし凝った背景のもの、たとえば**画面4**でサンプル表示されている「Cheese」などは楽しいのだが、字が読みづらいことが多くて、結局あまり使わなかったりする。

■もっとテーマを！

最初からインストールされているテーマを見尽くして、それでも「これは」というテーマが見つからなかったら、テーマ探しの旅にでかけよう。行き先は、Themes.org（**画面5**）。ウィンドウマネージャやツールキット、アプリケーションのテーマの集積所だ（http://www.themes.org/）。

「きっとプロだ！」と思わせるようなイカしたデザインから、「ジャパニメーションおたく発見！」という感じのあやしいものまで、揃っている。

そのほか非常にありがちなのが、「Linux上でも使い慣れた○○の外観を」（○○は任意のOSの名）というフェイク系だ。ただこのタイプは権利問題が生じやすく、つい先日リリースされた某社のOSをモチーフにした「水っぽい」テーマは、本稿作成時にはほぼ姿を消していた。やはり怒られたのだろうか。

上述のように、GTK＋とSawfishを組み合わせた環境では、両者のテーマが独立している。一人の作者が、同じ名前で両方のテーマを作っていることも多く、その場合は両方揃えることで、デザイン的により作者の意図に近いものが得られる。

なおページの都合で、WindowMakerやKDEについてふれることができなかったが、どちらにも「テーマ」を選択する機能がある。

画面3　Sawfishのテーマ設定画面
ウィンドウフレームの変更は、ウィンドウマネージャであるSawfishを利用する。

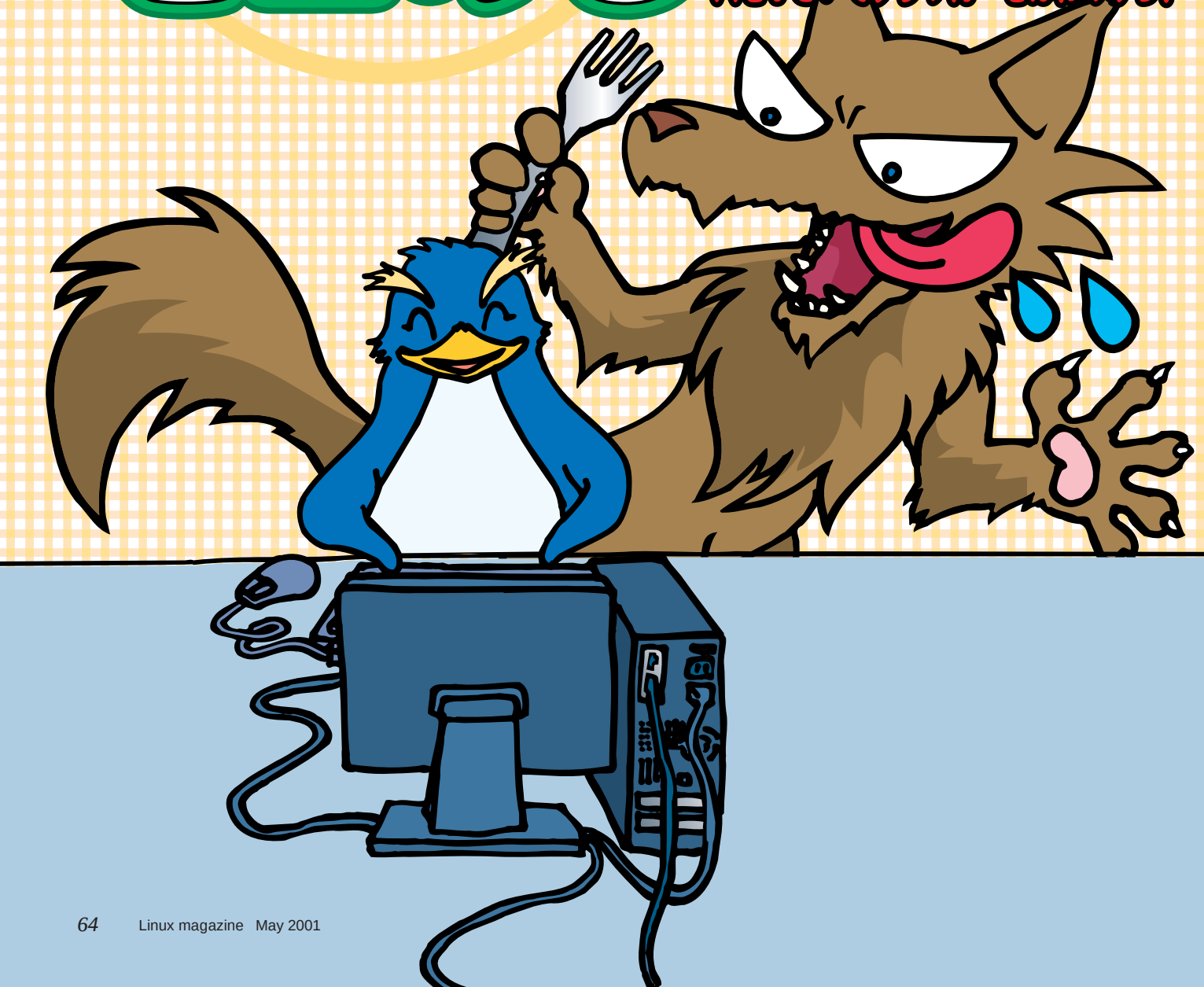
画面4　GNOMEのテーマ設定画面
ここでは、ウィンドウ内のさまざまな部品のデザインを変更する。

画面5　Themes.org
UNIX系OSのテーマ関係の探し物は、とりあえずここに来るのが正解。

ちょっと待て！　常時接続する前に……

ネットワークセキュリティを固める

インターネットにつないだ瞬間、
あなたのマシンはターゲットになる。
穴をふさいでクラッカーをはねのけろ！



ネットワークのセキュリティとは

文：編集部
Text : Linux magazine

インターネットは私たちの生活をより便利にしてくれた。WWWをはじめ、さまざまなサービスが使えるようになり、コンテンツも充実してくるにつれ、より高速でいつでも使える回線が欲しくなるのも当然のことだろう。フレッツ・ISDNやCATVによる常時接続環境を利用している人も増えてきた。さらに、21世紀を迎え、いよいよADSLによる高速で安価な常時接続環境が身近なものになってきている。

しかし、いいことばかりではない。常に回線がつながっていると、招かれざる客が訪れることも多くなる。それがクラッカーだ。

ネットワークセキュリティの必要性

クラッカーは、脆弱性を持つマシンを見つけ出し、弱いところを突いてマシンを乗っ取ったり、動作を停止させたりする。そのほかにも、目標となるマシンからデータを盗んだり、システムを破壊することさえあるのだ。

このような目に遭わないためには、インターネットに接続しているマシンから弱い部分をなくすか、インターネットから内部ネットワークを隠すしかない。現実的な方法は後者だろう。Linuxマシンをファイアウォールにして、内部のネットワークを守ることで、安全にインターネットを利用できるようになる。しかし、肝心のファイアウォールマシンが脆弱ではお話にならない。この特集では、いかにしてLinuxマシンから脆弱性を排除し、クラッカーの攻撃から身を守るかを考えることにしよう。

■どのような攻撃を受けるのか？

クラッカーの攻撃にはどのようなものがあるのだろうか。近年、被害が拡大しているのが「サービス不能攻撃」と呼ばれるものだ。DoS（Denial of Service）アタックともいうこの攻撃は、Webサービスなど、インターネットサービスを提供するサーバに対し、サーバのCPUパワー、メモリ、ディスクスペースや、ソケットなどのリソースを枯渇させることで、正常なサービスができない状態に陥れるものだ。手口としては、嘘の接続要求を大量に送りつけたり、サーバプログラムのバグを突くものが多い。

個人のLinuxユーザーにとって、DoSアタックよりも深刻なのは、マシンを乗っ取ることを目的としたクラッキングだ。クラッカーにroot権限を奪取されると、そのLinuxマシンはクラッカーの意のままにされてしまう。ファイルをのぞいたり、書き換えたりできるのはもちろん、こっそりとよからぬプログラムをインストールすることだってできる。

実際、クラッキングされると、rootkitと呼ばれるプログラム群やDDoSツール、バックドアを仕掛けられることが多い。rootkitには、login、ps、topなどのコマンドを改ざんしたものが含まれており、正しいプログラムを置き換えることで、クラッカーの実行しているプロセスを表示しない、ログに残さないなど、管理者が侵入されたことに気づかせないようにする。また、DDoS（Distributed Denial of Service）ツールは、設定された日時にあるサイトに対してDoSアタックをかけるツールだ。クラッカーは、多数のマシンにDDoSツールを仕込み、これらのマシンによって目標のサイトを同時多発で攻撃するのだ。DDoSツールを仕込まれ、攻撃の一端を担ってしまうと被害者であると同時に加害者にもなってしまう。バックドアとは、正規の認証を受けずにマシンに侵入するための仕組みで、これを作られると、いくら表門を閉じてもクラッカーは自由に出入りできるようになるのだ。

このほか、sendmailなどのメールサーバの設定を間違えると、広告など無差別に送られる迷惑なメール（SPAM）を送るために利用されたりする。また、プロキシサーバを外部から利用できるようにしておくと、Web掲示板への攻撃などで身元を隠すために使われることがある。

このような攻撃、不正利用をされないために、ネットワークのセキュリティを固めよう。

セキュリティポリシーの策定

セキュリティ対策の必要性はおわかりいただけただろう。では、どのような対策を講じるべきなのか？　セキュリティを高めるためにはコストがかかる。コストは金銭の問題にとどまらない。設定を行う知識や工数もコストの一部だ。また、セキュリティを高くすると、ネットワーク利用の自由度が低下することもある。安全性とコストを

秤にかけ、どのようなセキュリティ対策を行うかを決めなくてはならない。もちろん、ネットワークの構成によって、取るべき対策も変わってくる。自分の管理するサイトを、どのようなセキュリティポリシーによって運営するのか、よく考えよう。

■必要なサービスは何か

ポリシーを策定する際には、必要なサービスを決めなくてはならない。むやみにいろいろなサービスを提供しようとして、不必要なデーモンを動かすのは得策ではない。ネットワークサービスを提供するサーバプログラムには、しばしば脆弱性が発見されるからだ。多くの場合、この弱点を修正したバージョンがすぐに配布されるが、それに気づかずにいるとクラッカーの餌食にされてしまう。最近では、DNSサーバプログラムのbindにこのような不具合が発見され、すぐに修正版が用意された。しかし、多くのサイトがbindの更新を怠り、Webページを書き換えられたり、サーバを乗っ取られたりした。

Linuxマシンで提供したいサービスは何なのかをよく考え、動かすサービスを吟味しよう。たとえば、ルータ兼ファイアウォールとして利用するだけなら、WebやFTPのサーバを動かす必要はないわけだ。安全を第一に考えるならば、不必要なサーバプログラムはアンインストールしてしまおう。さらに、コンパイラなどの開発ツールもインストールしないのが望ましい。こうしておけば、万が一クラッカーに侵入されても危険なツールをインストールされにくくなる。とはいえ、実際にそこまですると使い勝手も悪くなる。特に、家庭で立てるサーバの場合、サービスごとに個別のマシンを用意するのは難しいので、1台のマシンでルータ、ファイアウォール、Webサーバなどを兼ねることも多いだろう。

■必要のないサービスは止める

多くのLinuxディストリビューションは、インストール直後からいろいろなネットワークサーバプログラムが動作している。たとえば、Vine Linux 2.1では、WebサーバのApache（httpd）や、プリントサーバ（lpd）などが起動されているが、これらのサービスが不要であれば止めておこう（前述の通り、アンインストールすればより安全だ）。

Linuxの起動時にどのサービスが起動されるかは、次のようにして調べる。

```
$ chkconfig --list
```

すると、**画面1**のような結果が出力される。デーモン名に続き、0〜6までの各ランレベルでそのデーモンが自動起動されるかどうかが表示される。ほとんどのディストリビューションでは、テキストログインの場合がランレベル3、グラフィカルログインではランレベル5を使うようになっている。

サービスの自動起動を設定する場合にもchkconfigコマンドを利用する。たとえば、httpdをランレベル3、4、5で自動起動しないようにするには、rootユーザーになって、

```
# chkconfig --level 345 httpd off
```

を実行すればよい。自動起動させる場合は、offをonに変える。また、ntsysvコマンドを使えば、テキストベースのGUIインターフェイスで設定を変更することも可能だ（**画面2**）。

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

■ネットワークの構成

ファイアウォールは、インターネット（外部ネットワーク）と内部ネット

```
netfs       0:off   1:off   2:off   3:on    4:on    5:on    6:off
httpd       0:off   1:off   2:off   3:on    4:on    5:on    6:off
autofs      0:off   1:off   2:off   3:on    4:on    5:on    6:off
named       0:off   1:off   2:off   3:off   4:off   5:off   6:off
dhcpd       0:off   1:off   2:off   3:off   4:off   5:off   6:off
gated       0:off   1:off   2:off   3:off   4:off   5:off   6:off
inet        0:off   1:off   2:off   3:on    4:on    5:on    6:off
innd        0:off   1:off   2:off   3:off   4:off   5:off   6:off
lpd         0:off   1:off   2:on    3:on    4:on    5:on    6:off
nfs         0:off   1:off   2:off   3:off   4:off   5:off   6:off
nfslock     0:off   1:off   2:off   3:on    4:on    5:on    6:off
sshd        0:off   1:off   2:on    3:on    4:on    5:on    6:off
identd      0:off   1:off   2:off   3:on    4:on    5:on    6:off
portmap     0:off   1:off   2:off   3:on    4:on    5:on    6:off
postfix     0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

画面1　chkconfig --listの出力結果（抜粋）

画面2　ntsysvコマンドの実行画面

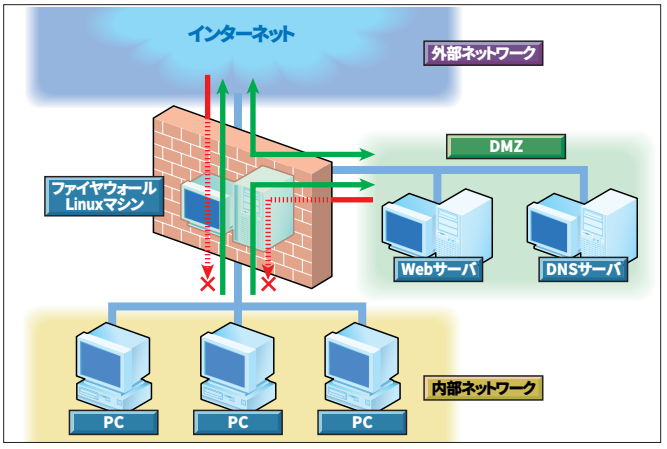


図1　DMZを置くネットワーク構成
外部からDMZへの通信を監視し、外部から内部、DMZから内部への接続を禁止してセキュリティを確保する。

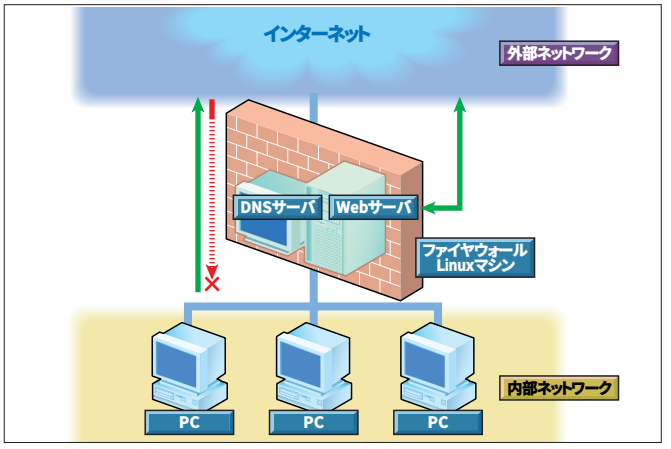


図2　Linuxマシンをファイアウォール兼サーバにする構成
安価に構成できるが、ファイアウォールマシンでサーバプログラムを動かすので、安全性が少々低下する。

ワークの間に設置され、ここを通るパケットの種類やアドレス情報、データの内容などをチェックする。そして、内部ネットワークへの不正アクセスなどを排除したり、アクセス制御を行う。

WebやDNSなど、インターネット向けのサービスを提供する場合は、外部ネットワーク、内部ネットワークのほかに、DMZ（DeMilitalized Zone：非武装地帯）ネットワークを作り、公開サーバはここに置くことが多い（**図1**）。サーバをDMZに置くことで、すべてのパケットがファイアウォールを通るようになるので、サーバを攻撃から守ったり、アクセスを制御することが可能になるのだ。DMZから内部ネットワークへの接続は禁止する。そうすれば、公開サーバがクラックされても内部ネットワークへの侵入を阻止することができる。

DMZを作るには、それなりのコストがかかるため、個人でサーバを運用する場合は、ファイアウォールと公開サーバを兼ねさせることが多いだろう。この場合、安全性は若干低下するが、あとで解説するパケットフィルタなどによってアクセス制限をすることで十分な安全性を確保できる（**図2**）。

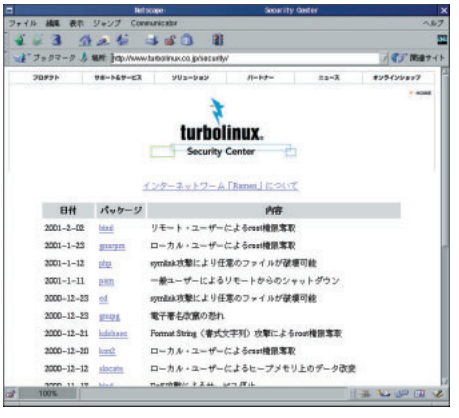
セキュリティフィックス

ここ数年、「バッファオーバーフロー」によるクラッキング被害が増えている。これはサーバプログラムが用意しているバッファに収まりきらないような、長い文字列を送ることで、本来実行できないはずのプログラムをサーバの実行権限によって動かすという手口だ。通常ではあり得ない状況を作り出し、サーバプログラムを誤動作させる方法はクラッカーがよく使う手だ。サーバプログラムのセキュリティフィックス情報をこまめにチェックし、修正版が発表されたらすぐにバージョンアップすることで、これらの攻撃に対処できる。

■アップデート情報をチェック

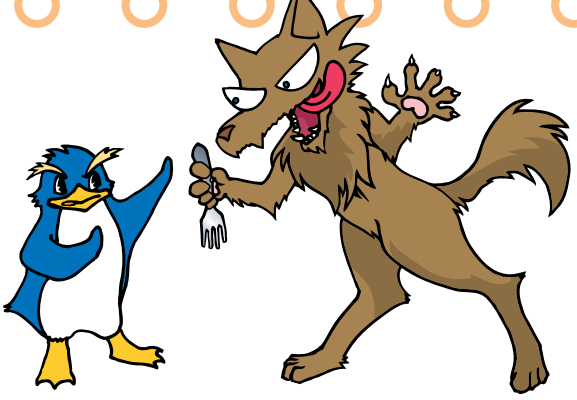
各ディストリビューターは、Webを通じてアップデート情報を配布している。最近では、セキュリティに関するアップデートはかなり迅速に行われているので、できるだけ頻繁にチェックするようにしたい。Red Hat Linuxに関しては、英語版のアップデートパッケージのほうが早く公開されるので、こちらもあわせてチェックするとよいだろう。

画面3　Vine Linuxのアップデートページ



画面4　Turbolinuxのセキュリティページ

ネットワークのセキュリティとは



文：編集部
Text : Linux magazine

インターネットに接続するマシンは、常に攻撃を受ける可能性がある。この攻撃から身を守るにはどうすればよいのだろうか。Linuxでは、外部からのアクセスを制限したり、怪しげなデータパケットをフィルタする仕組みを利用することができる。これを活用することで、ネットワークのセキュリティを向上させることができるのだ。

TCP/IPでの通信

インターネットでは、TCP/IPという通信プロトコルを使ってデータのやりとりを行う。セキュリティを高めるアクセス制限や、パケットフィルタの設定を行うためには、TCP/IPの基礎知識が必要だ。

ここでは、ごく簡単にTCP/IPの仕組みを説明しよう。

■パケットによる通信

TCP/IPは、さまざまなプロトコルの集合体であり、データを送受信する場合は、TCP、あるいはUDPというプロトコルが利用される。このほかにも、通信を制御するICMPなど、いろいろなプロトコルがある。

TCPやUDPでは、データを細切れのパケット（小包）にしてやりとりする。細切れにして送信されたデータは、受信側で再び組み立てられ、元のデータになるのだ。このパケットは、細切れにしたデータのほかに、宛先や送信元などが書かれたヘッダから成り立っている。

TCPが通信相手とコネクションを確立し、エラーで届かないパケットがあると再送信するなどの処理を行うのに対し、UDPではそのような処理を行わない。その分、UDPのほうが処理内容が軽くなっている。

■IPアドレスとポート番号

TCP、およびUDPパケットのヘッダには、送信元のIPアドレスとポート番号、宛先のIPアドレスとポート番号が書かれている。パケットは、宛先のIPアドレスに基づいて相手に届けられ、宛先のポート番号ごとに、WebやSSHなどのサーバプログラムに届けられる。これに対する返信は、送られてきたパケットの送信元IPアドレスとポート番号に対して送られる。

サーバプログラムは、特定のポート番号へのアクセスを待っていて、接続されるとそれに対してデータを送信する。待機するポート番号は、telnetは23番、Webなら80番というように、よく使われるサービスではだいたい決まっている。特に、0～1023番までのポートは、Well-knownポート番号と呼ばれ、インターネット全体で統一的に定められている。実際にLinuxシステムで使われるポート番号とTCP／UDPのプロトコル種別は、/etc/servicesというファイルで指定されている。

また、UNIX系のOSでは、スーパーユーザー（root）のプロセスやプログラムしかWell-knownポートを利用できない。

たとえば、ブラウザでWebサーバに接続する場合は、ブラウザを動かしているマシンの1024番以上のポートからWebサーバの80番ポートにつないでデータを送受信する。クライアント側のポートは、空いているポートの中からOSが選択して割り当てる。

アクセス制限の方法

TCP/IPの通信では、通信相手をIPアドレスで特定できる。また、サービスの種類は、サーバ側のポート番号で特定可能だ。すなわち、相手のIPアドレスや、接続先のポート番号によってアクセスを制限することが可能なのだ。

アクセスを制限する方法は、大きく分けると2つある。ひとつは、接続元のIPアドレスによって、サーバプログラムがサービスをするかどうかを判断する方法だ。これは、各サーバプログラムが個別に行うか、TCP_Wrappers、xinetdなどのスーパーサーバプログラムによって行う。スーパーサーバとは、サービスを提供するさまざまなサーバプログラムの代わりに接続要求を待つプログラムで、クライアントからの要求があると、対応するサーバプログラムを起動するプログラムだ。

もうひとつの方法は、パケットヘッダを分析して、そのパケットをプログラムに渡すかどうかを判断する方法だ。これをパケットフィルタという。

パケットフィルタを使うと、管理者が設定した内容に則って、不要なパケットを破棄することができる。破棄されるパケットは、サーバプログラムに渡されないので、安全性も増す。また、パケットの種類によるフィルタも可能だ。ただ、少々設定が難しいことと、

FTPのように複数のコネクションを張るプロトコルや、使用するポート番号が不定のプロトコルに対応するのが難しいという欠点もある。

■ルータによるパケットフィルタ

ISDNダイヤルアップルータや、ブロードバンドルータを使ってインターネットに接続している人も増えてきた。ほとんどのルータは、パケットフィルタの機能を持っているので、ここで不要なパケットを制限するように設定しよう。ただし、これらのルータは価格を抑えるために、フィルタリングルールを設定できる数が少ないなどの制限もある。その場合は、Linuxマシンをファイアウォールとしてルータと内部ネットワークの間に設置しよう。

スーパーサーバで接続を制限する

ほとんどのLinuxディストリビューションには、TCP_Wrappersというパッケージが含まれている。これは、アクセスコントロール機構を持つスーパーサーバだ。inetdが外部からのアクセスを待ち、接続要求があるとtcpdにそのリクエストを渡す。tcpdは、リクエストをログに記録するとともに、IPアドレスやホスト名をチェックし、問題がなければ適切なサーバプログラムを起動する。

Red Hat Linux 7では、inetdの代わりにxinetdというデーモンがインストールされる。これは、サーバごとの設定をinetdよりも細かく行えるスーパーサーバプログラムだ。

■TCP Wrappersの設定

inetd経由で起動するサーバプログラムは、/etc/inetd.confに登録する。初期状態で、主なサーバプログラムが登録されているが、そのほとんどは行のはじめにコメントマークの“＃”が付けられ、無効になっている。必要なものだけ“＃”を外して使えるようにしよう。

inetdを介して起動されるサーバプログラムのアクセス制限は、/etc/hosts.allow、/etc/hosts.denyという2つのファイルで設定する。接続を許可するサービスについてはhosts.allowに、禁止するものについてはhosta.denyに、それぞれ設定する。これらのファイルは、hosts.allow→hosts.denyの順に設定が有効になるので、hosts.denyには、

```
ALL：ALL
```

とだけ書いて、すべてのサービスへのアクセスを禁止しておき、有効にするサービスとIPアドレスなどをhosts.allowに追加するのがよいだろう。

リスト1のようにすると、FTPサーバへの接続は、pc01.ascii-linux.comというホストにだけ許可し、POP3サーバへはドメイン名がascii-linux.comのホストにだけ許可する。また、SSHサーバへの接続は、IPアドレスが192.168.0.1～192.168.1.254のホストにのみ許すことになる。

ここで、再び/etc/inetd.confを見ると、SSHサーバプログラムのsshdがどこにもないことに気づくだろう。通常、sshdはinetdを経由せずに利用するので、inetd.confには書かれていないのだ。しかし、sshdをコンパイルする際のconfigureスクリプトで、“--with-tcp-wrappers”というオプションを指定することで、TCP_Wrappersによるアクセス制限機能を利用できるようになっている（**76ページのコラム**参照）。

▼リスト1　/etc/hosts.allowの例

```
in.ftpd: pc01.ascii-linux.com
pop-3:    .ascii-linux.com
sshd: 192.168.0.0/255.255.255.0
```

■xinetdの設定

Red Hat Linux 7では、/etc/xinetd.confと、/etc/xinetd.dディレクトリ以下のファイルでxinetdを設定する。xinetd.confには、すべてのサーバプログラムで共通の設定を書き、xinetd.d以下のファイルにはサーバプログラムごとの設定を書く。

例として、POP3サーバに関するxinetdの設定をする/etc/inetd.d/ipop3を見てみよう（**リスト2**）。“disable＝no”でこのサービスを有効にしている。“disable＝yes”だとPOP3サービスが無効になる。また、“only_from＝192.168.0.0/24”で、接続可能なホストのIPアドレスを192.168.0.1～192.168.0.254に限定している。さらに、“access_times＝09:00-17:00”で接続可能な時刻を午前9時から午後5時の間に制限している。

このほか、only_fromの代わりにno_accessを使い、接続を禁止するホストを指定することも可能だ。これ以外の行の意味については、man xinetd.confを実行して、マニュアルを参照してほしい。

▼リスト2　/etc/xinetd.d/ipop3の例

```
service pop3
{
  disable = no
  socket_type     = stream
  wait            = no
  user            = root
  server          =
/usr/sbin/ipop3d
  log_on_success += USERID
  log_on_failure += USERID
  only_from     = 192.168.0.0/24
  access_times    09:00-17:00
}
```

Linuxのパケットフィルタ設定

Linuxカーネル2.2では、ipchainsというコマンドでパケットフィルタを設定できる。カーネル2.4では、パケットフィルタの機能が拡張されており、iptablesというコマンドを使って設定を行う。

どちらも、よく似た書式のコマンド体系になっており、チェーンと呼ばれるフィルタごとにパケットの処理を定めるルールを適用してパケットの制御を行う。初期状態では、入力パケットが入るinput、出力パケットが入るoutput、転送パケットが入るforwardというチェーンが用意されている(iptablesでは、INPUT、OUTPUT、FORWARDと大文字で書く)。

自分で定義したチェーンを新たに作ることも可能で、チェーンを鎖のようにつなぐことで複雑なフィルタルールを作ることも可能だ。

ここでは、Linuxマシンをサーバ兼ファイアウォールにする例を考える。このマシンには、インターネットにつながったppp0というネットワークインターフェイスと、内部ネットワークにつながるeth0というネットワークインターフェイスがあると仮定し、内部ネットワークは、192.168.1.0/24というIPアドレスを使っているとしよう。

設定のポリシーは次のようにする。

- **内部ネットワークからLinuxマシンへのパケットは受け入れる**
- **インターネットからLinuxマシンへのパケットは、公開するサーバサービス宛のものは受け入れる**
- **Linuxマシンからインターネットへつなぐために必要な、返信パケットは受け入れる**
- **内部ネットワークからIPマスカレードを使いインターネットに接続する**

■ipchainsによる設定

上記のポリシーにしたがい、**リスト3**のように設定した。このリストに沿って解説するので、**表1〜3**を見ながら読んでほしい。

1〜3行目は各チェーンに対するデフォルトポリシーの設定だ。これ以下で設定する条件に当てはまらないパケットはこのポリシー通りに処理される。ここでは、inputチェーンとforwardチェーンは破棄、outputチェーンは受け入れることにしている。すなわち、input、forwardは、受け入れる条件を設定しない限りすべてのパケットが破棄される。

5行目は、inputに対し、ローカルループバックインターフェイスからのパケットはすべて受け入れる設定だ。6行目は、送信元IPアドレスが内部ネットワークのもので、インターフェイスeth0から入ってくるパケットを受け入れる設定だ。これで、ローカルループバックと、内部ネットワークからLinuxマシンへのパケットがすべて受け入れられるようになった。

8行目は、Linuxマシンからインターネットのホストへ接続したときに、相手のホストからのパケットを受け取るための設定だ。TCPによる通信では、データの送受をする前にコネクションを確立する必要がある。そのためには、接続先に対して最初にSYNパケットという接続要求パケットが送られ、3ウェイハンドシェイクと呼ばれる手続きを行う。8行目では、インターネット側のインターフェイスから入ってくるパケットで、SYNパケット以外(! --syn)で、かつ宛先ポートは1024番以上(--dport 1024:)のものだけ受け入れるようにしている。SYNパケットは破棄されるので、インターネットからの接続要求は無視される。さらに、Webブラウザなど、多くのクライアントプログラムは、1024番以上のポートを使うので、このように設定すれば問題なく利用できる。例外はFTPとSSHだ。FTPは、サーバからクライアントにもコネクションを張るので、この設定ではパッシブモードでしか利用できない。また、SSHクライアントはデフォルトで1023番以下の特権ポートを使う。sshコマンドに-Pオプションを付けて、1024番以上のポートを使うようにすれば問題ない。

一方、UDPではコネクションを確立しない。9行目は、UDPのための設定だ。1024番以上のポートを開けている。

11行目で、ICMPパケットを受け入れるようにしている。

13行目では、内部ネットワークのIP

-P	デフォルトポリシーを変更する
-A	チェーンの最後にルールを追加する
-I	チェーンの最初にルールを追加する
-D	チェーンから指定したルールを削除する
-N	新しいチェーンを作る
-F	チェーンからすべてのルールを削除する
-X	空のチェーンを削除する
-L	チェーンに設定されたルールを表示する

表1　ipchains、iptablesの主なコマンド

-s	送信元のIPアドレス
-d	宛先のIPアドレス
-p	プロトコル(tcp、udp、icmp等)
--sport	送信元のポート番号
--dport	宛先のポート番号
-i	インターフェイス(eth0、ppp0など)
-j	ターゲット(ACCEPT、DENY、REJECT、MASQ、別のチェーンなど)

表2　ipchainsの主なオプション

▼リスト3　ipchainsでの設定例

```
 1: ipchains -P input   DENY
 2: ipchains -P output  ACCEPT
 3: ipchains -P forward DENY
 4:
 5: ipchains -A input -i lo -j ACCEPT
 6: ipchains -A input -i eth0 -s 192.168.1.0/24 -j ACCEPT
 7:
 8: ipchains -A input -i ppp0 -p tcp ! --syn --dport 1024: -j ACCEPT
 9: ipchains -A input -i ppp0 -p udp --dport 1024: -j ACCEPT
10:
11: ipchains -A input -p icmp -j ACCEPT
12:
13: ipchains -I input -i ppp0 -s 192.168.1.0/24 -j DENY
14: ipchains -A input -i ppp0 -p tcp --sport 1024: --dport 80 -j ACCEPT
15:
16: ipchains -A forward -i ppp0 -s 192.168.1.0/24 -j MASQ
17: ipchains -A forward -j REJECT
18: echo 1 > /proc/sys/net/ipv4/ip_forward
```

▼リスト4　iptablesではすでに接続されているものを許可できる

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

アドレスを詐称してインターネットから侵入しようとするパケットを破棄している。

14行目は、インターネットからこのLinuxマシンで動いているWebサーバ（ポート番号80）へ接続を許可するための設定だ。

16行目以降は、IPマスカレードを有効にする設定だ。

設定されたルールを表示するには、ipchains -L -vを実行すればよい。

■iptablesによる設定

iptablesでは、ipchainsとは違い、入力／出力インターフェイスをオプションで個別に指定する（**表4**）。また、パケットを破棄するためのターゲットはDENYではなく、DROPになっているのも大きな違いだ。

さらに、**リスト4**のようにすることで、コネクションが確立している通信に関するパケットを受け入れられるので、SSHの問題も起きないし、フィルタルールの設定を簡略化できる。

アクセス制限の難しさ

Linuxにはスーパーサーバ、パケットフィルタによるアクセス制限の仕組みが用意されている（**図1**）。さらに、サーバプログラム自身でアクセス制限

ipchains	iptables	意味
ACCEPT	ACCEPT	受け入れる
DENY	DROP	破棄する
REJECT	REJECT	拒否する

表3　ipchains、iptablesで指定する主なターゲット

-s	送信元のIPアドレス
-d	宛先のIPアドレス
-p	プロトコル（tcp、udp、icmp等）
--sport	送信元のポート番号
--dport	宛先のポート番号
-i	入力インターフェイス
-o	出力インターフェイス
-j	ターゲットの指定
-t	テーブルの指定

表4　iptablesの主なオプション

ができるものも少なくない。ネットワーク環境やポリシーが千差万別である以上、これらを適切に設定するのはたやすくない。ネットワークに関する知識も必要だ。しかし、くじけることなく、最適な設定を目指してがんばってほしい。

今回は、ipchainsを中心に、簡単なルール設定について解説したが、インターネット側から送られてくるはずのない、プライベートアドレスをもつパケットなども破棄するとよいだろう。付録CD-ROM Disc 2のSpecialディレクトリにipchains、iptablesによるパケットフィルタの設定例を収録しているので、参考にしてほしい。

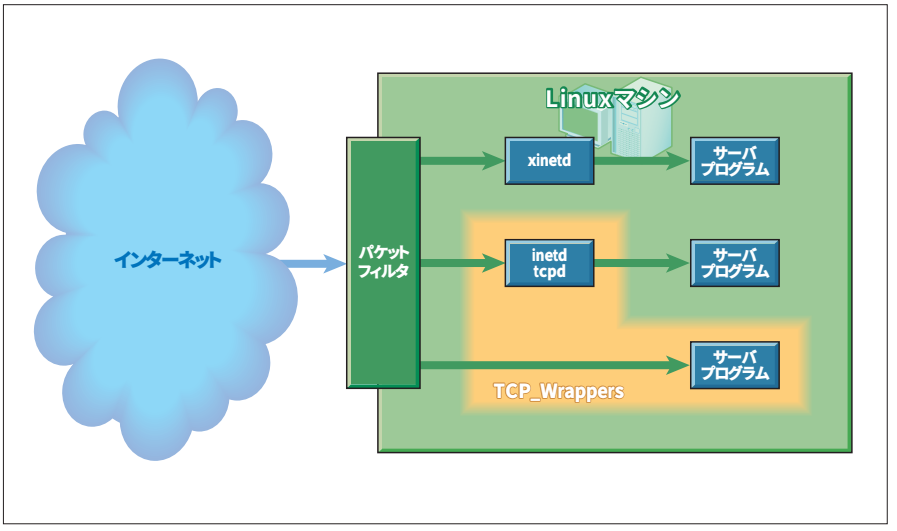
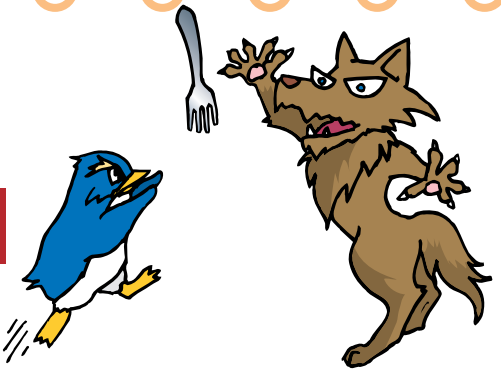


図1　外部からのアクセスを制限する

セキュアシェル（SSH）



文：安田幸弘
Text : Yukihiro Yasuda

インターネットでは、通信の内容が平文で送られるため、パスワードのような重要な情報は、そのまま送信してはならないというのは、今では常識だ。そこで、最近telnetやFTPの代わりとしてよく使われるようになったツールがSSHである。

SSHは、BSDのr＊コマンドの使い勝手をそのままに、セキュリティ機能を追加したツールだが、ネットワークにパスワードを流すことなく、リモートマシンへのログインやファイルの転送をすることができるほか、公開鍵暗号による自動ログインや通信経路の暗号化と圧縮、ポートのトンネリングなど、非常に便利な数々の機能を備えている。数年前にライセンスの制限が少ないOpenSSHがリリースされて以来、＊BSDのコミュニティでは、SSHを標準コマンド化する動きが広がっており、Linuxでも、最新のVineやRed Hatでは標準配布のオプションになったことなどにより、急速にユーザーが増えている。

もちろん、SSHは単なるツールでしかない。SSHは決して万能のセキュリティツールではないが、インターネットの最も弱い部分をがっちりとガードしてくれる便利で頼もしいツールであることは事実。インターネット時代のシステム管理者として、ぜひSSHは使えるようになっておきたい。

いろいろな“SSH”

＊BSDやLinuxではすっかり有名になったSSHだが、ひとくちにSSHといってもバージョンの違いや開発元の違いなどによっていくつかの種類がある。また、最近は名称についても商標に関する議論もあり、やや錯綜しているが、ここで簡単にさまざまなSSHについてまとめておこう。

■プロトコルバージョン

SSHの違いのうち、プロトコルのバージョンの違いは重要だ。SSHのプロトコルには、大別してバージョン1とバージョン2の区別があり、両者には全く互換性がない。また、バージョン1には初期のSSHに採用されていたバージョン1.0、1.3などがあるが、現在では1.0は全く使われておらず、また1.3 プロトコルも事実上、古いシステムとの互換性の維持という以上の意味はないと考えてよい。

プロトコルバージョン1、2の大きな違いは、バージョン1では公開鍵の交換アルゴリズムにRSAを採用しているのに対して、バージョン1ではDSA/DHを採用していることだ。現在、RSAはパブリックドメインになっているが、2000年9月まではRSAの特許が有効だったため、SSH社はRSAの特許問題を避けるために、DSA/DHを公開鍵アルゴリズムに使用するプロトコルバージョン2を開発したわけだ。

また、プロトコルバージョン2は公開鍵アルゴリズムの変更以外にも、バージョン1.5では安全性に関する議論のあったメッセージ改竄を防止するためのコードをハッシュ関数を使ったメッセージ認証アルゴリズムに変更するなど、安全性が高められている。ただし、コーディングを改善することで、バージョン1の問題点は避けられるといわれており、バージョン1が危険というわけでもないようだ。今後、SSHは次第にバージョン2に移行していくことになるのだろうが、バージョン1と比べて、バージョン2は動作が若干重いこと、すべてのプロダクトがバージョン2をサポートしているわけではないことなどの理由で、現在でもバージョン1系のプロダクトは現役だ。

なお、SSH社はそれぞれのプロトコルをSSH1とSSH2の別製品とし、SSH2のサーバにバージョン1で接続された時には、SSH1のサーバを起動する。それに対してOpenSSHではSSH1とSSH2の両方のコードが組み込まれた単一の実行ファイルで両者を自動判別している。

■各種のプロダクト

SSHには、SSH社が開発する「元祖」SSH1、SSH2のほかに、オープンソースコミュニティを含むサードパーティから、各種のプロダクトがリリースされている。

同じバージョンのプロトコルおよび暗号アルゴリズムをサポートするプロダクトは、原則として相互に接続が可能だが、プロダクトごとの独自の機能に互換性のない部分もあるので注意したい。現在、広く使われているSSH1／SSH2とOpenSSHは、暗号化アルゴリズムさえ同じであれば、ほぼ問題なく相互接続ができるようだ。

主なプロダクトの特徴は以下の通りである。

●SSH1／SSH2

SSH社により開発される商用版SSHだが、SSH1／SSH2ともにプライベートな利用や非営利の公益団体での利用には無償で公開される、最も標準的な実装とされるプロダクト。

OpenSSHなどの互換プロダクトとの大きな違いは、X.509やPGPなどの公開鍵サーバを利用できる機能を持つことで、大規模なサイトでの面倒な公開鍵の管理が容易になるほか、SOCKS4クライアント機能を内蔵している。OpenSSHと比べると、よりビジネスユースを意識した仕様を持つ。

このほかに、SSH1／SSH2には特許が設定されている暗号アルゴリズムも組み込まれているため、フリーの実装よりも暗号の選択幅が広い。UNIX版のほかに、Windows版やMacintosh版もリリースされている。

●F-Secure SSH

SSH Communications Security社からOEMされている商用SSH。基本的にSSH社の製品と同等と考えてよい。

●OpenSSH

ウロネンによる初期のフリーバージョンを元に、OpenBSDプロジェクトによって特許などに関連するコードを除去、各種の機能を追加したフリーのプロダクト。BSDライセンスで配布される。OpenBSDをはじめ、NetBSD、FreeBSDなど、BSD系のオープンソースOSでは、標準コマンドとして装備されており、ユーザーも多い。

SSH1／SSH2が統合され、どちらのプロトコルにもひとつのバイナリで対応できることなどが特徴。また、初期のバージョンではサポートされていなかったプロトコル2への対応や、scp、sftpコマンドの添付、PAMへの対応など、積極的な開発が続けられている。さらに、セキュリティホールの報告に対する反応が早い点も大きな魅力のひとつといえそうだ。

なお、OpenSSHを利用する場合は、暗号ライブラリとしてOpenSSLが必要になる。

●lsh

GNUライセンスで配布されるSSHプロトコルバージョン2互換のセキュアシェル。鍵交換アルゴリズムにSRPプロトコルを利用できる。

現在のバージョンでは、プロトコルバージョン1、scp、鍵リングなどの機能はサポートしないなど、機能面ではやや限定される。

●MindTerm

JavaによるSSHプロトコルバージョン1／2の実装で、ファイル転送やポートフォワードなども利用できる本格的なSSH対応ターミナルである。MindTerm v1.21まではGPLで配布されていたが、現在開発中のバージョン2ではライセンスが変更され、SSH1／2と同様にプライベート、非営利での利用は無料、商用での利用は有料となる予定だ。

Pure Javaにより作成されているため、スタンドアローンでの利用に加え、Webブラウザからアプレットとして呼び出すことができるほか、プラットフォームを問わずに実行することができる。

SSHと暗号

SSHは、いくつかの暗号技術を組み合せて安全なセッションを実現している。SSHによる接続の手順をごく簡単に説明すると、まず、SSHはホストの公開鍵（非対称鍵）暗号を使ってクライアントからサーバホストを認証する。ここで使われる公開鍵暗号が、RSA（v1.x）やDSA（v2）で、互いに相手の公開鍵と秘密鍵を組み合わせることで、パスワードや暗号鍵をネットワークに流すことなく、接続先を認証することができる。また、ホスト認証の時

Column

“SSH”の商標を巡る議論

“SSH”の名称およびロゴは、SSHの原作者ウロネンにより設立され、商用版SSHの開発にあたっているSSH Communications Security社（以下SSH社）が1996年に出願し、1998年に米国で認められた登録商標である。

しばらく前までは、SSHといえば、本家SSH社がプライベートな用途に限り無償で配布していたSSH1またはSSH2を意味していた。だが、昨年からOpenBSDのグループが開発・配布するSSH互換のOpenSSHのユーザーが急速に増加し、利用者の誤認が無視できなくなったことを理由として、SSH社はOpenSSHに対して、名称の変更を求めている。また、SSH社は“Secure Shell”についても商標を登録しているが、広く使われている名称であるとして“Secure Shell”や、コマンド名としての“ssh”などの文字列に関する権利を放棄している。このほか、利用者が比較的少なく、誤認の可能性が少ないTTSSHなどの小規模なプロダクトに対しては、黙認する姿勢をとっている。

なお、本稿では慣例にしたがい、SSHプロトコル互換のすべてのプロダクト群を一括して“SSH”と表記し、特にプロダクトの区別が必要な場合はSSH（SSH社のプロダクト）、OpenSSH（OpenBSDのプロダクト）などと表記する。

に、セッションを暗号化する対称暗号用の鍵（共通鍵）を生成し、公開鍵暗号でホストに送り、以後のセッションはすべてこの鍵を使って暗号化される。

ユーザー認証はホスト認証のあとに行われるが、SSHはユーザーの公開鍵を使った公開鍵暗号によるユーザー認証のほか、rhosts認証、パスワード認証などの認証手段を選ぶこともできる。この段階では、すでにセッションが暗号化されているため、パスワード認証でも平文のパスワードがネットワークを流れることはない。そしてユーザーが認証されると、セッションが確立し、以後、安全な通信が可能になる。

なお、ホストやユーザーの認証に公開鍵暗号を利用し、通信内容の暗号化に対称鍵暗号を使う理由は、公開鍵暗号よりも対称鍵暗号のほうが計算が少なくて済み、マシンの負担を減らすことができるからである。

また、対称鍵暗号システムとして、IETFのSSHプロトコルのドラフトでは、3DESを必須のアルゴリズムとし、推奨されるアルゴリズムにBlowfish、Twofish128、AES128を規定している。このほかに、オプションのアルゴリズムとして、IDEA、CAST、ArcFour（RC4）、Serpent、そして鍵長が192ビットまたは256ビットのTwofish、AESなど（すべてCBCモード）が定義されている。

もちろん、暗号アルゴリズムが異なると通信ができないため、異なるプロダクト間での通信には、プロトコルのバージョンだけでなく、対称鍵暗号の種類を同じものにしておく必要がある。主なSSHプロダクトとプロトコルのバージョン、および利用できる対称鍵暗号のアルゴリズムを**表1**にまとめておくので参考にしていただきたい。

OpenSSHを使う

OpenSSHは、プロトコル1と2を装備したSSH1／SSH2互換のリモートログインツールだ。インストール、設定についてはこの後で説明するが、まず基本的な使い方を説明しよう。ここで説明する使い方以外にも、さまざまなオプションにより多彩な使い方ができるので、SSHに慣れてきたらいろいろな使い方を試してみていただきたい。

■リモートホストにログインする

“ ssh ホスト名 ”、あるいは“ slogin ホスト名 ”でリモートホストにログインすることができる。telnetと違って、通信の内容は暗号化されているため、パケットの盗聴による情報漏洩の心配がない。また、ssh-agentを使うことで、一度パスフレーズを入力すれば、いちいちログインのたびにパスワードやパスフレーズを入力する必要がない。

■リモートホストのコマンドを実行する

“ ssh ホスト名 コマンド名 ”で、リモートホストのコマンドを実行できる。たとえば、“ ssh example.com w ”は、example.comのwコマンドを実行し、結果を表示する。X11の転送オプションをオンにすると、リモートマシン上のX11クライアントのウィンドウをローカルマシンで表示、実行することもできる。

■ファイルを転送する

“ scp ファイル名 ホスト名： ”で、ローカルのファイルをリモートマシンに転送することができる。また、リモートのファイルをローカルに転送する場合は、“ scp ホスト名：ファイル名 ディレクトリ ”を実行する。

もちろん、複数のファイル名を指定したり、“ -r ”オプションでディレクトリごと転送することもできる。

■インタラクティブなファイルの転送

“ sftp ホスト名 ”で、FTPと似たインターフェイスのファイル転送ができる。もちろん、FTPと違ってパスワードや転送の内容は暗号化されるので、盗聴の心配のない安全なファイル転送が可能だ。

■ポートの転送

ポート転送機能により、通信内容が保護されないプロトコルでも、ローカルとリモートのマシンの間で安全な通信を可能にする。

たとえば、“ ssh -2 pop.example.com -N -L 1111：localhost：110 & ”は、リモートマシン（pop.example.com）のポート110（POP3）をlocalhostのポート1111に転送する。

プロダクト	Ver.	利用できるアルゴリズム							
openssh	v1	3des	blowfish						
	v2	3des	blowfish	RC4	cast				aes
ossh	v1	3des	blowfish			idea	des		
ssh1	v1	3des	blowfish	RC4		idea	des		
ssh2	v2	3des	blowfish	RC4	cast		des	twofish	
f-secure ssh2	v2	3des	blowfish			idea			
lsh	v2	3des	blowfish	RC4		(idea)			
TTSSH	v1	3des	blowfish	(RC4)		(idea)	des		
MindTerm	v1	3des	blowfish	RC4		idea	des		
	v2	3des	blowfish	RC4	cast	idea	des	twofish	aes

表1　主なSSHプロダクトとプロトコルバージョン、利用可能な暗号アルゴリズム
注：プロダクトのバージョンやコンパイル時のオプションなどで利用できる暗号が異なる場合がある。

インストール

OpenSSHの最新版は2001年3月の段階で2.5.2p1となっている。Linuxの場合、Red Hat7.0ならオプションでOpenSSH 2.1（updatesには2.2がある）が選べるが、OpenSSHは進歩が早いので、OpenSSHのWebページ（http://www.openssh.com/）からリンクをたどって、最寄りのサイトから最新版のRPMを入手することをお勧めしたい。ほとんどの場合、RPMをインストールするだけでSSHが利用できるようになる（**表2**）。

なお、askpassユーティリティは、GNOME環境ならopenssh-askpass-gnome、そうでなければopenssh-askpassをインストールする。

自分でコンパイルする場合は、“ tar xvzf openssh-??.tar.gz ”でファイルを展開して、“ sh configure; make; make install ”すればよい。起動と終了は、通常のRPMのモジュールを使う場合と同様である。

なお、SSHを使えば、セキュリティの甘いrshやrloginは不要になるので、r＊系のサービスは停止するべきだろう。Red Hatなどのディストリビューションでは、/etc/inetd.conf に**リスト1**のような行があるが、これらの行はコメントアウトしておこう。

■ホスト鍵を作る

SSHでは、ホスト認証用の公開鍵ペアが、SSHのセッションを開始するために不可欠だ。インストールが終わったら、まずホスト認証用の公開鍵ペアを作る必要がある。

通常、ホスト認証用の鍵はSSHのインストール後、自動的に作成されるはずだ。/etc/ssh/以下のssh_host_key、ssh_host_key.pubがRSA、ssh_host_dsa_key、ssh_host_dsa_key.pubがDSAのホスト認証用の鍵だ。自動的にこれらの鍵が作成されない場合は、**画面1**のようにしてRSA、DSAのホスト鍵を作る。

これでssh_host_key.pubとssh_host_dsa_key.pubの2つの公開鍵ファイルと、ssh_host_keyとssh_host_dsa_keyの2つの秘密鍵のファイルができる。公開鍵はパーミッションが644で、誰にでも読み出せるが、秘密鍵はパーミッションを600として、root以外には読み出せないようにする。ほかのユーザーが読み出せる設定だと、sshdは接続を受け付けない。

■個人鍵を作る

SSHには、各種のユーザー認証の方法が用意されているが、SSHの特徴を生かした認証方法が、ホスト認証と同等の公開鍵による認証だ。

公開鍵の認証に使う個人鍵は、ssh-keygenコマンドを使って作る。SSH1互換で使うのであればRSA鍵、SSH2互換で使うのであればDSA鍵を作るが、どちらにも対応できるように両方の鍵を作っておくといい。RSA鍵を作るには、“ ssh-keygen ”をオプション

openssh-2.5.2p1-1.i386.rpm
openssh-askpass-2.5.2p1-1.i386.rpm
openssh-askpass-gnome-2.5.2p1-1.i386.rpm
openssh-clients-2.5.2p1-1.i386.rpm
openssh-server-2.5.2p1-1.i386.rpm

表2　OpenSSHのRPMファイル群

▼リスト1　rshとrloginを無効にする

```
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd
login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind
```

以下の行の先頭に“ # ”を挿入してコメントアウトする。

```
# /usr/bin/ssh-keygen -f /etc/ssh/ssh_host_key -N
# /usr/bin/ssh-keygen -d -f /etc/ssh/ssh_host_dsa_key -N
```

画面1　RSA、DSAのホスト鍵を作成する

```
RSA鍵の作成
$ ssh-keygen
Generating public/private rsa1 key pair.
Enter file in which to save the key (/home/user/.ssh/identity):
Enter passphrase (empty for no passphrase): <パスフレーズを入力>
Enter same passphrase again: <再度パスフレーズを入力>
Your identification has been saved in /home/user/.ssh/identity.
Your public key has been saved in /home/user/.ssh/identity.pub.
The key fingerprint is:
1b:fd:db:b7:75:64:f2:db:c6:79:35:62:31:90:1b:56 user@example.com

DSA鍵の作成
$ ssh-keygen -d
Generating public/private dsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_dsa):
Enter passphrase (empty for no passphrase): <パスフレーズを入力>
Enter same passphrase again: <再度パスフレーズを入力>
Your identification has been saved in /home/user/.ssh/id_dsa.
Your public key has been saved in /home/user/.ssh/id_dsa.pub.
The key fingerprint is:
9f:d2:fb:af:10:f7:26:d2:a5:1f:12:9c:9b:1b:5e:35 user@example.com
```

画面2　ユーザー鍵を作成する

なしで実行、DSA鍵は“ ssh-keygen -d ”を実行する（**画面2**）。

なお、秘密鍵は、パーミッションを600にして、自分以外のユーザーが読み出せないようにしておくこと。また、鍵を保護するためのパスフレーズは、十分に長く、推測が困難なものにする。秘密鍵は、この段階でパスフレーズによって暗号化されるが、パスフレーズが解読されてしまっては意味がない。目安としては、最低10文字以上でアルファベット、数字、記号類を混ぜて使ったパスフレーズで、辞書に載っている単語やローマ字、人名などは避ける、などの工夫が必要だ。

鍵を作ったら、さっそく今作った鍵で接続してみよう。“ ssh -1 localhost ”でSSH1プロトコルでログインすることができる。この場合、“ Warning: Permanently added 'localhost,127.0.0.1' (RSA1) to the list of known hosts. ”などの警告が出るが、これは接続先のホストの公開鍵リストに存在しないホストに接続しようとしている、という意味。DNSに細工をして、異なるホストに接続させ、トロイの木馬のようなプログラムでパスワードを盗もうとしても、この段階で気が付くわけだ。

さらに“ Enter passphrase for RSA key 'user@example.com': ”のようなプロンプトが表示され、先ほど作ったRSA鍵のパスフレーズの入力を行う。同様に、“ ssh -2 localhost ”を実行すると、DSA鍵のパスフレーズが要求される。

■ssh-agentとssh-addで快適ログイン

ssh-agentとssh-addは、RSA鍵やDSA鍵を記憶しておいて、パスフレーズを入力することなく接続ができるようにするコマンドだ。SSHの便利さは、ここにあるといっても過言ではない。

ssh-agentは、鍵を記憶するためのコマンド、ssh-addはssh-agentに鍵の追加や削除、一覧をするためのコマンドだ。基本的にはssh-agentから起動された子プロセスからSSHを使ってリモートログインすると、自動的にメモリーされた鍵を送出する。したがって、RSA／DSA鍵のパスフレーズが少々長いものでも、起動時に1回タイプするだけなので、ほとんど苦にならない。

一番簡単な使い方は、シェルで“ ssh-agent sh ”などのコマンドでさらにシェルを起動し、“ ssh-add ”コマンドで鍵を追加する。鍵を追加したら、リモートログインを試し、実際にパスフレーズなしでログインできることを確認してほしい。

しかし、毎回セッションを始めるときにssh-agentを実行するのは面倒なので、一般的にはウィンドウマネージャをssh-agentから起動する。startxを使ってウィンドウマネージャを起動している場合は、ssh-agent startxを実行すればいいが、xdmを使ったGUIログインの場合は、.xsessionなどの起動ファイルに“ ssh-agent .xinitrc ”、あるいはGNOMEなら“ ssh-agent gnome-session ”という行を書き加えるなど、それぞれのウィンドウマネージャをssh-agentの子プロセスとして起動させればよい。

さらに、ウィンドウマネージャの開始時に、ssh-addを実行させ、パスフレーズを覚えさせてしまうこともできる。この場合は、.xinitrcなどの起動時に実行されるrcファイルの適当な部分に“ ssh-add < /dev/null ”という行を書き込んでおく。これで、ウィンドウマネージャが起動するときに、パスフレーズ入力ウィンドウが開き、この段階でパスフレーズの入力を済ませてしまうことができる。なお、OpenSSHでは、x11-ssh-askpassとgnome-ssh-askpassのどちらかを選べるようになっていて、RPMのインストール時に適当な設定が行われるはずだが、うまく入力ウィンドウが開かない場合は、**リスト2**のようにssh-addの前に環境変数でパスフレーズ入力ツールのパスを指定するとよい。

Column

configureのオプションについて

バイナリで配布されているOpenSSHのSPECファイルを見ると、configureのオプションはリストのようになっている。

ほとんどの場合、これで問題はないが、--with-rshオプションはあまり意味がないので削除してもかまわない。その場合、代わりに--disable-suid-sshを指定することもできる。sshがsuid rootされている理由は、rshで特権ポートを利用するためだが、rshを使わない場合にはsuid rootは不要だからだ。また、suid rootを外すことで、runsocksコマンドによるSOCKSライブラリのプリロードができるようになる。

ちなみに、suid rootを外すには、バイナリ配布のRPMをインストールした後、自分で“ chmod 511 ssh ”のコマンドを実行してもよい。

このほか、Red HatのRPMでは、いくつかのパッチがあてられている。また、OpenSSHグループが配布しているRPMでは、openssslライブラリがスタティックリンクされる。

▼リスト　RPMファイルのconfigureオプション

```
configure \
 --sysconfdir=%{_sysconfdir}/ssh \
 --with-tcp-wrappers \
 --with-ipv4-default \
 --with-rsh=/usr/bin/rsh \
 --with-default-path=
/usr/local/bin:/bin:/usr/bin:/usr/X
11R6/bin
```

OpenSSHの設定ファイル

OpenSSHには、サーバ、クライアントそれぞれの動作を指定する設定ファイルが用意されている。

■サーバ（sshd）の設定ファイル

/etc/sshd_config（または/etc/ssh/sshd_config）は、sshdを起動するときに読み込まれ、sshdへの接続をコントロールする。

インストール直後の設定ファイルは、標準的なセキュリティの設定になっているが、一通りチェックし、必要に応じて設定内容を変更しておくとよい。設定ファイルで指定できる設定内容はマニュアルに詳しく書かれているが、特に注意すべき点を紹介しておこう。

・Protocol

“ Protocol ”は、接続用のプロトコルを指定する。プロトコル2を優先し、プロトコル2で接続できなければプロトコル1で接続する場合は“ Protocol 2,1 ”とする。もしプロトコル2だけを許可するのであれば、“ Protocol 2 ”とする。

・PermitRootLogin

rootのログインを許可するかどうか。デフォルトは“ PermitRootLogin yes ”だが、“ PermitRootLogin no ”にすると、rootのログインを拒否する。また、“ PermitRootLogin without-password ”としておくと、rootで接続する場合はパスワード認証を拒否する。インターネットに接続されているサイトの場合、“ no ”または“ without-password ”がいいだろう。

・StrictModes

“ yes ”の場合、ファイルのパーミッションを厳密にチェックする。デフォルトは“ yes ”で、変更しないほうが安全。

・X11Forwarding

X Window Systemの転送の許可（yes）または不許可（no）。デフォルトは“ no ”だが、“ yes ”にしても特に問題はない。

・RhostsAuthentication

・RhostsRSAAuthentication

rhosts、hosts.equivを使うr＊コマンドと同等のホスト認証またはユーザー認証の設定だが、rhostsは安全ではないので、どちらも“ no ”にしておくほうがよい。

・RSAAuthentication

RSA認証の利用の設定。ここは“ yes ”にしておく。

・PasswordAuthentication

RSA認証が使えない場合に、UNIXパスワードでのログインを許可するかどうか。セキュリティ的にはここを“ no ”にして、パスワード認証を拒否するほうが安全。

・PermitEmptyPasswords

空のパスワードを許可するかどうか。“ no ”にしておくべきだろう。

・IgnoreUserKnownHosts

ユーザーごとの~/.ssh/known_hostsを使うかどうか。デフォルトでは“ no ”になっているが、“ yes ”にすると、/etc/ssh/ssh_known_hostsだけを利用し、特定のホスト鍵を持つホスト以外からの接続を拒否する。

・Ciphers

バージョン2で使用する対称暗号のプロトコルを指定する。デフォルトは“ 3des-cbc,blowfish-cbc,arcfour,cast128-cbc ”だが、たとえばblowfishを優先、arcfourとcast128を使わないのであれば、“ blowfish-cbc,3des-cbc ”とする。

■クライアント（ssh）の設定ファイル

クライアントのデフォルト設定は、/etc/ssh_config（または/etc/ssh/ssh_config）で設定することができる。

ソースからコンパイルした場合、インストール直後のssh_configの内容は、すべての項目がコメントアウトされた状態になっているが、ユーザーごとの設定内容を~/.ssh/configで設定することができるので、ふつうは~/.ssh/configで個人ごとに設定することになる。もちろん、すべてのユーザーに共通の設定があれば、ssh_configで設定することができる。なお、ホスト全体に有効なssh_configも個人ごとの~/.ssh/configも、同じ書式で記述する。

設定項目の中には、コマンドラインで指定できるものもあり、同じ内容の設定は、コマンドラインでの指定が優先する。また、~/.ssh/configで設定された内容は、ssh_configよりも優先する。

詳しいconfigの書式はマニュアルを参照していただくこととして、ここでは設定でのポイントをいくつか紹介しておこう。

▼リスト2　環境変数でパスフレーズ入力ツールを指定

```
export SSH-ASKPASS=/usr/libexec/ssh/gnome-ssh-askpass ssh-add < /dev/null
```

注：Sawfishを使っている場合、gnome-ssh-askpassが使えないことがある。その場合は、x11-ssh-askpassを指定する。

・Host

configファイルは、必ずHostのキーワードで始まる。Hostには、接続先のホスト名を記述するが、同じ設定で接続するホストが複数ある場合は、カンマで区切って指定する。また、"＊"、"？"をワイルドカードとして使うこともできる。

たとえば、特定のドメインのホストについては、それ以外のホストと異なる設定で接続したい場合は、次のように設定する。

```
Host ＊.example.com
```

example.com用の設定内容

```
Host ＊
```

デフォルトの設定内容

・Protocol

接続用のプロトコルを指定する。プロトコル2を優先し、プロトコル2で接続できなければプロトコル1で接続する場合は“ Protocol 2,1 ”とする。プロトコル2だけを許可するのであれば、“ Protocol 2 ”とする。

・CheckHostIP

接続先ホストのIPアドレスをチェックする。IPアドレスは認証に使われないが、“ yes ”にしておくとDNSの改竄などを検出できる。

・Cipher

プロトコルバージョン1の対称暗号アルゴリズムを指定する。デフォルトは“ 3des ”だが、“ blowfish ”のほうが効率が高い。-cオプションで指定することもできる。

・Ciphers

プロトコルバージョン2で使う対称暗号アルゴリズムの種類と優先度を指定する。デフォルトは“ 3des-cbc,blowfish-cbc,cast128-cbc,arcfour ”。-cオプションで指定することもできる。

・Compression

・CompressionLevel

データの圧縮の設定。Comressionを“ yes ”にすると、データ圧縮が有効になり、高速なデータ転送ができる。

また、CompressionLevelでは、データ圧縮を行う場合の圧縮レベルを1（低圧縮率）から9（高圧縮率）までの数値で指定する。デフォルトでは6。

・DSAAuthentication

・RSAAuthentication

DSA／RSA認証をするかどうかを指定する。DSA／RSAで認証を行う場合は“ yes ”。

・IdentityFile

・IdentityFile2

RSA認証（IdentityFile）またはDSA認証（IdentityFile2）用の秘密鍵が収められているファイル名で、複数をカンマで区切って指定することもできる。デフォルトは~/.ssh/identity。DSA認証には~/.ssh/id_dsaを使うことも多い。また、-iオプションで指定することもできる。

・PasswordAuthentication

パスワード認証を使うかどうかを指定する。

・RhostsAuthentication

・RhostsRSAAuthentication

rhostsを使った認証の可否を指定する。特にrhostsを使う必要がある場合を除き、“ no ”にしておく。

・FallBackToRsh

SSHプロトコルでの接続が拒否された場合に、自動的にrshを起動する場合は“ yes ”にする。特にrshを使わなければならない理由がある場合を除き、“ no ”にしておく。

・UseRsh

rshを使う場合には“ yes ”を指定する。特にrshを使わなければならない理由がある場合を除き、“ no ”にしておくほうがよい。

・ForwardAgent

接続先のマシンにssh-agentの接続を転送する場合に、“ yes ”にする。デフォルトは“ no ”だが、“ yes ”にしておいたほうが便利。-Aオプション（転送）、または-aオプション（非転送）で指定することもできる。

・ForwardX11

SSHでX11のアプリケーションを使う場合は“ yes ”にする。

・GatewayPorts

ポートの転送を行う場合に、“ yes ”にする。

・StrictHostKeyChecking

未知のホスト鍵を自動的に~/.ssh/known_hostsまたは~/.ssh/known_hosts2に追加するかどうかを指定する。デフォルトは“ no ”で、自動的に追加する。このオプションを“ yes ”にすると、自動的な追加が禁止されるため、トロイの木馬が仕掛けられる可能性の高いホストでは、“ yes ”にしておくことが望ましい。

・User

ログインユーザー名を指定する。こ

れは異なるマシンでそれぞれユーザー名が異なる場合に、各マシンごとに指定しておくと、ユーザー名をタイプせずにログインできる。たとえば、次のようにしておくと、ドメインがexample.comのホストには、ichiroというユーザー名でログインする。

```
Host *.example.com
  User ichiro

Host *
```

デフォルトの設定内容

鍵の管理

SSHのセキュリティのポイントは鍵の管理だ。SSHの認証では公開鍵を使って認証が行われるため、鍵がすり替えられればセキュリティが破られることになる。そのため、SSHには鍵のすり替えを未然に防止したり、すり替えられたことを警告するためのさまざまな工夫が組み込まれているが、ユーザー側でも鍵の漏洩に注意するなど、管理に気を配ることが必要だ。

たとえば、ホストの秘密鍵が盗まれると、第三者からの接続を受け付けてしまう可能性や、DNSの詐称と組み合わせて偽のホストに接続し、データを盗まれる可能性がある。また、ユーザーの秘密鍵が盗まれれば、そのユーザーの権限でログインされてしまう。

ホストの秘密鍵は、rootユーザー以外には読み出せないようにすることで防衛するしかないが、何らかの理由でホストの秘密鍵が流出した可能性がある場合は、ホストの公開鍵ペアを作り直すとともに、必ずknown_hostsまたはknown_hosts2に記録された古いホストの公開鍵を削除する。known_hostsやknown_hosts2に古い鍵が残っていると、盗まれた鍵でアクセスされてしまうからだ。

ユーザーの秘密鍵は、他人に読み出せないようにすることはもちろん、必ず暗号化し、十分に長く、解読が困難なパスフレーズで守る。また、いうまでもなく、安全ではない環境からのパスフレーズの入力（たとえば、普通のtelnetでログインした状態でのパスフレーズの入力）は絶対に行わないようにする。鍵が盗まれたり、パスフレーズが盗まれた可能性がある場合は、やはり公開鍵のペアを作り直し、必ず各ホストに保存されているauthorized_keysやauthorized_keys2に記録されている古い公開鍵を削除する。

また、直接コンソールからログインしないリモートマシンでは「ユーザー鍵を作る必要はない」ということも覚えておきたい。ssh-agentを使って認証を転送するようにすれば、ユーザーの鍵が各マシンになくても、ssh-agentに記録されている鍵を使って認証を行う。鍵がなければ鍵が漏洩する可能性もなくなるので、それだけ鍵の管理も容易になり、安全性を高められる。

このほかには、ときどきknown_hostsとknown_hosts2、authorized_keysとauthorized_keys2の内容をチェックして、不審な鍵がないことを確認しよう。不審な鍵が見つかった場合、そのマシンはクラックされている可能性がある。

Windowsから使うSSH

SSHはUNIX系のOSを前提に設計されたツールだが、WindowsやMacintoshなどのパソコンOSで使えるSSHクライアントも増えている。パソコンOS用のSSHクライアントとしては、SSH社やF-Secureから発売されている商用のSSHが使いやすいが、フリーのクライアントでは、TeraTermをベースにSSHプロトコルを使えるようにしたTTSSHやPortForwarderなどが利用できる。ここでは、Windowsのフリーなsshクライアントの使い方を簡単に紹介しておこう。

■TeraTerm ProとTTSSH

TeraTerm Proは、寺西 高氏が作成した高機能なターミナルエミュレータソフトウェアだ。このTeraTerm Proの機能を拡張し、SSHプロトコル1.5でのリモートログイン／ポートフォワード機能を可能にするのが、Robert O'Callahan氏によって作成されたTTSSHである。TeraTerm ProとTTSSHは、それぞれ、http://hp.vector.co.jp/authors/VA002416/、http://www.zip.com.au/~roca/ttssh.htmlから入手できる。

TTSSHを使うには、まずTeraTerm Proをインストールし、次にzip形式のTTSSHのアーカイブを展開したあと、必要なファイルを手作業でTeraTerm Proをインストールしたディレクトリに移動する。具体的には、ttssh.exeとlibeay32.dll、ttxssh.dllの3つのファイルをTeraTerm Proがインストールされているディレクトリに移動する。

■TTSSHの設定

TTSSHの設定は、ttssh.exeを起動し、［Setup］メニューで行う。

リモートログインのターミナルに関する設定は、TeraTerm Proの設定と同じで、Setupメニューの［Terminal］、［Window］、［Font］、［Keyboard］などを必要に合わせて設定すればよい。

SSH接続に特有の設定としては、Setupメニューの［SSH］、［SSH

Authentication］での設定がある。また、TTSSHでポートフォワード機能を利用する場合は、［SSH Forwarding］を使って設定する。以下、それぞれの設定のポイントを説明しよう。

まず、［SSH］のウィンドウでは、圧縮レベルと暗号アルゴリズムを選ぶ。デフォルトの設定は、特に変更する必要はないが、IDEAとRC4は安全ではないとされているので、暗号アルゴリズムはBlowfishまたは3DESを選んでおく（**画面3**）。OpenSSHのknown_hostsと同等のホスト鍵の管理が必要であれば、一番下の段の「SSH Known Hosts」に適当なファイル名を入力する。

次に［SSH Authentication］のウィンドウを開き、ログインするユーザー名と認証の方式を選ぶ。UNIXパスワードでのログインは、「Use plain password to log in」を選ぶが、LinuxマシンでRSA鍵を作ることができれば、「Use RSA key to login」を選び、RSA鍵を使ったログインを設定するとよい（**画面4**）。この場合、RSA鍵が必要になるが、TTSSHにはssh-keygenコマンドが付属していないので、Linux上のssh-keygenコマンドを使って作成することになる。具体的には、“ ssh-keygen -f ttsshkey ”を実行し、作成されたttsshkeyをTTSSHがインストールされたディレクトリ、または適当なディレクトリにコピーする。ttsshkey.pubは特にコピーする必要はなく、そのままサーバ側のauthorized_keyに付け加えておけばいいだろう。

さらにポートフォワードの設定が必要なら、［SSH Forwarding］で転送するポートの設定を行う。たとえば、POP3の転送を行いたい場合は、次の手順で転送の設定を加える（**画面5**）。

1. **［Add］ボタンをクリック → ポート転送の設定ウィンドウが表示される**
2. **「Forward local port」のドロップダウンメニューから、pop3を選ぶ**
3. **「to remote machine」で、POP3サーバのホスト名を入力する**
4. **「port」のドロップダウンメニューで、「Forward local port」で選んだサービスのポートと同じものを選ぶ**

POP3以外に必要なサービスがあれば、上記の設定を繰り返す。なお、FTPの場合、ポート転送を行うポートはftpとftp-dataの両方を設定しておくこと。

このほか、Setupメニューの［TCP/IP］で接続先のホストを追加しておくと、メニューから接続先を選べるようになる。［TCP/IP］の「Host list」のテキストボックスに“ サーバ名／F＝ssh.ini ”などと入力、「Port#：」にSSHのポート（デフォルトでは22）を入力して、［Add］ボタンで設定を追加しておくと便利だ。

以上の設定が終了したら、設定内容をファイルに保存しておこう。Setupメニューから［Save setup］を選び、“ ssh.ini ”などの名前で保存する。

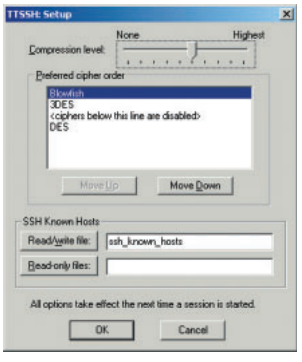
■TTSSHでログインする

TTSSHの設定が終了したら、telnetと同様にTTSSHの接続メニューから、設定したホストを選んで接続するが、接続前にTTSSHの認証ウィンドウが表示される点が、ノーマルのTera Term Proと異なる部分だ。このウィンドウには、UNIXパスワードではなく、作成したRSA鍵のパスフレーズを入力する。これで、SSHサーバに接続し、安全な通信が可能になる。

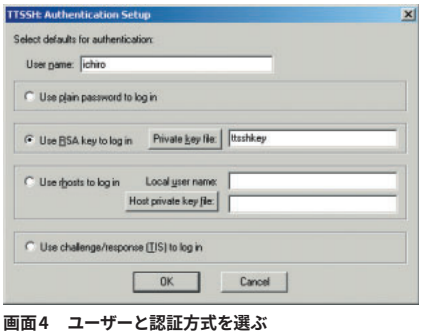
また、ポートフォワードを設定している場合は、クライアントの接続先を実際のサーバではなく、localhostにしておくこと。TTSSHでログインした後は、ポート転送の機能によって設定したPOP3などのクライアントがSSHの暗号化されたチャネルで通信するようになる。

■PortForwarderを使ったポート転送

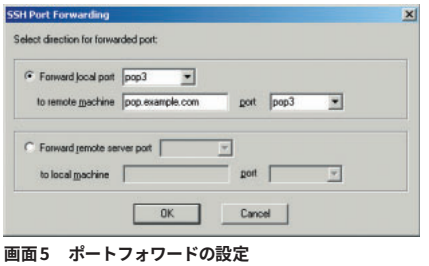
PortForwarderは、Portable OpenSSH、OpenSSLをベースとしてWindowsパソコンでSSHのポート転送機能を実現するフリーソフトウェアで、登 不二雄氏によって作成された（http://portforwarder.nttsoft.net/JP/）。PortForwarderは、Windows 95／



画面3　暗号の方式を選ぶ
TTSSH v1.54では、安全性の低いIDEAとRC4は削除された。



画面4　ユーザーと認証方式を選ぶ
できるだけRSA鍵を使ったログインを設定しよう。



画面5　ポートフォワードの設定
localhostのpop3ポート（110）に接続して安全にメールを転送できる。

98／NT／2000のほか、Windows CEでも動作し、SSHプロトコルバージョン1.5に対応する。

TTSSHにもポート転送機能があるが、PortForwarderは、その名の通りポート転送を主な目的として作られているので、ポート転送機能に関しては、こちらのほうが使いやすいだろう。もちろん、ポート転送機能でtelnetポートも転送できるので、その場合はノーマルのTeraTerm ProでSSHのチャネルを通じたリモートログインが可能になる。

■PortForwarderの設定

PortForwarderの設定は、OpenSSHで使われるものと同等のconfigファイルで行う。たとえば、**リスト3**のconfigファイルは、すべての接続ホストに対し、FTP、telnet、SMTP、POP3の各プロトコルをローカルホストに転送する。

リスト3のconfigファイルをテキストエディタで作成し、Windowsの適当なディレクトリに保存する。また、このときRSA鍵を“identity”という名前で、接続先のホストの公開鍵を含むファイルを“known_hosts”という名前でそれぞれ、configファイルと同じディレクトリに保存する。known_hostsは、Linuxで使っているものをそのままコピーすればいいだろう。TTSSHでRSA鍵のファイルを使っている場合は、そのファイルをidentityファイルとして使うことができる。identityファイルはLinuxで作ってもPortForwarderで作ってもかまわない。

■PortForwarderを使う

PortForwarderを起動すると、接続先を指定するウィンドウが表示される（**画面6**）。Hostのテキストボックスに接続先を入力または選択する。また、初めて起動するときには、configファイルを「Config File」の横の［...］ボタンをクリックして選択する。あとは［Connect］ボタンをクリックすれば、ユーザー名、パスワード／パスフレーズなど、接続に必要な情報の問い合わせの後（**画面7**）、指定されたSSHサーバに接続、PortForwarderのウィンドウの「Status」が「Connected」に変わり、ポート転送が開始される。

configファイルで設定したポートがWindows側（localhost）に転送されるので、telnetやFTPなどのクライアントでlocalhostに接続すると、接続先のマシンにつながり、安全な通信が可能になる。

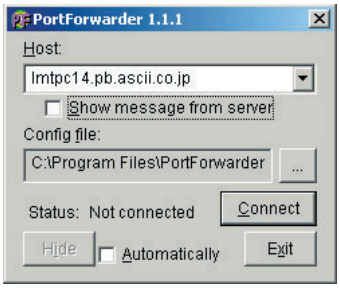
■WindowsでのSCP

Windowsでのリモートログインやポート転送は、TTSSH、PortForwarderなどでほぼ文句なく使えるが、やや弱い部分がscp相当のファイル転送である。MindTermなど、scp機能が組み込まれたプロダクトもあるが、GUIでの操作ができない。

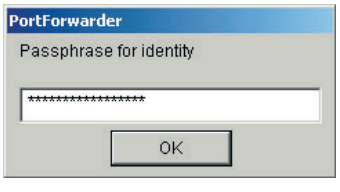
GUIをサポートするscp相当のWindowsアプリケーションとしてはWinSCP（http://winscp.vse.cz/eng/）というフリーソフトウェアがあるのだが、残念ながら日本語のファイル名の問題やRSA認証ができないなどの問題がある（**画面8**）。RSA認証ができないため、サーバ側の設定でパスワード認証を有効にする必要があり、セキュリティ上の弱点を抱えることになるので、むしろポート転送を設定して一般のFTPコマンドを使ったり、コマンドライン版のscpを使うほうがいいかもしれない。

▼リスト3　configファイルの例

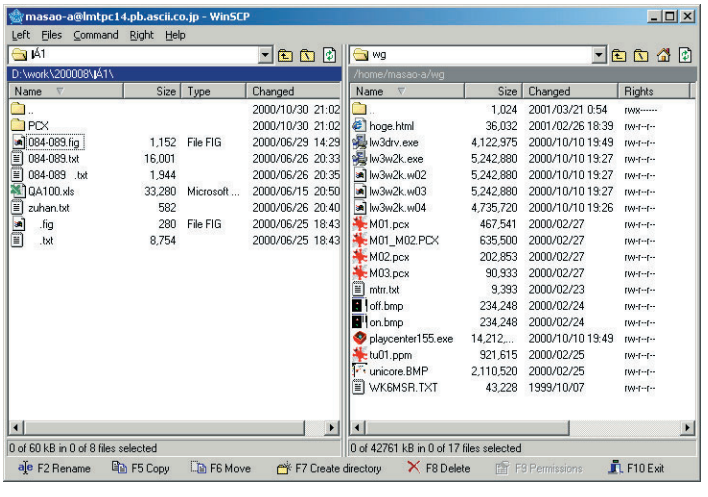
```
Host *
  User myuid
  # forward ftp
  LocalForward 21 localhost:21
  # forward telnet
  LocalForward 23 localhost:23
  # forward smtp
  LocalForward 25 localhost:25
  # forward pop3
  LocalForward 110 localhost:110
```



画面6　PortForwarderの設定
接続先とconfigファイルを指定する。

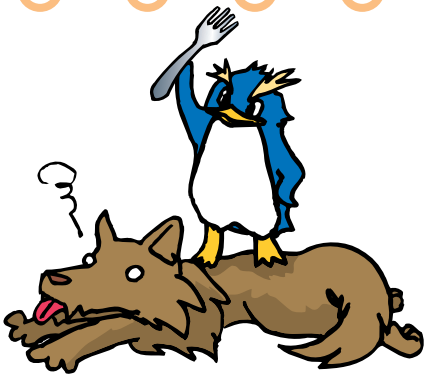


画面7　パスフレーズを入力
ここで、RSA認証が使えない場合はパスワード認証になる。



画面8　WinSCPの画面
RSA認証ができないのと、日本語ファイル名に対応していないのが残念。

ネットワークサーバのセキュリティを固める



文：安田幸弘
Text : Yukihiro Yasuda

インターネットに接続されているシステムの管理者にとって、ネットワークサーバのセキュリティは常に注意が必要な部分だ。最近は、ネットワークスキャナなどで無差別にIPアドレスをスキャンして、セキュリティの甘いサイトを見つけようとする不心得者も少なくないため、「ちょっと実験」のつもりでいいかげんな設定で立ち上げたサーバにまで、すかさず攻撃が加えられたりするので油断は禁物だ。

不要なサーバは起動しないこと、常にセキュリティの勧告に目を配り、セキュリティホールのないサーバを運転することは基本だが、設定によっては思わぬトラブルの元になることもある。ここでは、セキュリティという面から基本的なネットワークサーバの設定のポイントをチェックしよう。

sendmail

電子メールはインターネットの基本であるばかりでなく、UNIX系のOSではさまざまなシステム情報をユーザーに知らせてくれるメッセンジャーのような役割を持つサービスだ。それだけに、メールサーバの運転を止めることは難しく、攻撃者に狙われやすいサーバの筆頭格だ。

電子メールサーバとして最も利用者の多いsendmailは、さまざまなセキュリティ上の問題点を抱え、攻撃者の格好のターゲットとされてきた。頻繁なセキュリティアップデートを嫌って、セキュリティ的な問題が少ないといわれるqmailやPostfixなどに乗り換える管理者も少なくないが、ソースコードの大掃除の結果、sendmailも以前のように大きなセキュリティホールの心配はほぼなくなったようだ。

だが、どんなにsendmailが安全なソフトになったとしても、あるいはqmailやPostfixのようなほかのMTAに入れ替えたとしても、現在の電子メールそのものに少なからぬセキュリティ的な問題点があるということは忘れてはならない。たとえば、接続の認証やメールの不正中継、メールの盗聴、メールで増殖するワーム、カーネルの問題やMTAから呼び出されるプログラムメーラの問題など、単に「マシンが乗っ取られる」といった、ある意味ではわかりやすいトラブル以外にも、広義のセキュリティという意味で注意すべき点がある。

メールのようにセキュリティ的に敏感なサービスに関しては、プラスアルファの注意を払うというセンスを身につけておきたい。

■setuid rootの危険？

Postfixやqmailのような新しいメールサーバプログラムと比べて、sendmailは実行ファイルがsetuidされているという点が、セキュリティ上最大の問題だといわれる。

ご存じの通り、setuidとはちょうどプログラムの内部で“ su ”コマンドを実行するようなものだ。具体的にいえば、sendmailの場合、rootの権限にsetuidされているため、誰が起動してもsendmailは自動的にrootの権限で実行が始まる。これは、SMTPの特権ポートを必要とすること、スプールの書き込みにrootの権限が必要であることなどが理由だ。ちなみに、ユーザーのメールボックスへの書き込みは、sendmailとは別に、procmailやmail.localなどのローカルメーラが実行するが、これも共用のディレクトリにユーザーごとのメールボックスを作る必要があるためにsetuidされている。

setuidは、正しく使えばそれ自体で危険なものではないのだが、sendmailのように複雑な動作をするプログラムでは、少しのコーディングのミスでも大きなセキュリティホールになり得る。そして、すべてのプログラムにバグがあるとすれば、rootにsetuidすることに潜在的な危険があることは理解できるだろう。

もちろん、単にsetuidしなければいいというものではないが、qmailやPostfixなどはsetuidの危険を避けるために、setuidのメカニズムを使わずにメールの配信ができるようにコードが工夫されている。

sendmailを安全に使う

メールサーバのセキュリティについては、システム的なセキュリティと、接続認証関連の問題、そのほか一般的なメール配信における安全の確保という3つのポイントがある。

■システムセキュリティ

Red Hatなど、多くのディストリビューションでは、sendmailのパッケージにセキュリティパッチや独自のセキ

ュリティ機能が組み込まれている。常に最新のパッケージを利用するようにすれば、基本的にsendmailのバグなどによるセキュリティ上の問題のほとんどを避けることができるようになっている。

たとえば、オリジナルのsendmailをソースからビルドした場合、特に指定しなければプログラムメーラには/bin/shが使用されるが、Red Hatのパッケージをインストールすると、デフォルトのプログラムメーラが/bin/shよりも安全性が高いsmrshになる。smrshは、余計な環境変数をクリアし、指定されたプログラムだけを起動することで予期しないプログラムの実行を防ぎ、システムの安全を高めることができる。最新版の機能をすぐに使いたいなど、ソースからのビルドが必要な場合は、配布パッケージのデフォルトの設定を参考にするといい。

しかし、いくらパッケージが安全でも、利用方法によってはシステムのセキュリティをおびやかす設定は可能である。たとえば、smrshは予期しないプログラムの実行を防ぐとはいえ、管理者がインストールし、smrshに実行を許可したメーリングリストやメールフィルタなどのプログラムに問題があれば、そのプログラムがセキュリティホールになってしまう。

また、標準的なsendmail.cfは、ほとんどの利用者にとって安全な設定になっているが、添付されているsendmail.cfの内容を変更する場合も、十分に注意したい。特に、パーミッション関連のオプション（DontBlame Sendmail関連）の変更は、セキュリティを弱めることになる。このため、できるかぎり、パーミッションは変更するべきではない。

このほかに、パーミッション関連では、以下のような点にも注意しておきたい。

- **aliasesやmailertableなどのファイルはroot以外は書き込み禁止にする**
- **キューディレクトリはroot以外は書き込み禁止にする**
- **forwardファイルは、本人以外の書き込みを禁止する**
- **aliasesでのインクルードは、所有者のパーミッションに注意する**

■不正中継、SPAMへの対処

sendmailは、メールのルーティング機能を持つ。つまり、受け取ったメールを自動的に適当な宛先に転送する機能で、インターネットが商用化されるまではほとんど無制限にメールの転送が開放されていた。しかし、最近はインターネットのメールを使った悪質なSPAMなどに対する批判から、第三者からのメールは転送を拒否することが推奨されている。

パッケージとして配布されているメールサーバでは、デフォルトでこのような不正中継を防ぐために、ほかのホストからの中継を拒否するように設定されているので、特にsendmail.cfを変更する必要はないはずだ。

しかし、デフォルトの設定では、自サイトのマシンでもIPアドレスが異なるとそのマシンからのメールを受け取らない。また、これだけでは不正中継は防げても、ジャンクメールを送りつけてくるSPAMを拒否することはできない。

sendmailは、sendmail.cfの中で、メールの受信や転送の条件をきめ細かく設定することができるようになっている。ディストリビューションに標準添付されるsendmail.cfでは、ほとんどの場合、いくつかの設定ファイルで接続の許可／不許可を指定するようになっている。

linuxconfなどでこれらの設定ができるが、以下のファイルに基本的な接続許可に関する設定を書き込んで設定することもできる。

マシンの別名

→ `/etc/sendmail.cw`

接続を許可／拒否するアドレス

→ `/etc/mail/access`

それぞれの書式は**リスト1**、**2**のとおりだ。

なお、/etc/mail/accessを書き換えた場合は、“ makemap hash /etc/mail/access < /etc/mail/access ” を実行して、データベースの内容を更新しておくこと。

■そのほかの設定

電子メールの転送に使われるSMTPには、さまざまなコマンドが定義され

▼リスト1　sendmail.cwの書式

```
foo.example.com
www.example.com
mail.example.com
```

▼リスト2　accessの書式

```
localhost.localdomain        RELAY
localhost                    RELAY
somehost.spammer             REJECT
another.spammer              REJECT
bad.spammer                  DISCARD
```

接続を許可する場合はRELAY、拒否する場合はREJECT、エラーメッセージを返さずに接続を拒否する場合はDISCARDを指定する。

ているが、その中には通常のメール転送には使われないEXPNやVRFYといったコマンドがある。

EXPNは、メールの別名アドレス（alias）から実際のアドレスを引くためのコマンド、VRFYは指定されたメールアドレスが存在しているかどうかをチェックするためのコマンドだが、これらのコマンドはプライバシーの問題をはらむと同時に、思わぬセキュリティホールになる可能性がある。そのため、最近ではこれらのコマンドの実行を禁止しているサイトも少なくない。

電子メールのプライバシー設定は、sendmail.cfの“PrivacyOptions”で行うが、通常は“PrivacyOptions＝authwarnings”になっていることが多い。これは、電子メールのSMTP FROMの情報が詐称された場合に、警告のヘッダをつけるというものだが、“PrivacyOptions＝authwarnings,noexpn,novrfy”とすることでEXPNとVRFYコマンドの実行を禁止できる。このオプションではこのほかにもいくつかの設定ができる。たとえば、“PrivacyOptions＝goaway”とすると最も厳しいプライバシー設定ができる。

このオプションは、cfツールを使ってsendmail.cf全体を書き換えることで指定することもできるが、/etc/sendmail.cfを直接書き換えてsendmailを再起動してもよい（**リスト3**）。

Red Hatの場合、/usr/lib/sendmail-cf以下のツールで自サイト専用の接続設定を組み込んだsendmail.cf作成することができる。Red Hatでは、デフォルトのsendmail.cfは、cf/redhat.mcを使って“m4 /usr/lib/sendmail-cf/cf/redhat.mc > /etc/sendmail.cf”とすることで作ることができる。新しく独自のsendmail.cfを作るのであれば、redhat.mcをベースに必要なオプションを付けて、上記のコマンドを実行すればよい。

■procmailによるフィルタリング

Red Hat系のディストリビューションでは、sendmail.cfでローカルメーラにprocmailが指定されているので、procmailを使った個別のメールのフィルタリングをすることもできる。ちょっとしたスクリプトが必要になるが、たとえば、メールの添付ファイルを調べて、ウイルス混入の疑いがあれば、そのメールに警告を入れるといったことも可能だ。マシン全体のprocmailの設定は、/etc/procmailrcで、個人ごとの設定は~/.procmailrcで行うことができる。

BINDを安全に使う

BINDは、sendmailと同様に、独立したインターネットサーバの運転に欠かせないドメインネームサーバだ。やはりsendmailと同じく、非常に複雑で高度なプログラムで、これまでにも多くのセキュリティホールが指摘されている。

BINDが持つセキュリティ上の問題は、ルートの権限が乗っ取られるというような直接的なものもあるが、DNSのデータを書き換えてドメインを乗っ取ったり、ドメインを詐称してトロイの木馬を仕掛けたマシンにアクセスさせ、データを盗んだり、あるいは内部のネットワークの情報を取得して、サイトのクラッキングに使うといったものもある。いずれにしても、インターネットの接続に関わる重要な働きをするサーバだけに、セキュリティの確保が大切である。

■安全なnamedの実行

最近話題になったBINDの問題に、バッファオーバーフローのテクニックを使ったroot権限の奪取がある。これは、DNSのコマンドであるnamedがroot権限で動作している場合に、namedのコーディング上の欠陥を狙ってシェルを実行させるというものだが、インターネットを越えてターゲットマシンのrootとしてログインすることができるため、影響は大きい。

namedをrootの権限で起動すると、namedから起動されたシェルがrootの権限で動いてしまうため、マシンの奪取を防ぐために最近はnamed専用に

▼リスト3　sendmail.cfでいずれかのPrivacyOptionsを指定する

```
O PrivacyOptions=authwarnings
O PrivacyOptions=authwarnings,noexpn,novrfy
O PrivacyOptions=goaway
```

Column

sendmailデーモンは必要か？

sendmailをデーモンとして起動すると、外部からのSMTP接続を受け付ける。SMTPで配信されるメールを受け取る必要がある場合は、sendmailを起動しておく必要があるが、ダイヤルアップでインターネット接続するマシンやノートPCなど、SMTPによるメールの配信を受け付ける必要のないマシンでは、sendmailを起動しておく必要はない。また、sendmailをデーモンとして起動しなくても、外部にメールを送出したり、syslogdなどによって送られるメールをローカルマシンのメールボックスで受け取ることは可能である。

“ bind ”などの疑似ユーザーを作り、namedに“ -u bind ”オプションをつけてroot以外の権限で運用することが推奨されている。こうしておけば、仮にアタックが成功しても、ダメージは小さくて済む。

ちなみに、namedに限らず、この種の問題を未然に防ぐために、マシンで立ち上げるデーモンはできるだけroot以外の権限で実行するべきである。なお、bindのユーザーアカウントで単にnamedを起動してもよさそうな気もするが、namedは特権ポートを使うために、起動時は必ずrootの権限で起動しなければならない。

また、このほかには、namedが動作する環境をchrootで変更し、仮に乗っ取られたとしても、ほかのディレクトリにアクセスができないようにするという方法もある。bindなど、権限が限られたuidでシェルを起動されても、一度ログインされてしまうとpasswdをはじめ、マシンの攻撃に使える情報を提供してしまうことになるが、アクセスできるディレクトリを制限してしまえば比較的被害が少なくて済む。

chrootでnamedを起動するには、スタティックリンクされたnamedとDNSの動作に必要なファイル類のすべてを1つのディレクトリ以下にコピーし、次のコマンドでnamedを起動する。

```
# /usr/sbin/named -t /var/named -u bind -g bind
```

なお、namedをchrootで起動するには、標準のパッケージを使うことができないので、自分でコンパイル、設定をしなければならない。セキュリティホールがフィックスされた安全なバージョンのbindを使えば、特にchrootで起動する必要はないはずだ。

■DNSデータのアクセス制限

DNSの情報を見ると、サイトの構成を知ることができる。公開されているサーバのDNS情報は、そもそもクライアントから参照されることが前提だが、ある程度の規模のサイトになると、ゾーン転送によってマシンのアドレスを取得され、それを元にポートスキャンなどの攻撃が加えられるなど、サイトのプライバシー情報を取得されることになってしまう。

BIND 8.x以降のバージョンでは、ゾーン転送の制限を設定することができるようになったので、セカンダリDNSサーバだけにゾーン転送を許可するようにしておくことが望ましい。**リスト4**のように、ゾーン転送を許可するセカンダリネームサーバのIPアドレスを設定する。

FTPを安全に運用する

wu-ftpdは、多機能なFTPサーバとして広く使われており、LinuxでもRed Hatをはじめ、多くのディストリビューションで標準のFTPサーバに採用されている。

しかし、wu-ftpdはセキュリティに関するバグが多いことでも有名なプロダクトで、wu-ftpdを利用しているサイトはセキュリティ情報に十分注意する必要がある。特に、wu-ftpdは、オリジナルバージョンを開発したワシントン大学が1995年ごろからメンテナンスをやめてしまい、長い間、重大なセキュリティバグフィックスの空白時期が続いた。その間、ボランティアグループによってメンテナンスが続けられ、さまざまなバージョンの“ wu-ftpd ”が出回ることになったのだが、これが混乱を助長したようだ。

だが、その後、さまざまな派生バージョンを統合してwu-ftpd開発グループが活動を開始し、現在はセキュリティホールがフィックスされたバージョン2.6.1がリリースされている。

■wu-ftpdのセキュリティのポイント

セキュリティ面でのポイントは、何といってもセキュリティに関するバグがフィックスされた最新版を使うということに尽きる。Linuxの場合、ディストリビューションによっては常に最新版が提供されるとは限らないが、セキュリティ上の問題が見つかれば、その問題をフィックスしたセキュリティアップデートがリリースされるので、バグフィックス版にアップグレードすればよい。

またftpdのセキュリティ問題のほとんどは、Anonymous FTPの部分で発生する。必要がなければAnonymous FTPを使わないように設定するべきである。

さらに、FTPではパスワードや通信内容が暗号化されないため、パケットの盗聴によるパスワードやデータの漏洩の可能性がある。SSHのポート転送を使うと、通信内容が暗号化され、安全なFTPが利用できる。Anonymous FTPを使わないのであれば、wu-ftpdを無効にしてSSHのscpやsftpを使うほうが安全だ。

■Anonymous FTP の設定での注意

Anonymous FTPは、 anonymous

▼リスト4　ゾーン転送の制限をする

```
options {
    directory "/var/named";
    allow-transfer {
      XXX.XXX.XXX.XXX;
      YYY.YYY.YYY.YYY;
      ZZZ.ZZZ.ZZZ.ZZZ;
    };
};
```

（またはftp）のログイン名で誰にでもFTPサーバを利用できるようにする設定である。もちろん、プライベートなデータにはアクセスできないように注意しなければならないが、それ以外にも不正な利用を防ぐために、以下のようなポイントに注意したい。

- **anonymous ftp用のディレクトリにchrootする**
- **anonymous ftp用のpasswdファイルはダミーを使う**
- **ファイルをputできるディレクトリをincomingなどに制限する**
- **利用できるコマンドは必要なものに制限する**
- **incomingディレクトリではディレクトリの作成や読み出しは禁止する**
- **~ftpディレクトリは独立したパーティションに作成、またはディスククォータをかける**

▼リスト5　パーミッションの例

```
~ftp                dr-xr-xr-x   root  root
~ftp/bin            d--x--x--x   root  root
~ftp/bin/*          ---x--x--x   root  root
~ftp/etc            d--x--x--x   root  root
~ftp/etc/*          -r--r--r--   root  root
~ftp/lib            drwxr--r--   root  root
~ftp/lib/*          -rwxr--r--   root  root
~ftp/pub            dr-xr-xr-x   ftp   ftp
~ftp/pub/incomming  drwx-wx-wt   root  root
```

▼リスト6　/etc/ftpaccessの例

```
class   all   real,guest,anonymous  *
limit   all   10   Any              /etc/msgs/msg.dead
readme  README*   login
readme  README*   cwd=*
message /welcome.msg            login
message .message                cwd=*
log commands real
log transfers anonymous,real inbound,outbound
shutdown /etc/shutmsg
email <管理者メールアドレス>
chmod           no              anonymous,guest
delete          no              anonymous,guest
overwrite       no              anonymous,guest
rename          no              anonymous,guest
umask           no              anonymous,guest
# 圧縮やtarを許可する場合はyesにする
#compress       yes             all
#tar            yes             all
compress        no              all
tar             no              all
# uploadには、ファイル送信を許可するディレクトリを指定
upload  /home/ftp  *              no
upload  /home/ftp  /pub/incoming  yes  ftp  nobody  0600  nodirs
# ファイル名に許可される文字
path-filter     anonymous  /etc/msgs/pathmsg  ^[-A-Za-z0-9_\.]*$ ^\. ^-
path-filter     guest      /etc/msgs/pathmsg  ^[-A-Za-z0-9_\.]*$ ^\. ^-
# rfc822形式のパスワードを強制
passwd-check    rfc822 enforce
```

リスト5に、~ftpディレクトリのパーミッション設定例を、**リスト6**にwu-ftpdの設定ファイル/etc/ftpaccessの例を示す。さらに、/etc/ftpusersには、FTPアクセスを許可しないユーザーを指定する（**リスト7**）。

このほか、/etc/ftphostsで接続を許可、または拒否するユーザーとドメインを指定することもできる。必要があれば、次の書式で設定する。

```
allow ユーザー　ドメイン
deny　ユーザー　ドメイン　アドレス
```

日常のサーバ管理

セキュリティを確保するうえで、最も大切なことが日常的な管理だ。パソコン用のOSと違ってインターネットのサーバは、一度起動すれば何カ月も、場合によっては何年もシャットダウンせずに稼動し続けるが、利用にさしつかえるようなトラブルがない限り、特にメンテナンスはしないという管理者も少なくないらしい。しかし、クラッカーは何もサイトのクラッシュだけを狙うわけではない。

▼リスト7　/etc/ftpusersの例

```
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
operator
games
gopher
nobody
gdm
xfs
```

知らないうちに自サイトがほかのサイトへの攻撃の踏み台にされているかもしれない。こっそりとデータが盗まれているかもしれない。また、あるとき、突然原因不明の障害でダウンするかもしれない。そのようなトラブルを避けるためにも、常にメンテナンスを怠らないようにしたい。

日頃から管理しておきたい項目は次の通りだ。

■ログのチェック

メンテナンスの基本は、ログのチェックに始まる。Linuxの場合、ログは/var/log以下のディレクトリに保存されるが、サーバの挙動がおかしいときはもちろん、定期的にログの内容をチェックするようにしたい。/var/logに作られるファイルには**表1**のようなものがある。

しかし、これらのファイルには、毎日膨大な量のメッセージが記録されるため、すべてをチェックするのは、現実的にはかなり困難だ。とはいえ、こうしたログを丹念に調べてみると、意外と頻繁にほかのサイトから不正なパケットが送られていたり、侵入が試みられた痕跡などを発見できるはずだ。一度、そのような痕跡を見つければ、面倒だともいっていられなくなるに違いない。

だが、何でもかんでも記録してしまえばいいというものでもない。ログが増えると無数のメッセージに致命的な問題を示すメッセージが埋もれてしまい、発見が困難になる。

ログ管理の方法はディストリビューションによってデフォルトの設定が違い、また、ほとんどの場合、管理者は自分で管理しやすいように変更を加えている。マシンやサイトの事情に合わせて少しずつ記録の内容や形式を調整していくといいだろう。たとえばインストールされているサーバプログラムの数が増えれば、ログを分割したり、定期的にログのバックアップを取ることも有効だ。また、ログの改竄を防ぐために、侵入の疑いのあるメッセージはプリンタで打ち出すようにしているサイトもあるという。このほかにも“ err ”や“ failed ”などの文字列をgrepで拾うようなスクリプトを作って、cronで定期的に実行する、swatchなどのログ管理ツールでサーバの異常な挙動をピックアップし、不正な侵入などが疑われる場合は、携帯電話やポケットベルにメッセージを送るといった工夫をすることも管理者の心得だ。

もちろん、こうしたログの管理は、単に不正侵入の発見ばかりでなく、サーバやマシンの異常を早期に発見し、トラブルを最小限に食い止めることにもつながるので、まめにログのチェックをする習慣をつけたい。

■ファイルのチェック

ふだんから外部からの攻撃に気を配っていれば、そう簡単にセキュリティが破られることはない。だが、万一侵入されたときへの備えも重要だ。

サーバへの侵入があった場合、まずログが改竄され、syslogdが置き換えられ、ログに記録が残らないようにされることが多い。また、suやloginといったログイン関連のツールが置き換えられるかもしれない。また、sendmailのようにsuidされたプログラムに何かの仕掛けを施される可能性もある。さらに、発見が難しいファイル名や、ふだん使わないファイルに侵入のための「裏口」が仕掛けられることもある。

逆にいえば、これらの注意すべきファイルに変更が加えられたことを検出できれば、不正な侵入があったことを知ることができ、すぐに対策を取ることができる。そのため、自分がインストールしたファイルが第三者によって置き換えられていないことをチェックすることは、管理者にとって非常に重要な作業のひとつといえる。

すべてのファイルをひとつずつ管理者がチェックすることは不可能だが、Tripwireのようなソフトウェアですべてのファイルをスキャンし、CRCやハッシュ値を保存しておけば、ハッシュ値を比較することでファイルの置き換

syslogdで記録されるもの	
boot.log	ブート時のメッセージ
lpd.log	lpdの実行記録
maillog	メールの配信に関するログ
messages	各種のログ
spooler	uucpやINNなどの実行記録
そのほか	
cron	cronの実行記録
dmesg	ブート時のdmesgコマンドの記録 （dmesgはカーネル内のバッファにある記録を表示するツール）
lastlog	ログインの記録（lastlogコマンドで表示する）
secure	ログイン関連の認証の記録
sendmail.st	sendmailの統計情報
wtmp	直前のログインの記録
xdm-error.log	xdm実行時のエラー
xferlog	FTP転送の記録
httpd/＊	httpdの記録
samba/＊	sambaの記録

表1　/var/logディレクトリ以下のログファイル

えを知ることができる。

全部のファイルをスキャンすることはマシンの負荷が大きいため、1週間に1回ぐらいの頻度でチェックすることが多いが、syslogdのように、真っ先に狙われる重要なファイルや、suidプログラムのように危険なファイルは、毎日チェックしてもそれほど大きな負担にはならない。たとえば、GNUのtextutilsには、md5sumというファイルのハッシュコマンドがある。md5sumを使って必要なコマンドをチェックするスクリプトを毎日cronで実行するだけでもかなり効果的だろう（**リスト8**）。

もっとも、rootの権限を取られてこうしたスキャンツールやスキャンしたデータが置き換えられてしまうとどうしようもない。スキャンした結果のデータはフロッピーディスクに保存してチェック時にマウントしたり、暗号化して保存してチェック時にパスフレーズを入力して安全性を確保するなどの工夫が必要だ。

■ユーザーの管理

セキュリティに無関心なサーバの利用者は、最大のセキュリティホールかもしれない。管理者がどんなに注意していても、ユーザーがパスワードの管理を怠れば、外部から簡単にログインされてしまう。そして、ローカルユーザーには外部から攻撃するよりも多くの攻撃手段があるため、外部からの攻撃には頑丈なセキュリティも、一度ログインされればあっさりと破られてしまうこともある。自分以外にユーザーがいないサーバでも、たとえばテストのために一時的に作ったアカウントや、パスワードが空のアカウントを作ったまま、忘れてしまうこともある。

相手が人間だけに、ユーザーの管理は難しい部分が多いが、システム的な工夫の余地はある。たとえば、ユーザーが安易なパスワードをつけられないように、特別なバージョンのpasswdに置き換えたり、パスワードの有効期限を設定したり、定期的に使われていないアカウントの削除やパスワードが空のアカウントをチェックするようなスクリプトを走らせるということもできる。デフォルトのパスワードの有効期限は、/etc/login.defsの“PASS_MAX_DAYS”で指定できる。たとえば、“PASS_MAX_DAYS　200”とすると、200日後にパスワードが失効するので、それまでにパスワードを変更しなければならない。

また、メールやFTPだけしか使わないユーザーは、ログインの必要がないので、ログインシェルをダミーのファイルにしてもよい。具体的には、/etc/shellsに“/NO/LOGIN/USER”などの行を作り、ログインしないユーザーのログインシェルを“/NO/LOGIN/USER”に変更する。

このほかにも、ワンタイムパスワードを強制したり、FTPを閉じてSSHのファイル転送しか認めないようにするなど、ユーザーアカウントに対してさまざまな制限をかけることができる。

▼リスト8　syslogdの置換検出スクリプト例

```
#!/bin/sh
# 起動スクリプトに次のようなコマンドを入れ、必要なファイルの記録を取る
#   md5sum -b /sbin/syslog /usr/bin/w /usr/bin/last > /tmp/files.md5
#
if ! md5sum --status -c /tmp/files.md5 then
    md5sum -c /tmp/files.md5 | \
    mail -s "Critical files were replaced" root@example.com
fi
```

安全を守るそのほかのツール

Linuxで動作するセキュリティ関連ツールはたくさんある。最後に、代表的なツールをいくつか紹介しておこう。

■侵入検知 Snort

Snortは、外部から送られてきたパケットを捕捉して、そのパケットのパターンから攻撃を検出、記録するソフトウェアである。インターネットに常時接続されたマシンを持っている人なら、ぜひSnortをインストールしてどのような攻撃が加えられるのかを自分の目で実際に確認してみてほしい。早ければ数日のうちに、それまではどこか他人事のように思っていた「セキュリティ」が、現実的なものとして実感できるはずだ。

●Snortのダウンロードとインストール

SnortのWebページ（http://www.snort.org/）からSnortの最新版がダウンロードできる。ソースコードのほかに、RPMも用意されており、ライセンスはGPLだ。

面倒が嫌いな人なら、RPM版を“rpm -ihv ファイル名”でインストールしてしまえば、とりあえず使えるようになるが、起動できない場合はメッセージを参考に/etc/snort/snort.confの冒頭にある“var HOME_NET”と“var DNS_SERVERS”などの行を自分のネットワーク環境に合わせて書き換える。

●記録を見る

サイトに対する攻撃の疑いがあるパ

ケットのパターンが/etc/snort以下に保存されていて、このパターンに合致するパケットが検出されると/var/log/snort以下に記録される。

いろいろな記録が作られるが、とりあえずlogという名前のファイルに検出された攻撃が記録されるので、よくわからない場合はlogだけを読めばいい。

なお、初期設定のままだと、危険性の低いパターンも、とりあえず怪しいパターンはすべて検出、記録する。たとえば、Snort はpingやtracerouteなども、「不審なパケット」として記録する。確かに、pingやtracerouteは、ターゲットのネットワークに侵入しようとするとき、まず試してみるツールなので、これらを記録するのは正しい動作で、侵入の手口を知るときに役に立つかもしれないが、不要だと思えば、/etc/snort以下のファイルに書かれているルールを適当に修正してもかまわない。

また、Silicon Defenseが公開しているSnortSnarfというソフトウェアを使うと、Webブラウザを使ってsnortのログを閲覧できる（http://www.siliconddefense.com/snortsnarf/）（**画面1**）。

■ファイル改竄検知 Tripwire

Tripwireは、マシンのファイルが改竄されたことを検出するためのツールだ。最初、このソフトはフリーソフトウェアの形で公開され、あとで商用化されたのだが、昨年、商用版と同等の製品がGPLのフリーソフトウェアとして復活した。

●Tripwireのインストール

SourceForge（http://sourceforge.net/projects/tripwire/）から最新版のソースコードをダウンロードすることができるほか、オープンソース版TripwireのRPMパッケージがhttp://www.tripwire.com/products/linux/から入手できる。

なおTripwireのインストールは、RPMをインストールしたあとで、さらに“ /etc/tripwire/twinstall.sh ”を実行する必要がある。サイトとローカルのパスフレーズを入力すると、必要なファイルがインストールされる。最後に、/etc/twipwire/twcfg.txt、/etc/tripwire/twpol.txtをサイトに合わせて編集する。とりあえず試してみるだけなら特に編集の必要はない。

●実行

基本的な使い方は非常にシンプルだ。まずインストール後に、“ tripwire --init ”でデータベースを初期化する。これでファイルシステムのデータベースが作られる。このとき、“ No such file or directory ”のエラーが出るので、これを記録しておいて、/etc/tripwier/twpol.txtを編集するといい。

改竄をチェックするには“ tripwire --check ”を実行するだけでよい。

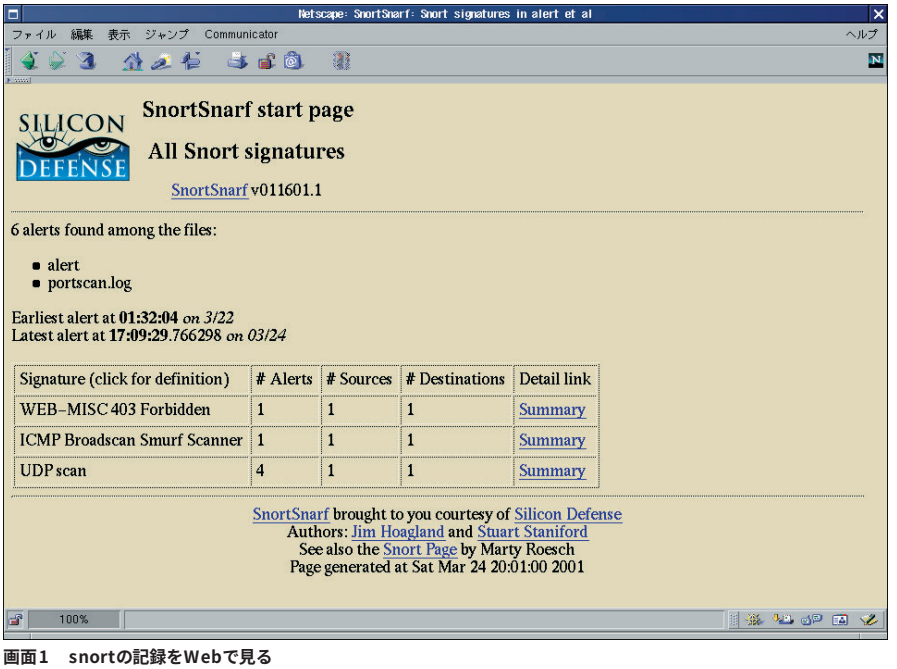
なお、初期化後にファイルのインストールや更新を行った場合、データベースのアップデートが必要になる。これには“ tripwire --update ”を実行する。

■ログ監視 swatch

膨大なサーバのログを毎日、丹念に読むほど、サーバの管理者は暇ではない。しかし、ログのチェックは欠かせない。そんなときは、swatchやlogsurferといったソフトウェアを使うといい。

これらのソフトウェアは、ログの中から重要なメッセージだけを拾い上げて、それが緊急なメッセージなら、メールやポケットベルなどにメッセージを送ってくれる。

swatch（http://www.stanford.edu/~atkins/swatch/）はシンプルで使いやすい監視ツールだが、logsurfer（http://www.cert.dfn.de/eng/logsurf/）をはじめ、さらに複雑な条件指定ができる監視ツールもある。



画面1　snortの記録をWebで見る
SnortSnarfを使うと、snortのログをWebインターフェイスで閲覧できる。

●swatchのインストール

Red Hatのディストリビューションには、powertoolsにswatchのRPMが入っている。そのほかにも各種のバージョンのRPMがWebで見つけられる。

swatchにはバージョン2と3があり、バージョン3のインストールにはいくつかのPerlモジュールが必要になるので、バージョン2のほうがインストールしやすいかもしれない。なお、2と3ではrcファイルの書式が異なっているが、基本的な動作は同じだ。

●実行

自分に必要なメッセージを拾えるように、サンプルのswatchrcをサイトに合わせて書き換え、/etc/swatchrcなどの名前で保存したあと、“ swatch --config-file＝ /etc/swatchrc --tail-file＝/var/log/messages ”などのコマンドで実行する。/var/log/messages以外にも監視したいログファイルがあれば、“ --tail-file＝監視対象ファイル ”を指定して必要な数だけswatchを起動すればよい。

swatchrcでは、監視するパターンと動作を指定する。たとえば、snortのログを監視して何かが検出されたら管理者にメールを送りたいときは、バージョン2なら次のように記述する。

```
/DETECTED/  mail=admin@my.host
```

バージョン3では、次のようにする。

```
watchfor  /DETECTED/
          mail=admin@my.host
```

■パスワードチェック

John the Ripper

パスワードの管理はセキュリティの基本だが、本当に十分に管理されているだろうか。UNIXのパスワードは、DESやrootのパーミッションで保護されているとしても、パスワードファイルが盗まれてしまうと、鉄壁のセキュリティもあっさりと破られてしまう。

John the Ripperは、さまざまなパスワードのパターンを試して暗号化されたパスワードを解読するツール。クラッカーにとっては必携ツールだが、サイトの管理者としては、管理パスワードのチェックや、弱いパスワードを見つけてユーザーに警告するといった用途に使える。

ただし、いうまでもないが、いくら管理者でも他人のパスワードを解読する作業はクラッキング行為である。ユーザーのパスワードの解読を試みる場合は、必ずユーザーに了解を取ってから行うべきだろう。また、作業中の情報の漏洩を防ぐために、作業は安全なマシンで行い、解読結果の書かれたファイルは確実に消去するなどの対策が求められる。

●John the Ripperのインストール

RPMはSuSEのパッケージなどがあるが、ソース（http://www.openwall.com/john/）からのコンパイルも難しくない。tarボールを展開して、srcディレクトリに入り、“ make linux-x86-any-elf ”とタイプすれば、たいていのLinuxでコンパイルできる。実行ファイルはrunディレクトリにできるので、runディレクトリを適当な場所にコピーする。

●実行

john.iniで、使用する辞書や解読ルールを指定できる。とりあえず試すだけなら、そのままでもかまわない。

“ john パスワードファイル ”のコマンドでパスワードの解読が行われ、画面に解読されたパスワードが表示される。また、結果はjohn.potというファイルに記録される。

ただし、シャドウパスワードのシステムでは、最初にunshadowを実行してターゲットのファイルを作る。たとえば、/etcディレクトリにpasswdとshadowがある場合、“ unshadow /etc/passwd /etc/shadow > passwd ”を実行する。

なお、パスワードファイルの解読はrootの権限で行う。

■バッファオーバーフロー対策

バッファオーバーフローとは、プログラムのメモリ管理の不備を突いて、実行中のサーバにシェルを実行するコードを送り込む攻撃手法の総称だ。

ソースコードのレベルでの問題なので、原則としてはプログラムのソースコードを修正、コンパイルしなおすことになるが、Stack Guard（http://immunix.org/stackguard.html）やLibsafe（http://www.avayalabs.com/project/libsafe/index.html）などを使ってコンパイルしたプログラムは、バッファオーバーフロー攻撃を検知して、被害を防ぐことができる。

このほかにも、Openwall Projectからカーネルへのセキュリティ強化パッチファイルが配布されている（http://www.openwall.com/linux/）。このパッチを適用したカーネルを使うと、バッファオーバーフローを起こしたプロセスが強制終了させられる。

ただし、一部のソフトウェアが動作しないなどの制限もある。また、2.4系カーネルへのパッチは、カーネルのバージョンが2.4.10くらいになるまでは提供されない見込みだ。

Column

良いパスワードと悪いパスワード

以下に列挙する特徴を持つパスワードは、パスワードクラッカー（パスワード破りプログラム）にかければ、ほぼ100%解読されてしまうので注意。次のようなパスワードは「悪いパスワードだ。」

・ありふれたパスワード
password、newpass、test、love、12345、secretなど、いかにもありふれたパスワードは厳禁。

・同じ文字だけのパスワード
たとえば、aaaaaaaa、99999999といったパスワードもありふれたパスワードの一種だ。

・簡単なパターン
キーボードの配列をそのままパスワードにしたqwerty、asdfなどのパターンや、12345678、abcdefghといったパターンもよくあるパスワードである。

・辞書に載っている単語を含むもの
パスワードクラッカーは、巨大な電子辞書を利用するので、この辞書に載っている単語で作られたパスワードは、一瞬で破られる。また、パスワードクラッカーは複数の単語の組み合わせや、大文字・小文字の組み合わせも試すため、単語を組み合わせただけでは、簡単に破られてしまう。

・誰かの名前を含むもの
パスワードクラッカーの辞書には、固有名詞も含まれているので、固有名詞で作られたパスワードも、必ず破られる。

・身近な人に関する情報を含むもの
パスワードのパターンに多いのは、若い母親なら子供の名前、恋愛中のカップルなら恋人の名前、ペットを飼っている人ならペットの名前、若い中高生なら好きな芸能人の名前、文学青年なら尊敬する文豪の名前、愛車家なら好きな車の名前や愛車のナンバーだという。身近な交遊範囲にクラッカーがいれば、パスワードクラッカーなどを持ち出す前に、まずこれらの固有名詞を試してみるだろう。これらの単語は「他人は知らないはず」の情報ではなく、「他人にも知られる可能性のある情報」だと考えるべきだ。

・単語に数字、記号などを付け加えたもの
パスワードには記号や数字を混ぜるとよいといわれるが、「myname!」とか「myname99」のように「単語＋記号」または「単語＋数字」の組み合せは、パスワードクラッカーのクラックパターンに登録されているため、やはり一瞬で破られてしまう。

・短いパスワード、小文字だけ、大文字だけ、あるいは数字だけのパスワード
8文字以上になると、いくら高性能のパスワードクラッカーでも破るのに何年もかかるが、6文字以下のパスワードや、数字だけのパスワード、大文字または小文字だけのパスワードなら、最近の高性能パソコンならそれほど時間をかけずに解いてしまう。パスワードの長さは最低でも8文字にするべきだ。

・ローマ字の日本語
ローマ字で表記された日本語はコンピュータにとっての情報量が少なく、容易に推測できる。26文字のアルファベットを8文字をでたらめに並べる場合の組み合わせの数は26の8乗（2088兆2706万4576）になるが、ローマ字表記の日本語はおよそ50の4乗（625万）程度の情報量しかない。

・文字の書き換え
数字の1と小文字のlを入れ替えたり、発音が似ているkとcの入れ替えといった、ありがちな書き換えは、やはりパスワードクラッカーの検査対象に組み込まれている。たとえば「loft」を「l0ft」にしたり、「cat」を「kat」に書き換えて「辞書にない」と思ってはいけない。

・単語の綴りを逆にしたもの
「peace」の綴りを逆にして「ecaep」というパスワードは人間の目には一見でたらめな文字の並びのように見えるが、よくあるパスワードのパターンのひとつ。当然、パスワードクラッカーの検査対象で、こうした機械的なごまかしは通用しない。

「良いパスワード」は、要するにランダムな文字列だ。パスワードとして許される有効な長さをいっぱいに使い、すべての文字種を含む（アルファベットの大文字、小文字、数字、記号）、無意味な文字列は、推測ができない。

ちなみに、覚えやすいパスワードの生成法として「歌詞の先頭の文字だけを組み合わせた文字列」が比較的よく使われているが、日本語の歌詞なら「悪いパスワード」に書いた「ローマ字の日本語」に相当する。また、英語の場合でも、単語の先頭に来やすい文字は限られているので、乱数としての情報量は少なくなるため、悪いパスワードとはいえないまでも良いパスワードともいえない。「良いパスワード」の作り方についてはいろいろな方法が公開されているが、公開されたパスワード生成の情報は、パスワードクラッカーに組み込まれている可能性が高いので、パスワードはランダムな文字列をお勧めする。

なお、無意味な文字列によるパスワードは「覚えにくい」ことが欠点だといわれる。だが、どんなに無意味なパスワードでも、UNIXの管理者なら毎日何度もパスワードを入力する機会があるので、1週間程度で完全に指が覚えてしまうはずだ。それに対して、パスワードの自動入力機能などは絶対に使うべきではない。マシンに記憶させた、つまりタイプする機会がないパスワードは、どんなに覚えやすいパスワードでも、必ず忘れてしまうからだ。

もし、パスワードを忘れてしまうことが心配なら、パスワードを紙に書き、厳重に封印して他人の手が届かないところにしまっておく。たとえば、透けないような厚紙に書いて、誰かに開けられたらわかるように糊付けし、しばらく定期入れにでも入れて持ち歩くといいだろう。毎日入力しているうちに必ず完全に覚えるので、その時点で封印したパスワードを焼却処分すればいい。

新世代64ビットCPU登場!

Itaniumマシン現る!

長らく噂が先行していたインテルの64ビットCPUが、いよいよ姿を見せつつある。強力なライバルが多数存在するハイエンドサーバ市場に、最後発として参入するIA-64アーキテクチャ。その最初の製品である、Itaniumを搭載したマシンの片鱗を紹介しよう。

文 : Linux magazineラボ
Text : Linux magazine Lab.
Photo : Junko Kitade (Pacia)

Itanium搭載マシン登場

コンピュータ業界の動向に興味を持つ本誌読者なら、“IA-64”や“Itanium”という言葉を聞いたことがあるだろう。IA-64は、'90年代半ばからインテルとヒューレットパッカード（HP）が共同で開発している、64ビットCPUのアーキテクチャの名称だ。IA-64をベースにした最初のCPUには、“Merced”というコードネームが与えられていたが、'99年に“Itanium”（アイテニアム）という製品名が発表された。

だが、それ以降なかなか具体的な製品が登場せず、本当に市販されるのか危ぶむ声さえ聞かれた。しかし実際には命名とほぼ同時期から、ハードウェアメーカーや一部の顧客に試作品が提供され、評価が進められていたのだ。IA-64はまったく新しいアーキテクチャであり、今までの実績があるx86系CPUよりも慎重かつ確実に評価を行う必要があるため、準備期間がこのように長期にわたってしまったのも当然といえるだろう。

今回、三菱電機製のパイロットリリース用Itanium搭載サーバを使用する機会を得た。完成版の製品ではないため、パフォーマンス測定などは行えなかったが、ハード／ソフトの両面からIA-64に迫ってみたい。

写真1　Itanium搭載マシンの外観
奥行き方向に長い大型筐体を採用している。

重装備のマザーボード

今回借用したItanium搭載サーバは、後ろに長い巨大なタワー型ケースを採用している（**写真1**）。わずかな変更で、高さ4Uのラックマウント仕様としても使用できるものだ。

マザーボードは、三菱電機オリジナルのもので、一般的なATXとは上下反対になっており、CPUが下部に、拡張スロットが上部に配置されている（**写真2**）。後述するようにCPU周りの部品は重量級なので、このような配置になっているのだろう。

チップセットは、インテルの460GXを採用している。460GXは11個のチップで構成されており、1～4 CPUに対応している。メモリは通常のSDRAM（ECC付きPC133）を使用可能で、マザーボードと専用のコネクタを介して接続された、ドーターボード上のDIMMスロットに装着するようになっ

CPU	Itanium 733MHz
メモリ	512MバイトSDRAM（ECC付きPC133、最大8Gバイト）
ハードディスク	Seagate ST318404LC（Ultra160 SCSI、18Gバイト）
リムーバブルドライブ	CD-ROM、SUPERディスク
グラフィックス	ATI RAGE 128 GL
ネットワーク	インテル PRO100＋
サイズ（mm）、重量	195（W）×650（D）×460（H）、約35kg
電源	500W×1（最大500W×3）

表1　Itanium搭載サーバの仕様

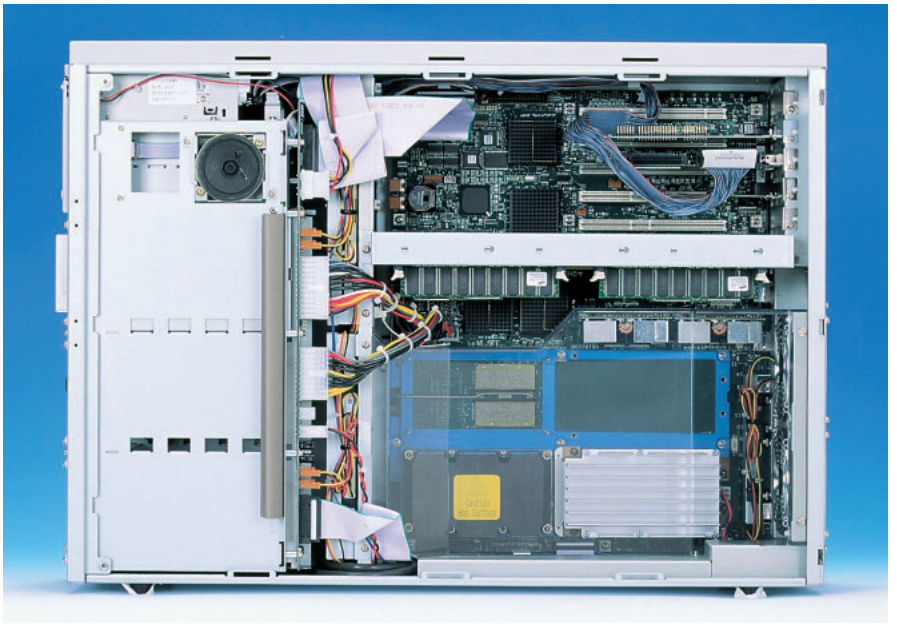


写真2　内部の構造
CPU周辺は、透明なダクトで覆われている。このまま展示できるような美しさだ。

ている。標準で512Mバイト、最大で8Gバイトまでメモリ増設が可能だ。

拡張スロットは、64ビットPCIを5本持ち、うち3本は66MHzに対応している。今回のマシンには、グラフィックスカードのATI RAGE 128 GL、ネットワークカードのインテル PRO/100＋、そして64ビットPCI対応のUltra160 SCSIカードであるアダプテック 29160が装備されていた。

ハードディスクは、Seagate社のCheetahシリーズが採用されている。前部にドライブベイが2台分設けられており、SCAコネクタを用いて、簡単にディスクの設置ができるようになっている。またホットスワップにも対応している。そのほかリムーバブルデバイスとして、CD-ROMドライブに加え、通常のフロッピーディスクも利用できるSUPERディスクドライブが装備されている。IA-64のコードサイズがx86と比較して大きいため、フロッピーディスクでは起動用ディスクの作成が困難なためと思われる。

PCサーバ市場では後発組の三菱電機だが、その分細部までていねいに作り込まれた製品に仕上がっている。

独特なCPU形状

Itaniumの製品版は、733または800MHzになると言われている。約2500万のトランジスタを集積したコアのほかに、3次キャッシュが必要になるため、CPU本体は、ケースに入れられている（**写真4**）。その形状は、Pentium IIなどで用いられたSlot 1などとも異なる独特のものだ。Slot 1のCPUは「ファミコンカセット」とも称されていたが、Itaniumの本体は「小型ハードディスク」にたとえたくなるような形状と重さだ。

裏面に418本のピンがあり、マザーボード上のソケットに設置する。なお、Pentium Ⅲ／4などは、電力も底面のピンから供給されるが、Itaniumに必要な電力は膨大であり、ピンからでは供給できない。代わりに、エッジタイプのコネクタを側面に持ち、CPUのすぐ横に設置されている電源（パワーポッド）から供給するようになっている（**写真3**）。

ヒートシンクの大きさから想像できるように、CPUとパワーポッドの消費電力と発熱量は、Pentium 4などと比較しても大きく、CPU単体で100Wを超えると言われている。本機には500Wの電源が1台搭載されているが、デュアルCPU構成にするためには、もう1台の電源が必要だ。筐体前部の電源スペースには、最大で3台まで収納可能で、デュアルCPU＋リダンダントという構成にもできる。

CPUとパワーポッドの排熱機構は、ほかの機種では見られない独特のものだ。写真2は少々見づらいが、CPUの上にアクリル製の透明なダクトが設置されていることがわかる。ダクトの前後に吸気×2、排気×2の巨大なファンを設け、強制的に気流を作り出している。運転時にダクトを外すことはできないので、確認はできなかったが、発熱量は相当多いと思われる。

IA-64 Linux

IA-64アーキテクチャ用のOSとして、Windows XP（Windows 2000の後継）、IBMのAIX、HPのHP-UXなどが移植されている。Linuxも'99年5月に発足したTrillian Projectのもとで移植が開始されている。同プロジェクトは、現在ではIA-64 Linux Project（**画面**

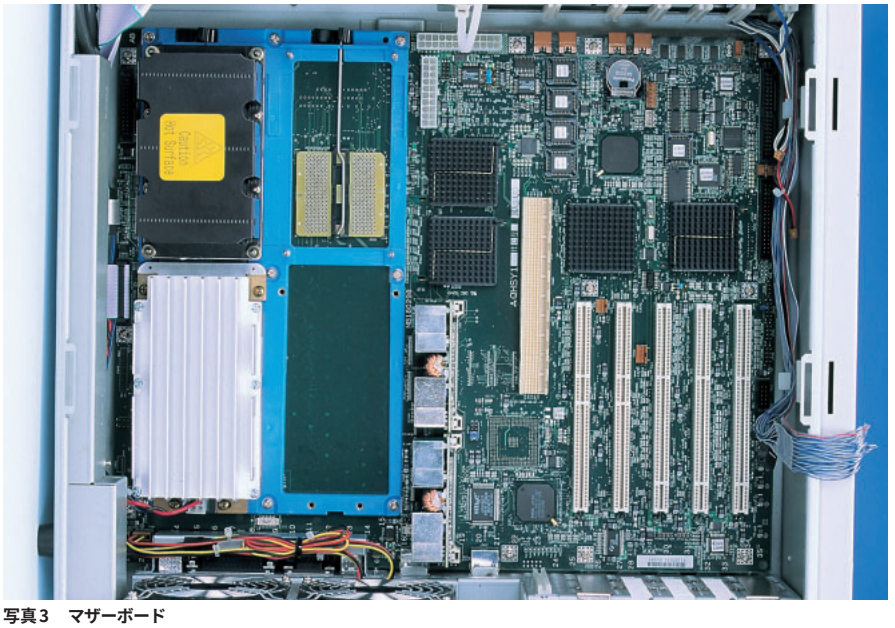


写真3　マザーボード
デュアルCPUに対応し、64ビットPCIスロットを5本持つ。左上がCPU、その下がパワーポッド。

写真4　Itaniumの裏面
Slot 1のCPUとは異なる構造のケースに収められている。

1）と改名し、移植作業が続けられている。

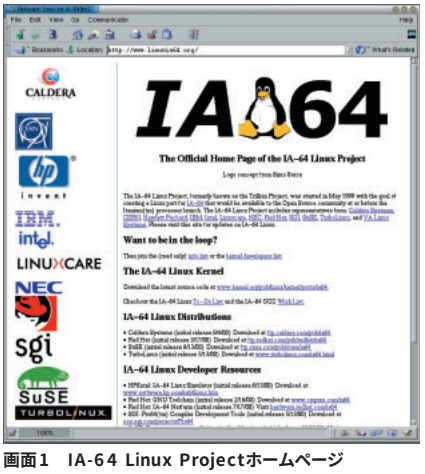
Windows XPは、x86版でさえ年内にリリースできるかどうかというスケジュールだ。また、そのほかの商用OSについても、あまり話が伝わってこない状態だ。IA-64 Linuxは、Itaniumシステムがスタートダッシュを決められるかどうかに、かなり重要な影響をおよぼすことになるだろう。

カーネル2.4をベースにしたIA-64 Linuxは、Red Hat、Turbolinux、Caldera Systems、そしてSuSEの各社からリリースされている。三菱電機から提供されたマシンには、Red HatのFisherとTurbolinux ベータ3がインストールされていたので、今回はこの2つのディストリビューションを試用することとした。

どちらのディストリビューションもベータ版であり、最終バージョンとは異なる可能性があることをお断りしておく。

■起動はEFIから

x86版Linuxでは、BIOSがブートセクタに書き込まれているLILO（LInux LOader）を読み込んで起動するのが一般的だが、LILOはx86 PCのBIOSに強く依存したプログラムであるため、



画面1　IA-64 Linux Projectホームページ

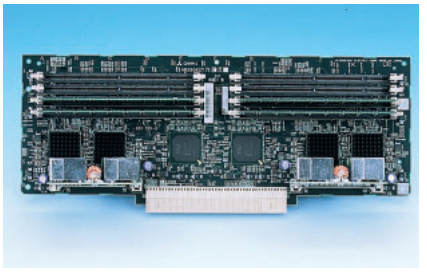


写真5　メモリ用ドーターボード
DIMMスロットを8本持ち、最大8Gバイト（ECC付きPC133）まで増設が可能。

ほかの環境では利用できない。

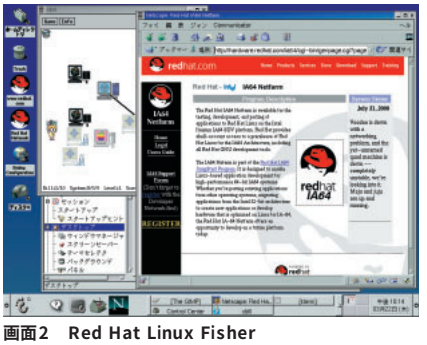
IA-64のシステムでは、BIOSに代わるExtensible Firmware Interface（EFI）と呼ばれる仕組みが用意されている。EFIはOSに依存しないため、IA-64 LinuxもEFIから直接起動できる。EFIについては、インテルの小池氏による解説（100ページから）があるので、そちらを参照していただきたい。

■Red Hat Linux Fisher

Red HatのIA-64 Linuxは、ベータ版のFisherである。x86に依存したプログラム以外は、4月号で紹介したx86版のFisherと同じだ（**画面2**）。カーネルは2.4.0.99、Cライブラリはglibc 2.2.1、X Window Systemは、XFree86 4.0.2を採用している。Xが起動してしまえば、x86版との違いを見つけるほうが難しいくらいだ。

■Turbolinux ベータ3

Turbolinuxは、IA-64 Linuxで初め



画面2　Red Hat Linux Fisher

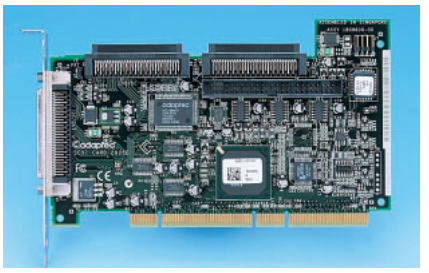


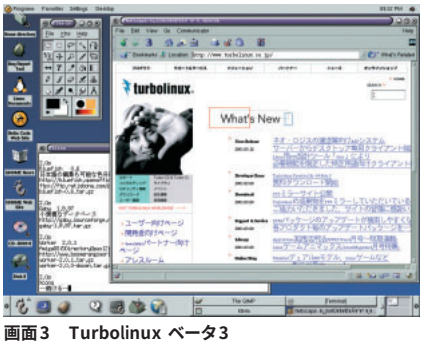
写真6　アダプテック 29160
Ultra 160SCSIに対応した64ビットPCIカード。BIOSは通常のx86版と同じ。

てカーネル2.4系列を採用した。カーネルは2.4.1、Cライブラリはglibc 2.2、X Window Systemは、XFree86 4.0.2を採用している。デスクトップ環境であるGNOMEは、1.2をベースにXimian（旧Helix Code）社がカスタマイズしたものを用いており、同時に標準のデスクトップテーマも新しくなっているので、見た目が新鮮に感じる。

■実力発揮はこれから？

どちらのディストリビューションも、ベータ版なので細かな問題点は見受けられるものの、特に不安定な要素はなかった。

また、速度の面でもふだん使っているx86版と同等で、ストレスを感じることもない。特に浮動小数点演算は、現段階でも相当な実力があるようだ。編集部でも試しにGIMPのフィルター機能を用いて、画像ファイルの処理を行ってみたが、フィルターの種類によっては、常用しているx86 PCを上回る処理理速度を示した。



画面3　Turbolinux ベータ3

Itaniumって何？

21世紀早々、CPUの世界に大きな変動が訪れようとしている。その震源地はインテル、言わずと知れた世界最大の半導体メーカーだ。同社が開発したx86アーキテクチャのCPUは、PCの心臓部として世界中に存在している。x86アーキテクチャのCPUを製造するメーカーはインテル以外にもあるが、それらのシェアは小さく、PCにそれほど詳しくない人から見れば、「CPUはインテル製」というイメージがある。

IA-64の衝撃

'90年台半ばに、インテルはヒューレットパッカード（HP）と共同で、x86に代わる新アーキテクチャのCPU、IA-64の開発を行うことを発表した。

IA-64に基づいた最初の製品は、Itanium（コード名“Merced”）と命名され、多少スケジュールは遅れたものの、昨年10月からItaniumを搭載したシステムが、限定された顧客へ提供（パイロットリリース）され始めている。間近に迫った正式リリースの前に、新しいCPUの実態に迫ってみよう。

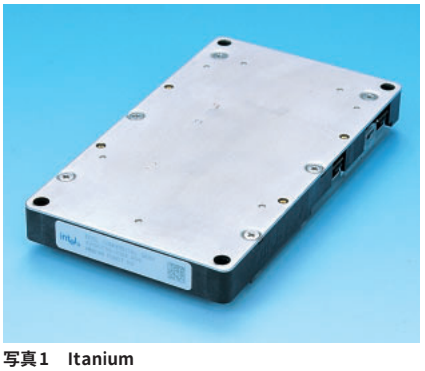


写真1　Itanium
残念ながら開けることはできなかったが、これが本物のItaniumだ。

IA-64の命令セットは、x86（IA-32）の命令セットとは互換性を持たない、完全に新しいものだ。もっともItaniumは、x86用のコードも実行可能なように設計されているが、Itaniumの性能を完全に発揮するためには、IA-64ネイティブのコードが必須だ。Itanium上でIA-32のコードを実行した場合、同じクロックのIA-32 CPUよりも性能が低くなることは、インテルも認めている。

新しい高性能なCPUが市場に受け入れられるためには、旧製品との互換性を持つことが必要だ。PCを買い替える時に、ソフトも買い直さなければならないとしたら、PCの買い替え需要ははるかに少ないだろう。それを最もよく心得ているのは、他ならぬインテルだろう。32ビットCPUだけに限っても、i386以降、i486、Pentium、Pentium Pro／II／III、Pentium 4までコードの互換性を保ってきたからこそ、かくも長期にわたって市場で勝利し続けられたのだから。IA-32からIA-64への移行は、インテルにとっても大英断だったに違いない。

CPUの速さは十分か？

そもそもなぜIA-64という新しいアーキテクチャが必要なのか？ それはCPUに求められる性能が、今後も上昇していくのは確実であり、16ビットCPUの頃からのしがらみを持つx86 CPUの改良では、性能の向上に限界があるからだ。

メーカー間の激しい競争により、ここ数年でPCの部品は、性能向上と価格低下が同時に進行した。その結果、ネットワークのクライアントPCや家庭内で使うPCに関しては、ローエンドのモデルでも十分な性能を持つようになったと言える。

それではこれ以上CPUを速くする必要はないのか？ もちろん答えはノーだ。インターネットが全世界に急激に普及しつつある今、ネットワークに流れるデータ量も増加する一方だ。その

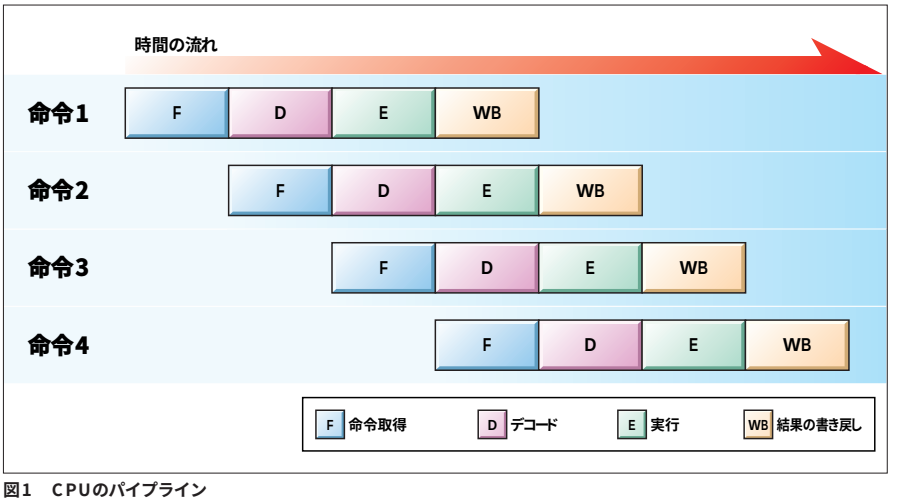


図1　CPUのパイプライン
命令処理を数ステップに分割し、命令を並列実行するようになっている。

データを作り出し、コントロールするためのサーバにも、よりパワフルなCPUが要求される。

またクライアントPCが十分な性能を持つといっても、それは現在用いているOSやソフトウェアを前提にした話でしかない。リアルタイムの翻訳機能や、音声認識によるPCの操作など、現在のCPUではちょっと無理そうなことでもこれからCPUが進化していけば、実現できるかもしれないのだ。

今まで常により速いCPUが求められ続けてきたのは、古いCPUでは処理しきれないほど「重い」ソフトウェアが登場し続けてきたからだ。そのソフトが魅力的ならば、人々は新しいCPUを買い求める。確かにここ1年ほどは、ソフトの重さにハードが追い付いた感があるが、きっとまた魅力的で重いソフトが現れるだろう。まだまだCPUは速くなる必要があるのだ。

CPUの高速化

CPUを速くするには、大きく分けると2通りのやり方がある。ひとつは、

(1) 命令の処理時間を短くする

ことであり、もうひとつは、

(2) 多くの命令を同時に処理する

ことである。(1)は、CPUのクロックを上げることである。この記事の作成時には、x86系CPUのクロックは、1.5GHz(Pentium 4)に達している。ただ現状では、CPUのクロック上昇にメモリなど周辺装置の速度が追いつかないため、CPUが待たされることになり、クロックを上げても実効性能はそれほど上がらなくなっている。

そこで高クロック化と同時に、(2)の複数の命令を同時に処理(並列処理)することも多くのCPUで採用されている。

CPUは高速化のために、命令の実行を複数のステップに分割して行っている。このステップ全体のことを、「パイプライン」という。**図1**は基本的なパイプラインと、命令の処理のされ方を示したものだ。Pentium以降のx86 CPUでは、パイプラインの各ステップ(**図1**のF、D、E、W)を複数用意することで、1クロックサイクルに複数の命令を実行する。この方式をスーパースカラーという。

しかし一般的なプログラムコードは、並列処理を前提としていない。そのため、依存性のある複数の命令を同時に実行してしまうと、正しい結果が得られない。そこでスーパースカラーアーキテクチャを採用したCPUは、実行時にプログラム内の命令群を調べ、同時に実行可能な命令を抽出し、時には実行の順序を入れ替えるなどして、なるべく多くの命令を同時に実行できるように工夫をしている。それでも1クロックあたりの平均実行命令数は、多くの場合2を少し下回る程度だと言われている。

VLIW

このように、まず命令群を読み込んだあとに、依存関係を調べるスーパースカラーは、ハードウェアの負担が非常に大きくなる。より高度な並列化を行おうとすればするほど、ハードウェアは複雑化し設計も困難になっていく。

そこで、プログラムを作成する時にソフトウェア(コンパイラ)でできるだけ並列処理できるようなコードを出力させ、CPUはそのコードを単に実行するだけというアプローチが考え出された。この方式をVLIW(Very Long Instruction Word)という(**図2**)。並列実行できる命令をまとめて、とても長い(Very Long)命令にまとめるため、この名前が付いている。まとめられた命令は、同時に実行できることがわかっているので、VLIW方式のCPUでは、長い命令を読み込んだら内

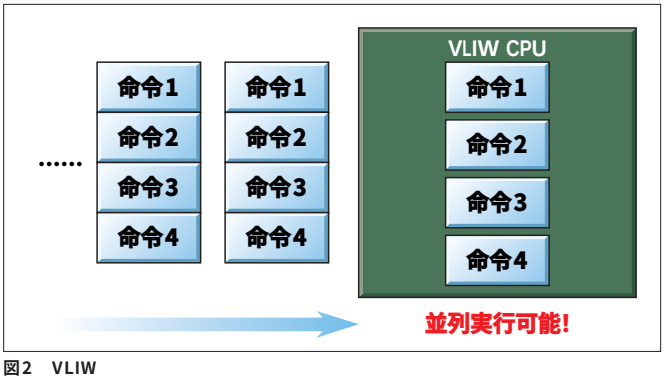


図2 VLIW
ひとまとめにされた命令群は、同時実行可能だ。

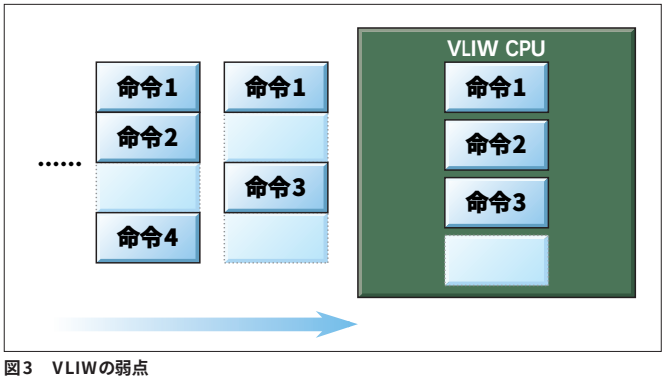


図3 VLIWの弱点
コンパイラの性能が低いと、性能が上がらないうえに、プログラムサイズが大きくなってしまう。

部の命令をそのまま処理すればよい。スーパースカラーと比較すると、ハードウェア（CPU）は非常に単純なものになる。

余談だが、このVLIW方式は低消費電力CPUとして知られているTransmetaのCrusoeにも利用されている。

スーパースカラーに対するアンチテーゼとして、VLIWが考え出されたように書いてしまったが、実際にはVLIWはスーパースカラーよりも前から研究されていた方式である。そんな前から研究されていたのに、なぜVLIW方式のCPUは、市場に現れなかったのか？ 原因のひとつは、VLIWがコンパイラの性能に非常に依存することだ。コンパイラの性能が低いと、**図3**のように命令で満たされていない長い命令が読み込まれることになる。その場合、空いたところには、「何もしない」命令が書き込まれるため、性能が上がらないばかりか、プログラムのサイズが不必要に大きくなってしまうことになる。

またハードウェアが進歩して、同時に実行できる命令数が増えた場合、コードの互換性がなくなってしまうという問題点もある。CPUを買い替えるたびにプログラムも再コンパイルするようでは、マシンのアップグレードなどできないだろう。そのほか、x86と互換性がないこともあり、今までVLIW方式のCPUが主流になることはなかったのだ。

EPIC

上記のいくつかの欠点のために表舞台には出てこなかったVLIWだが、インテルはこれらの欠点を克服し、VLIWをベースにしたEPICというデザインを生み出して、IA-64アーキテクチャに採用した。EPICは“Explicitly Parallel Instruction Computing”の略であり、直訳すれば「明示的並列命令コンピューティング」となる。

IA-64の命令は、6種類のタイプに分類されており、それぞれ処理される実行ユニットが決められている（表1）。たとえば命令タイプAに分類される、単純な加減算の命令は、IユニットまたはMユニットで実行可能だ。

命令タイプ	説明	実行ユニットのタイプ
A	比較的簡単な整数演算	IユニットまたはMユニット
I	整数演算	Iユニット
M	メモリ処理	Mユニット
F	浮動小数点演算	Fユニット
B	分岐処理	Bユニット
L+X	拡張演算	IユニットまたはBユニット

表1　命令タイプと実行ユニットの関係

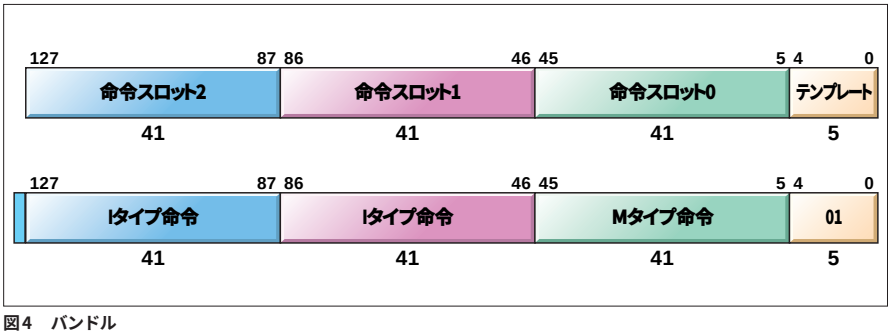


図4　バンドル
128ビットのバンドル内には、3個の命令と、テンプレートが含まれる。

各命令は、個別に扱われるのではなく、3個ずつ「バンドル」と呼ばれる「長い命令」にまとめられている（**図4**）。命令の長さは41ビットであるのに対し、バンドルは128ビットの長さを持つ。残りの5ビットは「テンプレート」と呼ばれ、バンドルに含まれている3つの命令の、

- **実行ユニット**
- **同時実行の可否**

が記述されている。**図4下**のバンドルを例にとると、命令0はMユニット、命令1、2はIユニットで処理すればよいことがわかる。また命令2の後ろに二重線が描かれているが、これは後続のバンドル内の命令とは同時実行ができないことを表している。この同時実行が可能かどうか示す境界は、「ストップ」と呼ばれる。

このようにEPICでは、並列実行できる命令群を「長い命令」の内部に限らず、ストップで「明示」することにしたため、VLIWのような不必要なコードの膨張や、CPU世代間の非互換といった問題を解決している。また、各命令をどの実行ユニットで処理するかは、テンプレートを見るだけで簡単に判別できるため、命令の解釈（デコード）も短時間で済むようになっている。

速くするための工夫

このように並列性を明示する仕組みを持つことで、多くの命令を同時実行できる可能性が得られた。しかしこれは、単にハードウェア的に対応できるということであり、実際に性能が上がるかどうかは、いかにプログラムから並列処理できる命令群を抽出できるかにかかっている。そのための工夫は非

常に多岐にわたり、ここで紹介しきれるものではないが、代表的なものを取り上げてみよう。

■大量のハードウェアリソース

初代のItaniumは、1クロックに2つのバンドルを同時に発行できるので、最大6つの命令が同時に実行されることになるが、それに対して4つのメモリ／整数演算ユニット、2つの浮動小数点ユニット、3つの分岐ユニットと十分な実行ユニットを備えている。

また、CPU内でデータを保持するために用いられるレジスタも64ビット長の整数レジスタ、82ビット長の浮動小数点レジスタをそれぞれ128本持っている。

■スペキュレーション

日本語では「投機実行」と訳されるスペキュレーションは、「多分こうなるだろう」と予想して、命令を実行してしまい、あとでその予想が正しいかどうか確認する方式だ。確認のための命令が増えてしまうので、かえって遅くなりそうだが、ある種の命令に対してはとても有効な方法だ。

現在は、メインメモリの速度がCPUの速度よりもはるかに遅いため、メモリからデータをロードする命令のすぐうしろにそのデータを利用する命令を置くと、この命令はロードが済むまで待たされることになる。データがキャッシュ上になく、メインメモリから読み込まなければならない場合の待ち時間は、非常に大きくなる。これを防ぐために、ロード命令を前方に移動して先に実行してしまうのが、スペキュレーションの一例だ。

以下の例は、インテルのソフトウェアデベロッパーズマニュアルからの引用である。

```
store(st_addr, data)
load(ld_addr, target)
use(target)
```

データをメモリに書き込むstore命令のアドレスst_addrと、データをメモリから読み込むload命令のld_addrが同じかどうかは、実行時までわからない。このようなプログラムがあった場合、IA-64では、次のように命令を並び替える。

```
aload(ld_addr, target)
：
：
store(st_addr, data)
acheck(target, recovery_addr)
use(target)
```

このようにスペキュレーション用のロード命令を先に行い、もともとロード命令があった場所では、整合性のチェックだけを行う。たとえばstore命令とaload命令のアドレスが同じだった場合には、aload命令で得られた値は正しくないので、再度実行することになる。

やり直しの回数が多くては性能が上がらないので、相対的にやり直す確率が低く、ロード命令を移動することで、並列度が上がると判断された場合に、このような並び替えが行われる。

■プレディケーション

プレディケーションとは、条件付きの命令実行を意味する。言葉だけ聞いてもよくわからないので、これも実際のコードを参照しよう。

```
if (r1)
  r2 = r3 + r4;
else
  r7 = r6 - r5;
```

これは、r1が0でなければ第2行を、0だったら第4行を実行するというプログラムだが、プレディケーションを用いて、IA-64のアセンブラコードに変換すると、以下のようになる。

```
cmp.ne   p1,p2=r1,0;;
(p1)add  r2=r3,r4
(p2)sub  r7=r6,r5
```

まず、r1と0の比較を行い、その結果をプレディケーションレジスタp1、p2に格納する。アセンブラコードの第2行は、「p1の値が真ならr3とr4を足した結果をr2に入れる」ことを表す。第3行も同様だ。なお、第1行の比較の結果が2、3行で用いられているため、第1行と2、3行を同時に実行することはできない。2つのセミコロンは、ここにストップが入ることを示している。

このようにしても行数が減っているわけでもなく、あまり効果がないように思えるかもしれないが、上記のアセンブラコードには分岐命令がないため、パイプラインを乱すことなく実行できるので、非常に有効だ。

高速化の鍵はコンパイラ

上記のプレディケーションやスペキュレーションをはじめとした、IA-64に盛り込まれている高速化のための手法をフルに利用するには、非常に賢いコンパイラが必要だ。

Itaniumの正式リリースを間近にした現在でも、コンパイラの改良は続けられているという。IA-64が高性能だと認められ、普及していくかどうかは、各社がどれだけ優秀なコンパイラを提供できるかにかかっているといっていいだろう。

Itanium Ready Extensible Firmware Interface

文：小池浩之（インテル株式会社） Text : *Hiroyuki Koike*

「Extensible Firmware Interface（以下「EFI」と表記）」は、オペレーティングシステム（OS）とファームウェア間の新しいインターフェイスです。EFIは、プラットフォームに関する情報を集めたデータテーブルと、OSやOSローダから呼び出せるブートサービスおよびランタイムサービスから構成されます。この2つを組み合わせることで、OSをブートするための標準的な環境を提供します（**図1**）。

「PC-AT」型の問題点

従来のいわゆる「PC-AT」型のブート環境では、新しいハードウェアが開発されるたびに、ファームウェア開発者は非常に複雑な問題を考慮しなければなりません。また新しいハードウェア技術を活かすために、OS側でブートコードを修正する必要がある場合もあります。これは多くの投資と時間を必要とする作業です。

この問題を解決するための新たなブート環境がEFIです。EFIは以下のような機能を提供します。

■一貫性のある、拡張可能な環境

EFIは、ファームウェア内でプラットフォームの機能をすべて記述することで、その能力をブート処理中のOSからも十分に活用できるようにすることができます。その適用範囲は非常に広く、最新のインテルアーキテクチャによるあらゆるシステムの設計に適用可能です。

■OSの抽象化

EFIはプラットフォームの能力を利用するための、標準的なインターフェイスを規定します。抽象化されたインターフェイスを使用することで、プラットフォームやファームウェアの細かな違いを意識することなく、OSローダを開発することが可能になります。

これは、プラットフォームやファームウェアの実装と、OSローダとの境界を明確に定義することで実現されます。このインターフェイスだけを使って実装されていれば、ファームウェアやOSローダを独立に変更しても、問題なく動作することになります。

■デバイスの抽象化

従来の「PC-AT」のBIOSインターフェイスを使うと、OSローダは個々のハードウェアデバイスの動作を考慮する必要があります。これがOSローダの開発を困難にしていました。

抽象化されたインターフェイスを使用することで、デバイスについて特に意識することなく、さまざまなプラットフォーム上で正しく動作するコードの開発が可能になります（**図2**）。

■システムパーティションの定義

プラットフォームの性能を向上させ、新しいデバイスを追加する際には、ソフトウェアのサポートが欠かせません。OSが対応する前にプラットフォームに新しい技術が導入されると、それを利用するためにはプラットフォーム固有のコードで対応しなければならないことがままあります。従来はフラッシュメモリなどを使い、プラットフォームの製造段階でコードを組み込むことでこれらに対応していました。EFIでは、これらのコードを大容量記憶装置に記録する方法を定義することで、ファームウェア開発者やOSベンダー、サードパーティが、記憶装置上の空間を安全に共有できるようにします。

基本設計

上記の機能を実現するため、EFIの各部分は、以下のように設計されています。

■ブートサービス

デバイスやシステムの機能を使うためのインターフェイスで、ブート時に使用されます。デバイスへのアクセスは、「ハンドル」および「プロトコル」の概念を使って抽象化されています。この機能は既存のBIOSコードを再利用していますが、個々のデバイスについ

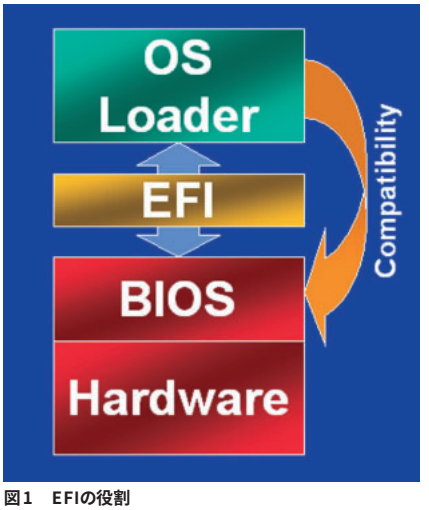


図1　EFIの役割
OSローダとBIOSのインターフェイスが標準化される。

て意識することなくブート処理を実装可能になっています。

■ランタイムサービス

OS起動後に利用するハードウェア資源を適切に抽象化した、最小限のランタイムサービスを提供しています。

■システムパーティション

各OSが、さまざまな目的で共有可能なシステムパーティションを独立して用意することで、フラッシュメモリなどの不揮発性の記憶領域を用いずに、プラットフォームに付加価値を組み込むことが可能になります。

そのほか、これまでのOSやファームウェアのコード開発にかけた投資を無駄にしないため、Intelアーキテクチャのプラットフォームで一般に実装されているACPI、SMBIOSなどはEFIに準拠するプラットフォームでも実装されています。

図3にEFIを構成する主な要素と、ハードウェアやOSとの関連を示します。ファームウェアは、EFIシステムパーティションからOSローダのイメージを検出し、起動することができます。EFIには、ハードディスク、CD-ROM、DVDのほか、ネットワークブートを含む、さまざま記憶デバイスのタイプが規定されています。また、拡張可能なプロトコルインターフェイスを使用して、上記以外のブートメディアのタイプを追加することも可能です。

ブート処理

EFIを使うと、OSローダやファームウェアが別々に用意していたブートメニューを、単一のメニューに統合することができます。EFIブートサービスが対応していれば、メディアやパーティションを問わず、OSローダを選択できます。

またEFI OSローダでは、ユーザーがさまざまなオプションを選択できます。たとえば従来のように、A:ドライブやC:ドライブからのブート方法も、ファームウェアメニューに表示させられます。市販のマルチOSブートセレクタなどのツールを使用する必要はありません。

EFIに含まれるブートマネージャを利用して、EFIで定義されたファイルシステム上の任意のファイルから、あるいはEFIで定義されたイメージローディングサービスを使用して、EFIアプリケーションやEFIドライバをロードできます。

現在のPC-ATクラスのシステムでは、基本的にフロッピーディスク、ハードディスク、CD-ROM、ネットワークカードからしかブートできません。1台のハードディスクから複数のOSをブートさせることも可能ですが、設定には熟練が必要です。EFIによって、PC-ATクラスのシステムでは、ブート処理の柔軟性が大きく拡大します。

EFI シェル

EFIで定義されたAPIや、データ構造を利用したソフトウェアは、デバイスドライバやシステムユーティリティ、あるいはOSローダといったさまざまな形態をとりますが、そのうちのひとつにEFIシェルがあります。EFIシェルは他のEFIアプリケーションを呼び出すことのできる特殊なEFIアプリケーションと言えます。

EFIシェルは、非常にシンプルな対話型環境で、この上でEFIデバイスドライバのロードやEFIアプリケーションの実行、あるいはOSのブートを行うことが可能です（**画面1**）。

EFIにより、特定のOSやデバイスから解放されたソフトウェア／ハードウェア開発が可能になり、コスト削減と高い安定度の両立が可能になります。

まずItaniumプロセッサファミリーシステムから採用されるEFIですが、将来的にはIA-32システムもすべてEFIに移行する予定です。既存のリソースを活かしながら、より扱いやすい環境を提供できるEFIは、次世代システムのキーテクノロジーなのです。

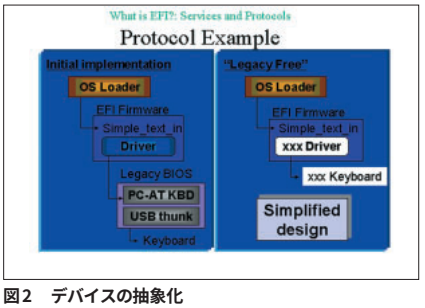


図2　デバイスの抽象化
OSローダは、個別のハードウェアを意識する必要がない。

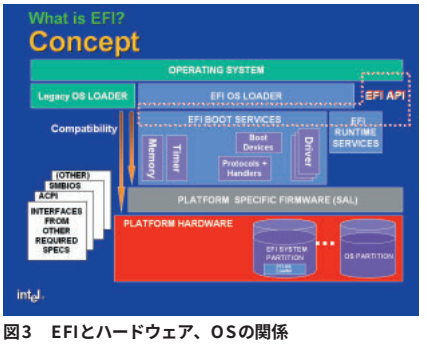


図3　EFIとハードウェア、OSの関係
過去の資産を継承しつつ、OSに依存しないブート環境を実現できる。

画面1　EFI シェル
MS-DOSやUNIXのシェルに似たシンプルなインターフェイス。

短期集中連載

Linuxでテレビ録画に挑戦！

1万円で作るテレビ自動録画サーバ

第2回　バッチ処理で無人録画する

前回は、比較的安価に入手できるTVキャプチャカードをLinuxで使う方法について説明した。Linuxのキャプチャカードドライバは、Windows用のドライバよりも安定しており、予想以上に快適な視聴環境を構築できるはずだ。今回はさらに進んで、無人でテレビ録画を行う方法について説明しよう。なお、今回作成したリストは長いため、ほとんどが一部しか掲載できなかった。あらかじめ本誌ホームページ（http://www.ascii.co.jp/linuxmag/）からダウンロードしていただきたい。

文：竹田善太郎
Text : Zentaro Takeda

前回説明した「Xawtv」というプログラムは、見た目は質素だが、Windows用のテレビソフトに見劣りしない機能を持っている。さらに、アプリケーションの安定性などの面では、キャプチャカードに付属するWindows用の（正体の知れない）ソフトに比べて格段に優れているので、これだけでもLinuxでTVキャプチャカードを導入する意味はある。しかし、Linuxが優れているのはこれだけではない。Windowsが苦手とする「無人運転」、すなわちcron機能を活用すれば、無人での予約録画なども、簡単なコマンド操作でできるようになるのだ。

現時点では、録画されるビデオデータのファイルサイズがかなり巨大なため、数時間分もの番組を予約することは難しいが、次回以降に説明する、ビデオデータの圧縮を併用すれば、数週間分の番組などを予約録画することができるようになる。とりあえず今回は、できるだけ手軽にテレビ放送の予約録画機能を実現するため、xawtvに付属する「streamer」というコマンドの使い方を中心に解説しよう。

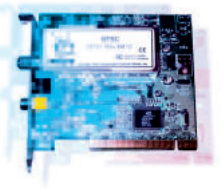
streamerコマンドの仕事

xawtvをコンパイルしてインストールすると、/usr/local/binディレクトリに、「streamer」という名前のコマンドファイルが同時にインストールされる。これが今回説明するstreamerコマンドの本体である。

streamerコマンドは、xawtvのプログラムからX画面へのテレビ画面の表示機能を取り去ったようなもので、チューナーの制御から動画データのキャプチャまで、TVキャプチャカードに関連するほとんどの処理を行うことができる（**図1**）。streamerコマンドは、コマンドラインから各種のオプションをつけて起動するだけで、音声と映像のキャプチャを行わせることができるので、無人録画にはまさにうってつけのツールである。

ただし、streamerコマンドでは、xawtv本体の場合と同様に、動画をmpeg形式などに圧縮してキャプチャすることができない。簡便な動画圧縮である「motion jpeg」には対応しているが、それでも1分あたり約19Mバイト、30分で約600Mバイトというかなり大きなファイルになってしまう。とはいえ、20〜30GバイトのHDDがあたりまえの現在では、たとえばNHK教育テレビの語学番組（1回30分）を数回分保存しておくような用途なら、ある程度実用に耐えると思う。

キャプチャした動画データを再圧縮したり、あるいはキャプチャを行うときにリアルタイムに圧縮するプログラムを使



うことで、このファイルサイズをさらに半分以下に圧縮することもできるが、この方法については多少複雑になるので、次回以降に改めて説明する。

streamerコマンドの基本的な使い方

streamerコマンドのオプション一覧と、それぞれの設定項目の詳細を表1に挙げておく。オプション一覧は--helpオプションをつけてstreamerコマンドを起動しても表示できる（ただし、表示は英語になる）。

オプション一覧を見ればわかると思うが、streamerではキャプチャする動画データの形式や音声の形式などを、きわめて細かく設定できる。キャプチャするとき、どのような形式を指定すればよいかは、使用するマシンの処理性能や、キャプチャ済みの画像をどのように利用するかによって変わってくるため、一概に決めることはできない。

ここでは、12fps（フレーム／秒）、320×240ピクセル、24ビットrgb、jpeg（motion JPEG）、AVI形式、音声はモノ

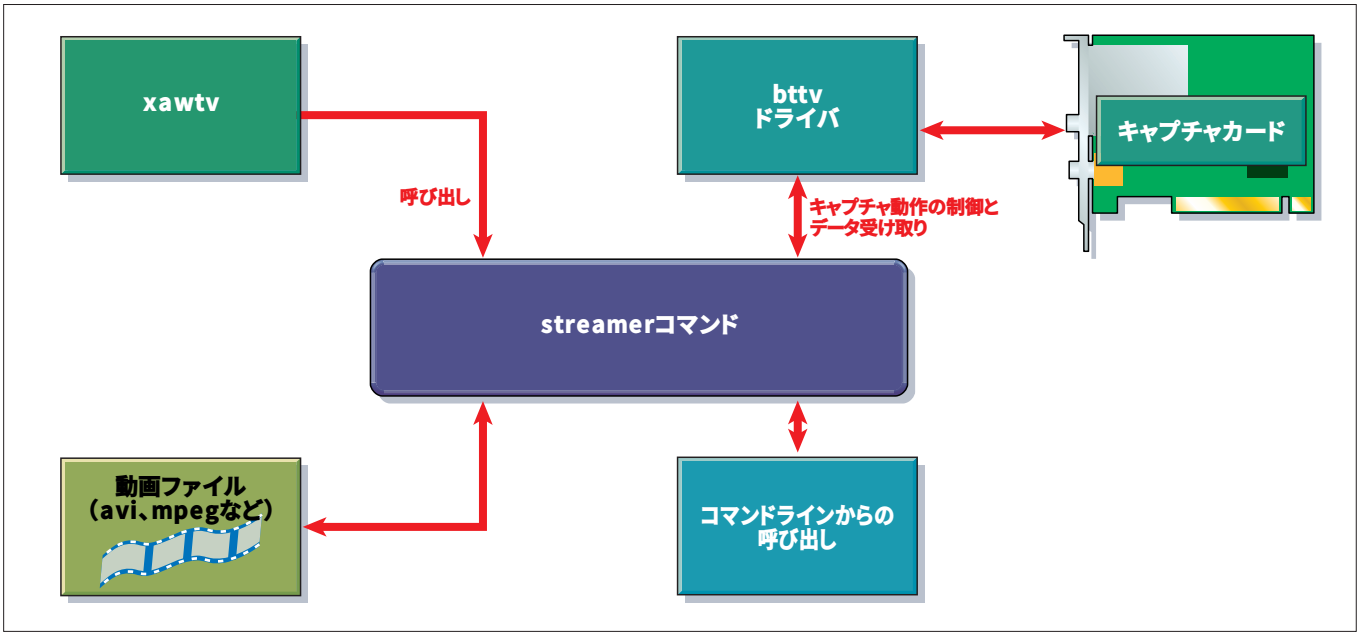
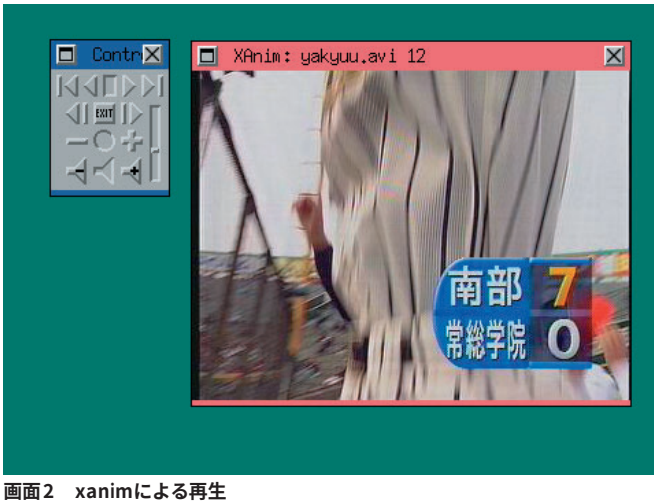


図1　streamerコマンドの役割
streamerコマンドはxawtvに付属するツールで、動画キャプチャのほとんどの処理を請け負っている。xawtvで動画の録画（recordボタン）を行う場合も、内部ではstreamerコマンドが呼び出されて、実際の処理を行っている。

```
kterm
[zen-t@zen06 zen-t]$ streamer -o yakyuu.avi -t 720 -f mjpeg -F mono8
avi / video: JPEG / audio: 8bit mono
v4l: bttv version 0.5.35
v4l: prehistoric bttv version found, please try to
     upgrade the driver before mailing bug reports
real: 8.851s   audio: 8.823s   video: 8.900s
```

画面1　streamerコマンド実行中の様子
AVI、motion jpeg、8ビットモノラル音声の形式で、streamerコマンドを使ってキャプチャしているようす。bttvドライバが古いという警告メッセージが表示されるが、とりあえず気にする必要はない。一番下の行に表示される途中経過の数値で、音声（audio）の数値と動画（video）の数値が著しくずれるようなら、キャプチャ処理が追いついていない可能性がある。この画面でも2つの数値がずれているように見えるが、これはキャプチャの頻度が1/10秒単位なので、丸め誤差が含まれているせいで、問題はない。



画面2　xanimによる再生
「xanim ファイル名」でxanimを起動すると、このような画面が表示される。左側に表示されているのがコントロールパネルで、コマ送り、逆再生、一時停止、音量調整、明るさ調整などができる。「q」キーを押せば、xanimを終了させることができる。

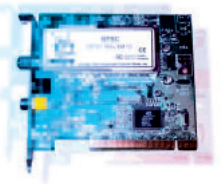
ラル8ビット、44.1kHzサンプリングの設定で10分間（10分×60秒×12fps＝7200フレーム）キャプチャしてみよう。この設定でキャプチャを行うには、streamerを次のように起動する。

```
$ streamer -o capture.avi -f mjpeg -r 12 -s 320x240 -t 7200 -n ntsc -F mono8
```

すると、**画面1**のように、コンソール上に途中経過を表示しながらキャプチャが開始される。Video for Linux（v4l）

全般オプション		
-h	ヘルプの表示	
-q	メッセージを表示しない	
-w seconds	キャプチャ開始まで待機する時間（秒単位）	
動画関連オプション		
-o file	録画ファイルの名前	
-f format	動画形式の指定（以下の「動画オプション」を参照）	
-c device	使用するVideo for Linuxデバイスの指定[/dev/video]	
-r fps	フレームレート（フレーム／秒）[10]	
-s size	記録サイズ[320x240]	
-t times	記録するフレームの数[1]	
-b buffers	バッファの数[16]	
-j quality	mjpegやjpegの品質（圧縮の程度）[75]	
-n tvnorm	カラースキームの設定（ntsc/ecam/pal）	
-i input	入力の切り替え（チューナーかライン入力か）	
音声関連オプション		
-O file	出力する音声ファイルの名前（音声を別ファイルに保存する場合に指定）	
-F format	音声形式の設定（以下の「音声オプション」を参照）	
-C device	使用する音声キャプチャデバイス（DSP）の指定[/dev/dsp]	
-R rate	サンプリングレート[44100]	
動画ファイルの諸形式（上の-fオプションや-Fオプションに設定する値）		
1つの動画ファイルでなく複数の静止画ファイルとして動画を保存する場合	**動画形式（-fオプションの設定値）**	
	ppm	24ビットTrueColor
	pgm	8ビットStaticGray（モノクロ）
	jpeg	JPEG
	音声形式（-Fオプションの設定値）	
	mono8	8ビット モノラル
	mono16	16ビット モノラル
	stereo	16ビット ステレオ
raw形式（圧縮なし）のビデオファイルに保存する場合	**動画形式（-fオプションの設定値）**	
	rgb	24ビットTrueColor
	gray	8ビットStaticGray（モノクロ）
	422	16ビットYUV 4:2:2
	422p	16ビットYUV 4:2:2 (planar)
	音声形式	
	mono8	8ビットモノラル
	mono16	16ビットモノラル
	stereo	16ビットステレオ
Microsoft AVI (RIFF) 形式のビデオファイルに保存する場合	**動画形式（-fオプションの設定値）**	
	rgb15	15ビットTrueColor（圧縮なし）
	rgb24	24ビットTrueColor（圧縮なし）
	mjpeg	motion JPEG（圧縮あり）
	音声形式（-Fオプションの設定値）	
	mono8	8ビットモノラル
	mono16	16ビットモノラル
	stereo	16ビットステレオ

表1　streamerコマンドのオプション一覧（[ ]内はデフォルトの設定値）



から「bttvのドライバが古いのでアップグレードしろ」という意味の警告メッセージが表示されるかもしれないが、ここでは気にしなくてもよい。スピーカーからは、テレビ放送の音声が聞こえてくるが、映像は表示されない。指定された時間（ここでは10分）経過すると、コマンドは終了して、カレントディレクトリにcapture.aviというファイルが生成されているはずだ。これを、xanimで再生してみよう。

```
$ xanim capture.avi
```

うまく再生できれば、キャプチャは成功である（**画面2**）。フレームレートや画面サイズがCPUの性能に比べて大きすぎるような場合、画像が飛び飛びに表示されたり、音声と画像が少しずつずれてしまうことがある。このような場合は、フレームレートや画像サイズを小さくして、もう一度キャプチャを試してみるといいだろう。

●チューナーの制御はv4lctlコマンドで

コマンドラインからテレビ番組のキャプチャができるようになったが、このままではチャンネルの切り替えができないので、無人運転での番組録画はできない。Linuxでビデオカードのチャンネルを切り替えるには、これもxawtvに付属のユーティリティ「v4lctlコマンド」を使う（**表2**にv4lctlコマンドの主なオプションを掲載する）。v4lctlコマンドは、キャプチャカード上のさまざまな機能を設定できるコマンドだが、もっともよく使うのが、チューナーのチャンネル切り替え機能だろう。チャンネルの指定は、前回説明した~/.xawtvファイル（**リスト1**。掲載の全リストは本誌ホームページよりダウンロードされたい）に記述した「放送局名」で行う。たとえば、.xawtvファイルの中で、チューナーの3chに「NHK-EDU」という放送局名を設定してある場合、チューナーを3ch（NHK-EDU）に設定するには次のようなコマンドを実行する。

```
$ v4lctl setstation NHK-EDU
```

すると、何行かのメッセージが表示される以外にとくに変化は起こらないが、これでチューナーのチャンネルが変更され、次にstreamerを起動したときには、ここで設定したチャンネルの番組がキャプチャされることになる。

このように、streamerとv4lctlの2つのコマンドさえ使えれば、コマンドラインからのキャプチャが可能になる。これらのコマンドは、Xが起動していない状態でも使えるので、マシンにログインしておらず、Xも立ち上げられていない状態、すなわち完全な無人の状態でも、電源が入っていてLinuxが起動してさえいればTV録画ができるわけだ。

setstation [<チャンネル名> | <チャンネル番号> | next | prev | back]	TVチューナーのチャンネルを変更する。~/.xawtvで定義したチャンネル名か、チャンネル番号で設定できる。nextを指定すると、現在の設定の次のチャンネルに、prevまたはbackを指定すると、前のチャンネルに変更する
setchannel [<チャンネル名> | next | prev]	setstationとほぼ同じ働きをするが、チャンネル番号での指定や「back」の指定ができない
setinput [<入力> | next]	ビデオ入力を切り替える。<入力>に指定できるのは、Television、Composite1など。nextを指定すると、ビデオ入力を順次切り替えることができる
capture [on | off | overlay | grabdisplay]	キャプチャモードの切り替え
volume [<n> | mute]	音量を0～65535の絶対値、あるいは-32767～+0～+32767の相対値で設定する。muteを設定すると、音声を消すことができる
color <n>	色の濃さを0～65535の絶対値、あるいは-32767～+0～+32767の相対値で設定する
hue <n>	色合いを0～65535の絶対値、あるいは-32767～+0～+32767の相対値で設定する
bright <n>	明るさを0～65535の絶対値、あるいは-32767～+0～+32767の相対値で設定する
contrast <n>	コントラスト値を0～65535の絶対値、あるいは-32767～+0～+32767の相対値で設定する
snap [jpeg | ppm] [full | win | widthxheight] <ファイル名>	静止画像をキャプチャする
webcam <ファイル名>	静止画像をjpeg形式でキャプチャする。avi形式の録画の最中でもキャプチャ可能。キャプチャに失敗した場合でも、以前にキャプチャした画像ファイルが残されるので、Web上のライブカメラの画像を更新するような用途に利用できる
movie file <ファイル名> | start | stop	aviファイルの記録を開始、停止する
fullscreen	フルスクリーン表示モードに切り替える。環境によってはフルスクリーンモードは安定しないことがあるので要注意
msg <テキスト文字列>	TV画面上に文字列をスーパーインポーズ表示する
quit	xawtvを終了する

表2　v4lctlコマンドの主要なオプション一覧
v4lctlコマンドは、コマンドラインオプションとして以下のような「コマンド」を与えることで、チューナーなどの設定を変更できる。「コマンド」の前にハイフンなどをつける必要はない。たとえば、チャンネルをsetstationコマンドで変える場合には「v4lctl setstation NHK」のようにすればよい。

streamerをもう少し使いやすくする

streamerコマンドを起動するのに長いオプションを入力したり、チャンネルを設定するためにv4lctlコマンドを起動するのはちょっと面倒である。また、録画ファイルの名前をいちいち考えるのも面倒だろう。そこで、これらのコマンドをあらかじめ設定したオプションをつけて起動する、簡単なスクリプトを作成してみよう。

この記事ではスクリプトを記述する言語としてPerlを使っている。実際にはどんなスクリプト言語を使ってもよいので、本記事を参考にして自分でスクリプトを記述する場合には、自分の好きなスクリプト言語を使ってもらいたい。

●スクリプトの仕様

スクリプトの名前は「tvcapture」にする。次のように、-pオプションで番組名、-lオプションで番組の長さ（分単位）、-cオプションでチャンネル名をオプションに指定すると、指定されたチャンネルのキャプチャを開始し、番組名と録画開始日時、チャンネル名をファイル名に含むaviファイルがカレントディレクトリ作成されるようにする。また、無人運転をする場合に備えて、録画ファイルを作成するディレクトリを-dオプションで指定できるようにしておく。

```
$ tvcapture -p ENGLISH -l 30 -c NHK-EDU -d /tmp
```

キャプチャ時のオプション設定などは、スクリプトの先頭部分でまとめて設定するようにして、独立した設定ファイルなどは使わないことにする。個人的な利用ならこれで十分だろう。

作成したスクリプトを**リスト2**に掲載する。このスクリプトを/usr/local/binディレクトリなどにコピーして、実行可能フラグを設定しておけば、コマンドラインから録画を行えるようになる。

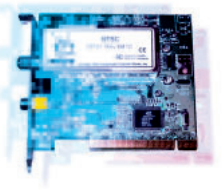
```
# cp tvcapture /usr/local/bin
# chmod +x /usr/local/bin/tvcapture
```

作成される録画ファイルのファイル名は、**図2**のような形式になる。録画日時をファイル名に含めているのは、あとで録画済みファイルを整理するときに便利だからだ。

▼リスト1　~/.xawtvファイルの設定例（再掲）

```
[global]
freqtab = japan-bcast
pixsize = 128 x 96
pixcols = 1
jpeg-quality = 75
mjpeg-quality = 75
keypad-ntsc = no
osd = yes

# [Station name]
# capture = overlay | grabdisplay | on | off
# input = Television | Composite1 | S-Video | ...
# norm = PAL | NTSC | SECAM | ...
# channel = #
# fine = # (-128..+127)
# key = keysym | modifier+keysym
# color = #
# bright = #
# hue = #
# contrast = #

[defaults]
norm = NTSC
capture = over
input = Television
color = 35%
contrast = 35%


[NHK]
channel = 1
norm = NTSC

[SHOP-CH]
channel = 2
norm = NTSC

[NHK-EDU]
channel = 3
norm = NTSC

[NTV]
channel = 4
norm = NTSC

[TV-SAITAMA]
channel = 5
norm = NTSC

[TBS]
channel = 6
norm = NTSC

[FUJI-TV]
channel = 8
norm = NTSC

[TV-ASAHI]
channel = 10
norm = NTSC

[U-AIR]
channel = 11
norm = NTSC

[TV-TOKYO]
channel = 12
norm = NTSC
```

各ユーザーのホームディレクトリに作成する.xawtvファイルの例。後述する「tvmaster」ユーザーのアカウントを作成したら、同じファイルを/home/tvmasterディレクトリにもコピーしておくこと。さもないと、自動録画でチャンネル切り替えができなくなる。



▼リスト2　/usr/local/bin/tvcaptureスクリプト（一部）

```
#!/usr/bin/perl

# tvcapture -- capture TV program using streamer
#
# usage : tvcapture -p <program_name> -l
<length_in_min> -c <Channel_Name> -d <directory>
#
# Copyright (c) 2001 by Zentaro Takeda
# Personal use Only

require "getopt.pl";

$capture_program = "/usr/local/bin/streamer";
$v4lctl = "/usr/local/bin/v4lctl";
$bash = "/bin/bash";
$fps = 12; # frames per second
$streamer_opt = "-f mjpeg -r 12 -n ntsc -F mono8";
# options for streamer

# channel name must be same as defined in
~/.xawtv
%channel = ("NHK"=> 1,
            "NHK-EDU"=> 3,
            "NTV"=> 4,
            "TBS"=> 6,
            "FUJI-TV"=> 8,
            "TV-ASAHI"=> 10,
            "U-AIR"=> 11,
            "TV-TOKYO"=> 12);

&Getopt('plcd');

if ($opt_p =~ /[^A-Z]/ ) {
  $program_name = "PROGRAM";
} else {
  $program_name = $opt_p;
}

if ($opt_l <= 0) {
  die "ERROR: Please specify length of recording
in minutes.\n";
} else {
  $minutes = $opt_l;
}

$flames = $minutes * 60 * $fps;

if ( $channel{$opt_c} > 0) {
  $channel_name = $opt_c;
} else {
  die "ERROR: unknown channel name $opt_c . \n";
}

if (length($opt_d)==0) {
  $dirname = ".";
} elsif ( opendir(FILEDIR, $opt_d) ) {
  $dirname = $opt_d;
} else {
  die "ERROR: cannot access directory $opt_d .\n";
}


# get current date and time
```

簡単な予約録画の方法

コマンド一発で録画ができるスクリプトを作成してしまえば、cron機能を使って簡単な予約録画を行うことができる。cron機能についての詳細はここでは説明しないが、crontab -eコマンドを使って、どのような時間にどのようなコマンドを起動するかを設定してやればよい。たとえば、毎週金曜日の午前6時40分から、/tmpディレクトリ内に「CHINESE」という番組名で、「NHK-EDU」チャンネルを30分間録画する場合には、次のような行をcrontabに追加する。

```
40 6 * * 5 /usr/local/bin/tvcapture -p CHINESE -l 30 -
c NHK-EDU -d /tmp
```

crontabでは、コマンドの起動時刻を、分、時、日、月、曜日の順に指定する。ここでは、左から「40分、6時、（毎日）、（毎月）、金曜日」というような意味になる。つまり、毎週金曜日の午前6時40分ということになるわけだ。

もう少し高度な予約録画

この連載は、最終的にはメールを利用した予約録画もできるようにすることを目的にしている。電子メールによる予約の仕組みについては、いずれ詳しく説明するとして、ここではそれを実現するための下準備をすることにしよう。

メールを使った予約では、メールの本文に含まれる予約情報を読み取って、その内容を元に録画を行うわけだが、上で説明したcronによる予約方法では、このような機能を実現するのはなかなか難しい。crontabの内容を、バッチ処理で書き換えるのは、かなり高度なテクニックがいるからだ。

そこで、少し発想を変えて、**図3**のような仕組みを考えてみた。録画を予約するときには、直接crontabに書き込むのではなく、予約内容を記したファイルを適当なディレクトリに作成するのだ。いわゆる「スプーリング」という手法である。

このディレクトリの内容を一定時間ごと（たとえば5分ご

図2　録画ファイルのファイル名形式
tvcaptureスクリプトで録画した録画ファイルの名前には、-pオプションで指定した番組名、録画開始の日時、-cオプションで指定した放送局名などが含まれる。

と）に検査して、5分以内に録画を開始すべき予約が存在していたら、その録画を実行するスクリプトをバックグラウンドで実行する。たとえば、2分（120秒）後に開始する予約の場合は、sleepコマンドを併用して、バックグラウンドで次のようなジョブを起動すれば、きっちり2分後に録画が開始されることになる。

```
sleep 120; tvrecord -p SPANISH -l 30 -c NHK-EDU -d
/tmp
```

予約ファイルの検査とジョブの起動を行うプログラム（スクリプト）を「予約エンジン」と名づけよう。この予約エンジンを5分おきにcronで起動すれば、定期的に予約状況を検査して、自動録画を行ってくれるようになるはずだ。

ところで、このままでは5分以内にすぐに開始させたい予約を行うことはできない。しかし、予約ファイルを作成した直後に手動で予約エンジンを起動できるようにしておけば、すぐに録画ジョブが開始されるのでこの問題も解決できるだろう。

●予約ファイルの形式

予約ファイルは、ファイル名そのものに録画開始時刻やチャンネルなどの情報を持たせるようにして、ファイルの内容は空のままにすることにした。そのほうがスクリプトが簡単にできるからだ。

これは余談だが、Linuxのファイルシステムは、それ自体を一種のデータベースシステムとして利用できる。ファイル名にさまざまな情報を持たせておけば、Linuxのシステムが備えているいろいろな関数を使って、その情報の検索が簡単にできるのである。もちろん、このような使い方に異論を唱える人は多いかと思うが、個人的に扱う範囲の情報を整理するのなら、十分に実用的な使い方だと考えている。ファイルの内容を読み出して処理したり、データベースシステムに関するプログラミングを行うのより、ずっと簡単にできる。

さて、予約ファイルの形式は、**図4**のように決めた。先頭から、録画番組名、チャンネル名、開始日時、録画時間（分単位）、繰り返しの間隔（日単位、繰り返さない場合は0）の順に、それぞれの項目の間は「_」で区切っている。拡張子は「.tvr」としておく。

たとえば、「ITALIAN」という番組名、「NHK-EDU」チャンネル、2001年3月19日、午前6時40分から30分間の番組を、毎週同じ日に録画する場合には、次のような予約ファイルを作成するわけだ。

```
ITALIAN_NHK-EDU_200103190645_30_7.tvr
```

●録画エンジン実行用のアカウントを作る

予約録画処理を無人で実行し、電子メールなどで予約でき

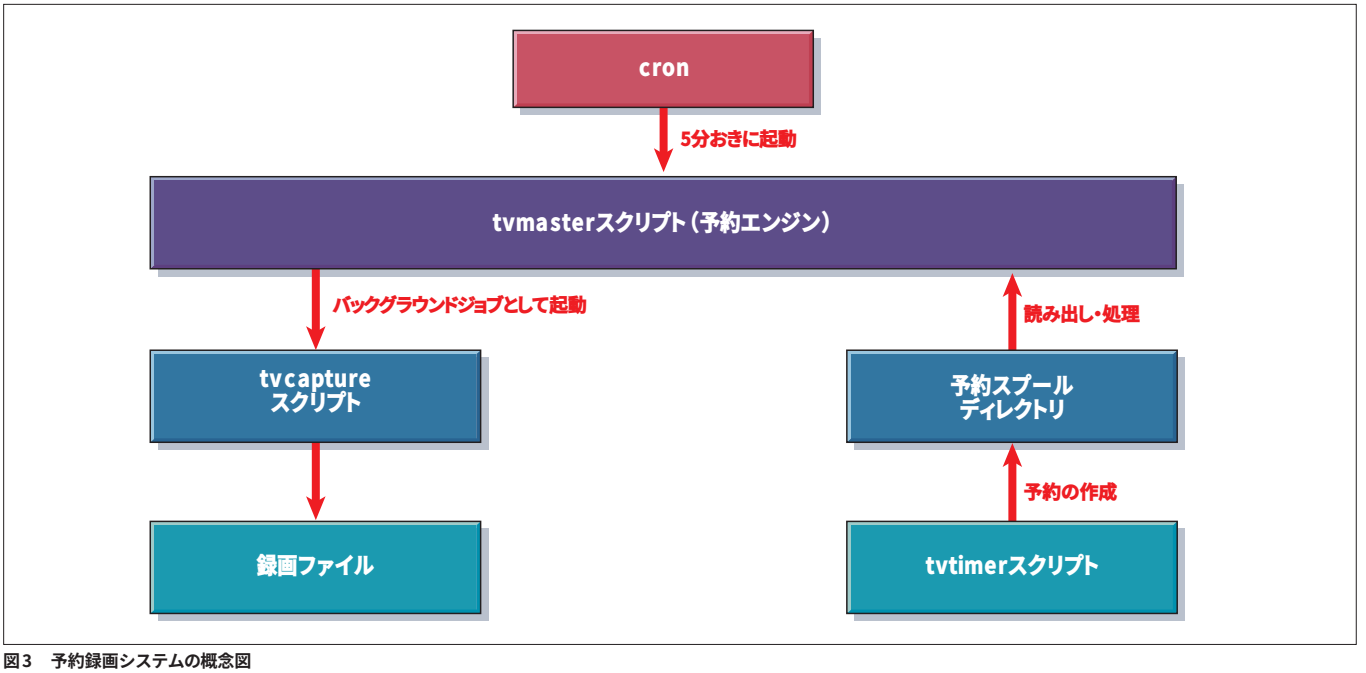


図3　予約録画システムの概念図
crontabの内容をバッチ処理で自動的に書き換えるのは難しいので、予約の実行は、tvmasterという名前のスクリプトを仲介して行うことにした。tvmasterスクリプトはcronによって定期的に起動され、一定時間以内に実行する必要のある予約がないかどうかを調べ、必要があれば録画ジョブ（tvcaptureスクリプト）をバックグラウンドで起動する。

るようにするために、専用のユーザーアカウントを作成した。閉じられた環境で個人的に運転するのであれば、ふだん使っているユーザーアカウントでも問題ないのだが、たとえば電子メールで予約を受け付けるような場合、それ専用のメールアカウントがあったほうが都合がよい。

ここでは、「tvmaster」という名前のユーザーアカウントを作成した。イタズラをされないように、パスワードも忘れずに設定しておこう。

```
$ su
# adduser tvmaster
# passwd tvmaster
```

Red Hat LinuxやTurbolinuxでは、adduserコマンドを使えば、ユーザーアカウントの作成やホームディレクトリの作成ができる。ほかのディストリビューションでも、同じコマンドが使えたり、同等の働きをするツールがあるはずだ。ユーザーアカウントの作成方法や、作成後の設定についてはここでは触れないが、作成したユーザーアカウントから/usr/local/bin/以下のコマンドを実行できるように、コマンドパスなどを設定しておくことを忘れないように。また、/devディレクトリ以下にある、サウンド関連のデバイスファイル（/dev/dsp?、/dev/mixer?など）のパーミッションが、tvmasterユーザーでも読み書きできる状態になっているかどうかも確認する。もし、そうなっていない場合は、suコマンドでrootユーザーになっている状態で、

```
# chmod 666 /dev/dsp?
# chmod 666 /dev/mixer?
```

などのように、これらのデバイスのアクセスができる状態に設定しておくこと。

●スプール用ディレクトリの作成

予約レコードファイルや録画済みファイルは、tvmasterのホームディレクトリ以下に置くことにした。これらのファイルを保存するために、次のようなディレクトリを作成した。

```
# mkdir /home/tvmaster/spool
# mkdir /home/tvmaster/spool/removed
# mkdir /home/tvmaster/recorded
```

いずれのディレクトリも、マシン上のすべてのユーザーからアクセスできるようにしておく。

```
# chmod 777 /home/tvmaster/spool
# chmod 777 /home/tvmaster/spool/removed
# chmod 777 /home/tvmaster/recorded
```

ここではセキュリティ上の問題については考慮していないが、もし気になるようなら、spoolディレクトリ以下だけは、tvmasterユーザー以外は読み書きできないように設定してもよい。ただし、このように設定すると、ローカルマシン上で予約を行う場合には、いちいちtvmasterユーザーとしてログインするか、suコマンドで一時的にtvmasterユーザーになる必要がある（次回以降に説明する電子メールによる予約では、ユーザーアカウントを気にする必要はない。tvmasterユーザー宛てにメールを送ればよい）。

●予約ファイル生成スクリプトの作成

予約ファイルをスプール上に生成するスクリプトは「tvtimer」という名前にした。tvtimerスクリプトの内容を**リスト3**に掲載する。そこそこに長いスクリプトになってしまったが、ほとんどがコマンドラインオプションと日付関連の処理で占められていて、主要な部分は最後の4行だけである。

tvtimerスクリプトでは、-p、-c、-d、-s、-l、-rの6つのオプションを指定できる。それぞれ、番組名の指定、チャンネルの指定、日付の指定、開始時刻の指定、録画の長さ（分単位）、繰り返し間隔（日単位）の指定を行う。このうち、番組名を省略すると「TMR」という名前になり、日付を省略すると当日を指定したことになる。また、開始時刻を省略すると、現在時刻が開始時刻となる。繰り返し間隔を省略すると、1回限りの録画予約を指定したことになる。チャンネル、録画の長さを省略することはできない。また、チャンネルや開始時刻などで入力ミスがあるとエラーメッセージが表示さ

図4　予約ファイルのファイル名形式
tvtimerコマンドを使うと、予約スプールディレクトリに、この図のような名前の空のファイルが作成される。

れる（**画面3**）。日付や時刻の指定は、コロンやスラッシュなどを一切含まない数値文字列で指定する。時刻は24時間制である。たとえば、2001年3月30日午後3時15分を指定するときは、

```
-d 20010330 -s 1515
```

というようなオプションをつけることになる。

日時の計算を一部簡略化しているので、過去の番組を予約しても、警告メッセージなどは表示されず、予約レコードが作成されてしまう。もちろん、過去の日時をチェックする処理は難しい処理ではないので、付け加えることは簡単にできるだろう。ちなみに、過去の日時の予約レコードは、後述するtvmasterが実行されると、自動的に削除されるようにしている。

なお、tvtimerコマンドは一般ユーザーが実行して予約ファイルを作成するので、作成される予約ファイルの「所有者」にはさまざまなユーザーのUIDが設定されることになる。このため、予約実行後の予約ファイルの処理をしやすくするため、作成した予約ファイルのアクセス権を、一意に読み書き可能に設定している。

▼リスト3　/usr/local/bin/tvtimerスクリプト（一部）

```
#!/usr/bin/perl

# tvtimer -- reserve a program for TV capture
#
# usage : tvtimer -d <start_date> -s <start_time>
-p <program_name> -l <length_in_min> -c
<Channel_Name> -r <repeat_duration>
#
# Copyright (c) 2001 by Zentaro Takeda
# Personal use Only

require "getopt.pl";

$spool_dir = "/home/tvmaster/spool";
$bash = "/bin/bash";

# channel name must be same as defined in
~/.xawtv
%channel = ("NHK"=> 1,
            "NHK-EDU"=> 3,
            "NTV"=> 4,
            "TBS"=> 6,
            "FUJI-TV"=> 8,
            "TV-ASAHI"=> 10,
            "U-AIR"=> 11,
            "TV-TOKYO"=> 12);

# get current date and time
($sec, $min, $hour, $mday, $mon, $year, $wday,
$yday, $isdst) = localtime();

$year = $year + 1900 ; # y2k correction
$mon = $mon + 1;

# command line options
&Getopt('dsplcr');

if (length($opt_d) == 0) {
  $start_day = substr("0000$year", -4) .
substr("00$mon", -2) . substr("00$mday", -2);
} elsif (length($opt_d) <= 4) {
  $start_day = substr("0000$year", -4) .
substr("00$opt_d", -4);
} else {
  $start_day = $opt_d;
}

if (length($opt_s) == 0) {
  $start_time = substr("00$hour", -2) .
substr("00$min", -2);
} else {
  if (length($opt_s)==4){
```

●予約エンジンの作成

「予約エンジン」は、予約スプール（/home/tvmaster/spool）上にある予約レコードファイルを一定期間ごとに調べて、直近に実行しなければならない予約レコードがあったら、それをバックグラウンドで開始する処理を行う。実行済みの予約レコード、あるいは録画開始時刻が過去の日時のレコードについては、繰り返しの指定がある場合は日付の部分だけを書き換えてファイル名を変更し、繰り返しの指定がない場合（繰り返し期間のフィールドがゼロの場合）は、/home/tvmaster/spool/removedディレクトリに移動する。

予約エンジンのスクリプト本体は「tvmaster」という名前にして、/usr/local/binディレクトリに置くことにした。tvmasterスクリプトの内容は**リスト4**のとおりである。

このスクリプトも、内容の大半は日時関係の処理になっているが、少し注意して見てもらいたい部分として、ディレクトリの排他ロック処理を行っている部分がある。37～38行目

画面3　tvtimer実行のようす
オプションに指定するチャンネル名を間違えたり、録画の長さを指定しなかったりすると、エラーメッセージが表示される。エラーがない場合は何も表示されない。予約が間違いなく行われているかどうかは、予約スプールディレクトリ（/home/tvmaster/spool）の内容を調べればわかる。

と96～99行目がそれである。

tvmasterスクリプトは、cron機能によって定期的に実行される以外に、5分以内に開始予定の予約を行ったときなどに、ユーザーが直接起動することも想定している。すると、同じマシン上で2つ以上のtvmasterスクリプトが起動される可能性が起こるわけだ。このような場合、もし、一方のスクリプトがスプールの内容を変更するような処理（予約レコードの削除や書き換え）を行うと、もう一方のスクリプトはその変更の内容を知ることができず、同じ録画処理ジョブが重複して実行されたり、削除されたはずのレコードを実行してしまったりする危険性がある（**図5**）。

これを避けるため、tvmasterスクリプトがスプールディレクトリの中身を処理している最中は、ほかのtvmasterスクリプトはその処理が終わるまで、スプールディレクトリを読み込む前の段階で待機するようにしている。このような排他処理は、マルチタスク処理のシステム上では、なかば常識的に行われているものだ。

ところで、tvmasterスクリプト同士では排他処理を行っているのに、前述のtvtimerプログラムなどでは行っていないのは、一貫していないのでは、という疑問が起こるかと思うが、スプールの内容が追加される処理については、排他処理を行わなくても致命的な結果にはならないのだ。たとえば、tvmasterがスプールを処理している最中にレコードが追加されても、そのレコードは次回にtvmasterが実行されたときに

▼リスト4　/usr/local/bin/tvmasterスクリプト（一部）

```
 1: #!/usr/bin/perl
 2:
 3: # tvmaster -- check timer spool and execute
capture command
 4: #
 5: # usage : tvmaster
 6: #         (there is no command line option)
 7: #
 8: # Copyright (c) 2001 by Zentaro Takeda
 9: # Personal use Only
10:
11: $spool_dir = "/home/tvmaster/spool";
12: $record_dir = "/home/tvmaster/recorded";
13: $bash = "/bin/bash";
14: $timer_command = "/usr/local/bin/tvcapture";
15:
16: use Time::Local;


    （中略）


37: # lock spool
38: open(SPOOLLOCK, ">/tmp/tvmaster.lock") or die
"ERROR: unable to open lock file.\n";


    （中略）


91: #    print $commandline;
92:   }
93:   }
94: }
95:
96: #end critical section
97: flock (SPOOLLOCK, 8); #unlock
98: close (SPOOLLOCK);
99: unlink("/tmp/tvmaster.lock");
```

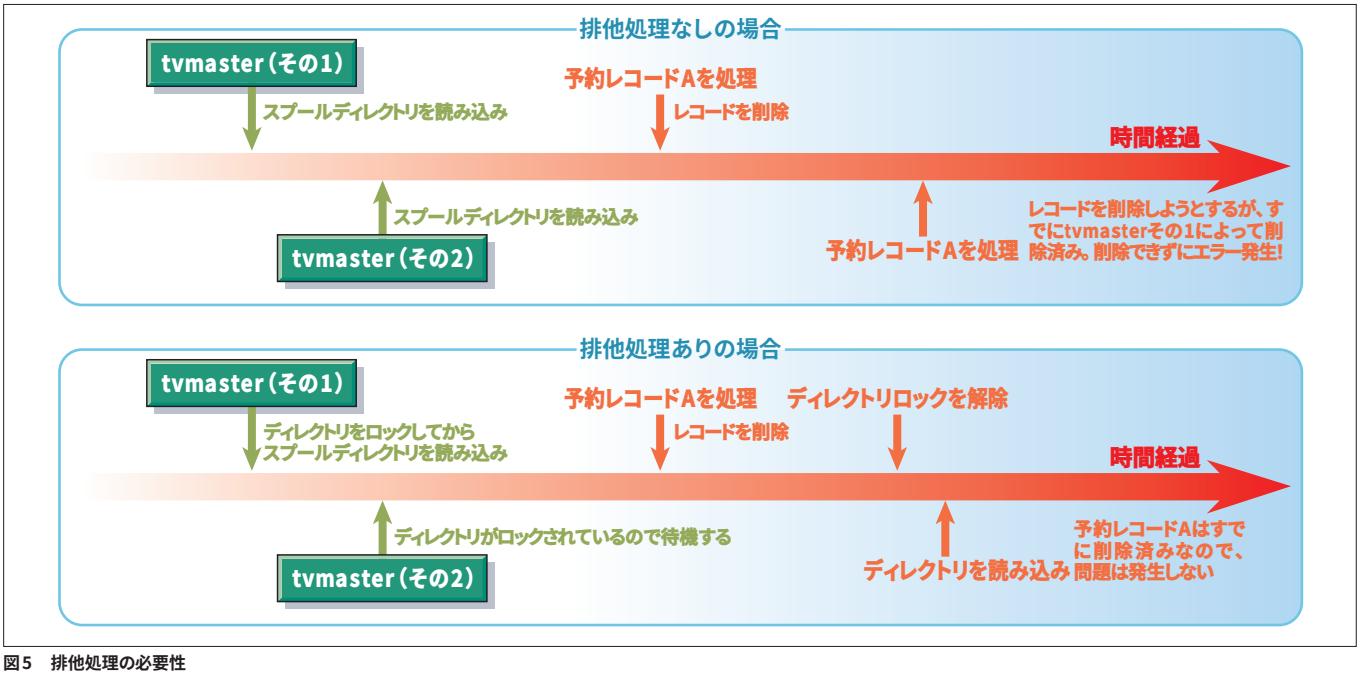


図5　排他処理の必要性

tvmasterスクリプトが2つ以上同時に起動されると、それぞれのスクリプトが同一の予約レコードを処理しようとして、問題が起こる可能性がある。これを避けるため、スプールディレクトリの内容を、複数のtvmasterスクリプトが同時に操作することのないように、「ロック」をかける。

問題なく処理される。tvmasterは5分間隔で実行され、10分以内に開始する録画ジョブを実行するようになっているので、問題は起こらないはずだ（**図6**）。

●予約エンジンの設定

tvmasterスクリプトを作成したら、これを/usr/local/binディレクトリにコピーし、実行属性を設定する。

```
$ su
# cp ./tvmaster /usr/local/bin
# chmod 555 /usr/local/bin/tvmaster
```

試しに、いろいろな内容の予約レコードをtvtimerスクリプトを実行して作り、tvmasterスクリプトを実行してみて、想定どおりの動作をするかチェックしてみよう。

問題なく動作するようなら、tvmasterスクリプトが定期的に実行されるように設定する。ここでは、5分おきに実行するように設定する。tvmasterユーザーとしてログインしている状態で、crontab -eコマンドを実行して、実行スケジュールを**リスト5**のように設定する。

```
# su tvmaster
$ crontab -e
```

リスト5の設定では、毎時1分から5分おきに起動するように設定している。なぜ「0分、5分、10分、……」というようにきりのよい時刻にしなかったかといえば、TV番組の録画は0分、15分、30分といったように、きりのよい時刻に設定されることが多いので、tvmasterスクリプトが実行されるタイミングと実際の予約録画が開始される時刻を、できるだけずらしておこうと考えたからだ。いまのままのスクリプトでも、録画処理とtvmasterの動作が同時に発生しても問題が起こる可能性は少ないが、トラブルの種はできるだけ減らしておいたほうがよい。

Webページから予約できるようにしてみる

tvtimerによる予約操作は、コマンドラインで開始時刻やチャンネル、録画時間などを指定するだけでよいので、慣れてしまうと、普通のビデオデッキのリモコンよりも使い勝手が良い。予約を間違えても、スプールファイルにある予約をrmするだけで、簡単に取り消しができる。繰り返し録画の設定は、現在はごく簡単な指定（録画する日付の間隔）だけだが、tvmasterスクリプトやtvtimerスクリプトに手を加えれば、たとえば平日のみ録画するように指定したり、「毎月最終金曜日」といった指定もできるようになるだろう。

録画予約は指定のスプールディレクトリにファイルを作成するだけの簡単なものなので、tvtimer以外のプログラムから予約を行うことも可能だ。次回以降に、電子メールによる予約を行えるようにするが、今回のおまけとして、Webページから予約する簡単なインターフェイスを作成してみよう。

Webページから録画済ファイルを再生したり、予約の取り消しなどをできるようにする方法については、改めて解説することにしたい。また、今回はWebサーバ（Apache）の設定方法については触れないので、これ以降は、すでにApacheが利用可能な状態になっていて、かつ各ユーザーのホームディレクトリにあるpublic_htmlディレクトリ以下の内容が、「http://ホスト名/~ユーザー名/」というURLで参照可能になっていることを前提とする。また、/home/httpd/cgi-bin以下のディレクトリに置かれたCGIスクリプトが実行可能な状態になっている必要もある。

●予約用ページのHTML

まずは、**リスト6**のようなHTMLファイル（timer.html）

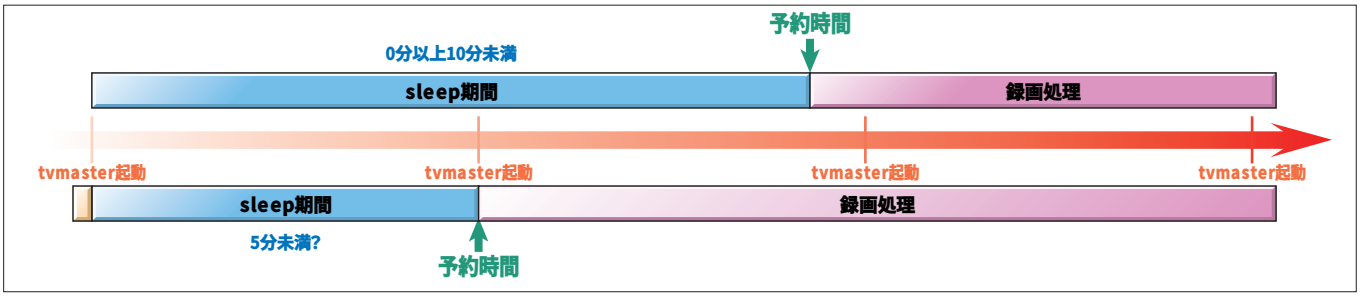
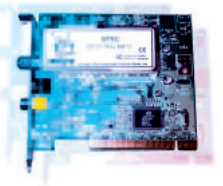


図6　5分間隔で予約を実行する様子
cronによってtvmasterが起動されるタイミングは、正確に5分間隔になるという保証はない。もし、正規のタイミングより早くtvtimerが起動されてしまった場合、実行開始まで5分以上あると判断されてしまって、予約が「漏れる」可能性がある（下）。10分未満の設定で起動するようにしておけば（上）、たとえ1回目のtvtimerで「漏れ」てしまっても、次にtvtimerが起動したときに確実に予約が実行される。



▼リスト5　crontab -eの設定例

```
1-56/5 * * * * /usr/local/bin/tvmaster
```

▼リスト6　/home/tvmaster/public_html/timer.html（一部）

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
  <head>
    <title>TVWEBMASTER</title>
  </head>

  <body>
    <h1>TV録画予約</h1>
<hl>
<FORM action="http://zen06/cgi-bin/webtvtimer"
method="GET">
<P>
<LABEL>番組名(半角大文字アルファベット):<INPUT
type="text" name="Program"></LABEL>
</P>
<p>
チャンネル
<select name="Channel">
<option>NHK</option>
<option>NHK-EDU</option>
<option>NTV</option>
<option>TBS</option>
<option>FUJI-TV</option>
<option>TV-ASAHI</option>
<option>U-AIR</option>
<option>TV-TOKYO</option>
</select>
```

▼リスト7　/home/httpd/cgi-bin/webtvtimer

```
#!/usr/bin/perl

@rarg=split(/\&/, $ENV{'QUERY_STRING'});

foreach $parg (@rarg) {
  @argexp=split(/=/,$parg);
  $garg{$argexp[0]}=$argexp[1];
}

print "Content-type: text/html\n\n";
print "<HTML>";
print "<BODY>";

$timeroption = " -p ".$garg{"Program"}. " -c
".$garg{"Channel"}." -d ".
  $garg{"Year"}.$garg{"Month"}.$garg{"Day"}.
  " -s ".$garg{"Hour"}.$garg{"Minute"}.
  " -l ".$garg{"Length"}.
  " -r ".$garg{"Repeat"};

print "次のコマンドを実行しました。ご確認ください。
<br>\n\n";
print "tvtimer ";print
$timeroption;print"<br><hr>\n\n";

# footer
print "This page is provided for personal use
only.\n";

#closing

print "</BODY>";
print "</HTML>\n";
system "/usr/local/bin/tvtimer ".$timeroption;
```

を作成して、/home/tvmaster/public_htmlディレクトリに配置する。このHTMLファイルには、予約時刻やチャンネルの入力を行うためのリストボックスなどが記述されている（**画面4**）。どのような内容なのかは、リストと画面を見比べれば見れば大体わかるだろう。このHTMLファイルは、「フォーム」の初歩的な例としても見てほしい。

設定された内容は、「予約」ボタンをクリックすることで、Webサーバに送信される。Webサーバは予約の内容を受け取って、それをCGIスクリプトに渡す。

Webサーバから受け取った予約の情報を処理するCGIスクリプトが、**リスト7**の「webtvtimer」スクリプトだ。このスクリプトは、Webサーバから環境変数経由で受け取った情報を使って、tvtimerスクリプトを呼び出している。

webtvtimerスクリプトを作成したら、

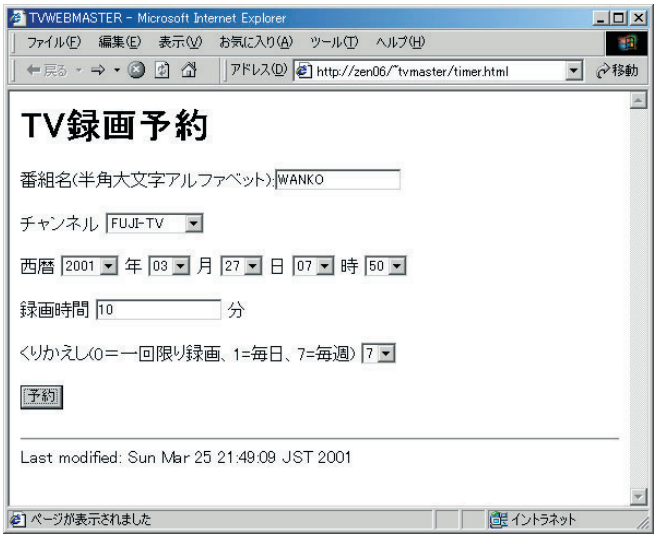
```
# chmod +x /home/httpd/cgi-bin/webtvtimer
```

で、実行可能に設定しておくこと。

予約を行った結果は、spoolディレクトリの内容を見ればわかる。また、「予約」ボタンを押した結果、どのようなコマンドが実行されたかをWebページ上に表示するようにしてみた。

この予約用ページは、きわめて簡単なもので、予約済みの情報などを見ることはできないが、とりあえず、今回作成したtvmaster、tvtimerなどのスクリプトが、それなりに汎用性が高いことがわかっていただけただろうか。

次回は、録画するデータをMPEG-1形式に圧縮することで、より長時間の録画を行える方法を紹介したいと思う。



画面4　録画予約画面

連載特集

申し込みから独自ドメイン運用まで徹底解説！

フレッツ・ISDNで常時接続インターネットサーバ!

独自サーバで自分だけのインターネットサイトを構築しよう

5回に渡って解説してきた、フレッツ・ISDNで常時接続サーバを構築するというこの連載特集も、いよいよ今回が最終回だ。最終回を迎えた今回は、独自ドメインによるWebサーバの構築と、LANの構築、そして最後にファイアウォールとセキュリティについて解説することにしよう。

この特集は今回で終了だが、先月号で特集した入門・Linux LANの設定を参照して、LAN内のサービスを強化していくのもいいだろう。ここまでサーバを構築できたのであれば、あとは自分なりのカスタマイズも可能だ。自分なりのサイトを構築していってほしい。

ACT. 5

ISDN フレッツ 常時接続

Webサーバの構築

今回までWebサーバの構築に関してはほとんど触れていなかった。しかし、独自ドメインサーバを構築したら、やっぱりWebページも公開したくなるだろう。

Linuxでは、Apacheを使ってWebサーバを構築するのが一般的だ。ApacheはLinuxを始め、多くのUNIX上とWindowsで稼働するフリーのWebサーバで、Apache Software Foundationが開発を行っている。事実上、フリーのWebサーバのデファクトスタンダードだといえる。

さて、早速Apacheを使ってWebサーバを構築しよう、といいたいところだが、実際のところ、インストールさえしてしまえば、ほとんどデフォルトのままの設定で問題なく稼働することができる。もちろん、Apacheにはアクセス制限など、豊富な機能が用意されているが、通常、Webページを公開するだけであれば設定ファイルを変更する必要はない。

まずは、Apacheがインストールされているかどうかを確認しよう。

```
# rpm -q apache
apache-1.3.12-25
```

もし、インストールされていないのであれば、rpmコマンドでインストールしよう。

次に、Apacheが起動しているかどうかを確認する。

```
$ /etc/init.d/httpd status
```

または、

```
$ /etc/rc.d/init.d/httpd status
```

Apacheが起動していれば、

```
httpd (pid ……) is running...
```

と表示されるはずだ。起動していないのであれば、ntsysvコマンド（Turbolinuxでは、turboserviceコマンド）で、Apacheがシステム起動時に有効になるように設定しておこう。

なお、Apacheが起動している状態でWebブラウザを起動し、「http://＜サーバ名＞/」を表示させると、Apacheのテストページが表示されるはずだ（**画面1**）。Apacheのテストページが表示されれば、Apacheは正常に起動している。

公開するコンテンツについて

Webサーバの準備ができたら、HTMLファイルを作成する。ここではHTMLファイルの作成方法については詳しく触れないが、HTML出力機能を持ったワープロソフトや、HTMLエディタなどで作成するといいだろう。編集部のフレッツ・ISDNサーバでは、**画面2**のようなWebページを用意した。

ただし、フレッツ・ISDNでは画像ファイルなど、大きなファイルを公開する場合は注意が必要だ。たくさんの画像をサーバから転送しようとすると、通信帯域が64kbpsしかないISDNでは、表示が極端に遅くなってしまうからだ。

しかしこれは、画像をたくさん貼り付けたWebページが公開できないとい

画面1　Apacheのテストページ

画面2　フレッツ・ISDNサーバ用に用意したトップページ

うことではない。よく、ISDNでは帯域が細いため、Webページを公開するのは難しいといわれるが、ちょっとしたテクニックを使えば転送時間を大幅に軽減できるのだ。

もし、画像をたくさん貼り付けたWebページを公開したい場合などは、別のプロバイダのサーバにあるレンタルスペースなどに画像を置いておき、HTMLファイルだけをフレッツ・ISDNのサーバ上で公開するようにすればいい（**図1**）。こうしておけば、Webページはあたかもフレッツ・ISDNサーバ上から転送されたかのように見えつつ、画像は専用線で接続されたレンタルスペースから転送されるため表示が高速になる。これで、フレッツ・ISDNの通信帯域のボトルネックを気にすることなく、画像ファイルを貼り付けることが可能だ。

実際、編集部のサーバでも、多くの画像は別のサーバ上の画像にリンクさせている。

コンテンツの用意

コンテンツが作成できたら、Webサーバのドキュメントルートにコピーする。ドキュメントルートとは、WebブラウザのURLに「http://＜サーバ名＞/」と入力したときにアクセスされるディレクトリだ。Webブラウザはドキュメントルートに保存された、「index.html」をトップページとして表示する。

ドキュメントルートがどのディレクトリに設定されているかはディストリビューションによって異なる。編集部のサーバにインストールしたRed Hat Linux 7Jでは、「/var/www/html」がドキュメントルートになっている。ほかのディストリビューションでは、「/home/httpd/html」や、「/usr/local/apache/htdocs」になっているかもしれない。ドキュメントルートに設定されるディレクトリは、/etc/httpd/conf/httpd.confファイル内の、「DocumentRoot」行を変更することで設定可能だ。通常はドキュメントルートディレクトリを変更する必要はないだろう。また変更する場合、インターネット上からアクセスされるディレクトリとなるので、/ディレクトリなど、セキュリティ上問題となるディレクトリを指定してはならない。

それではコンテンツをドキュメントルートにコピーしよう。ずいぶん簡単だが、以上でWebサーバの設定は完了だ。WebブラウザにURLを入力して、Webページが表示されるかどうかを確認してみよう。

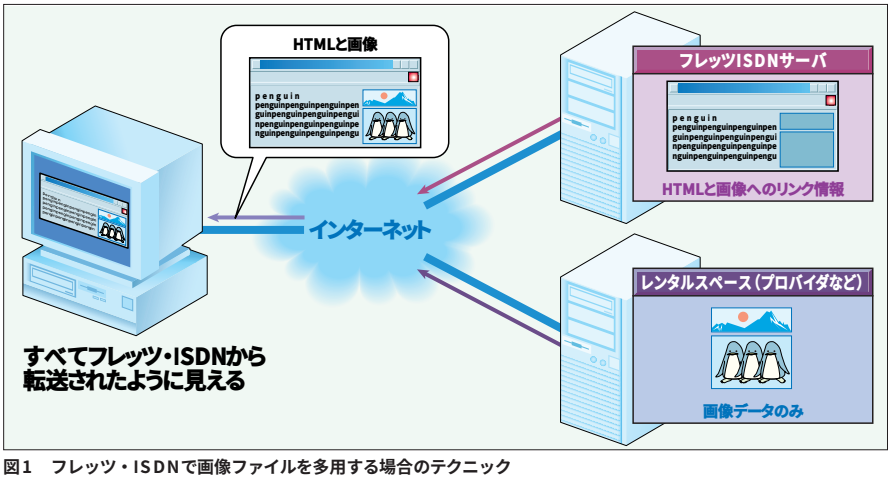


図1　フレッツ・ISDNで画像ファイルを多用する場合のテクニック

ユーザーディレクトリ

サーバのトップページは表示することができたが、各ユーザーのホームページはどのように公開したらいいだろうか。ユーザーのホームページを公開する方法は2種類ある。1つはドキュメントルート以下にディレクトリを作成し、そのディレクトリの所有者とグループパーミッションをそのユーザーに設定する。これでそのユーザーはこのディレクトリに自由にコンテンツを置くことができる。ユーザーのコンテンツにアクセスする場合は、「http://＜サーバ名＞/＜ディレクトリ名＞/」というURLになる。

もう1つは、ユーザーのホームディレクトリに「public_html」というディレクトリを作成し、このディレクトリにコンテンツを置く。この場合は「http://＜サーバ名＞/~＜ユーザー名＞/」というURLになる。

どちらの場合も、ディレクトリ属性の実行権限を全ユーザーに対して許可しておかなければならない。たとえば、

```
$ chmod 701 dirctory
```

と設定しておく必要がある。これは、Apacheがユーザーapacheとして実行されるため、全ユーザーに対して実行権限が必要となるためだ（ディストリビューションによってはユーザーnobody権限で実行される）。public_htmlに置いたHTMLを公開する場合は、ユーザーのホームディレクトリ自体にも全ユーザーに対して実行権限が必要となるので注意しよう。表示させようとした際に、「Forbidden」と表示される場合は、ホームディレクトリ自体の実行権限が設定されていない可能性がある。

LANの構築

ここまで、フレッツ・ISDNのサーバはインターネット側へのみ接続されていた。しかし実際に運用していくと、サーバ以外のマシンからもインターネットに接続したいだろう。そこでLANを構築し、LAN内のマシンからインターネットに接続できるように設定しよう。このように、インターネットとLANという2つのネットワークを相互に接続する場合は、インターネットとLANの間にルータを設置する必要がある。このルータ機能をフレッツ・ISDNサーバに持たせて、インターネットとLANのルーティングを行うようにしてみよう。

Linuxをルータとして機能させるためには、ネットワークカードが2枚必要となる。1枚はすでにダイヤルアップルータに接続されているインターネット向けのものだ。もう1枚はLAN側に接続するLAN向けのものだ。現状では1枚のネットワークカードしか装着されていないので、新しいネットワークカードを追加することにする。

編集部のフレッツ・ISDNサーバではプラネックスコミュニケーションズのFW-100TXを購入した。このネットワークカードはREALTEKのコントローラ（rtl8139）を搭載しており、Linux Readyとなっているものだ。

ネットワーク計画

さて、LANを構築するにあたり、まずネットワーク計画を立てよう。すでにインターネット側のネットワークは構築してあるので、LAN側のみの計画だけでOKだ。

まずはLAN側のIPアドレスを決定しよう。LANは直接インターネットに接続されるわけではなく、サーバを介してインターネットに接続される。このため、LAN側のマシンにはグローバルIPアドレスは必要ないので、プライベートIPアドレスを割り振るようにしよう。このほうがセキュリティ上も安全だし、IPアドレスの自由度が高い。

プライベートIPアドレスとは、グローバルIPアドレスとは独立した、LAN内で自由に割り振ることができるアドレス空間である。プライベートIPアドレスのアドレス空間は次の3つが用意されており、LANの規模に応じて選択すればいい。

```
10.0.0.0〜10.255.255.255
172.16.0.0〜172.31.255.255
192.168.0.0〜192.168.255.255
```

これらのアドレスはインターネット上に存在することが禁止されたアドレスである。

編集部のサーバではそれほど大きいLANを構築する必要もないので、192.168.0.0/24というプライベートIPアドレスの空間を使用することにした。

マシンの各IPアドレスはサーバのLAN側のIPアドレスを192.168.0.1に割り当て、LAN内の各マシンは192.168.0.10から順に割り当てることにする。

プライベートIPアドレスの割り振りをまとめると**図2**のようになる。

■ハードウェアの設定

それでは新しいネットワークカードを追加しよう。

サーバをshutdownして電源を切り、新しいネットワークカードをマシンに装着しよう。FW-100TXはLinuxに対応したネットワークカードなので、装着してLinuxを起動すればPlug and Play機能であるkudzuによって自動的に認識される。Linuxを起動すると、kudzuが新しいハードウェアを認識す

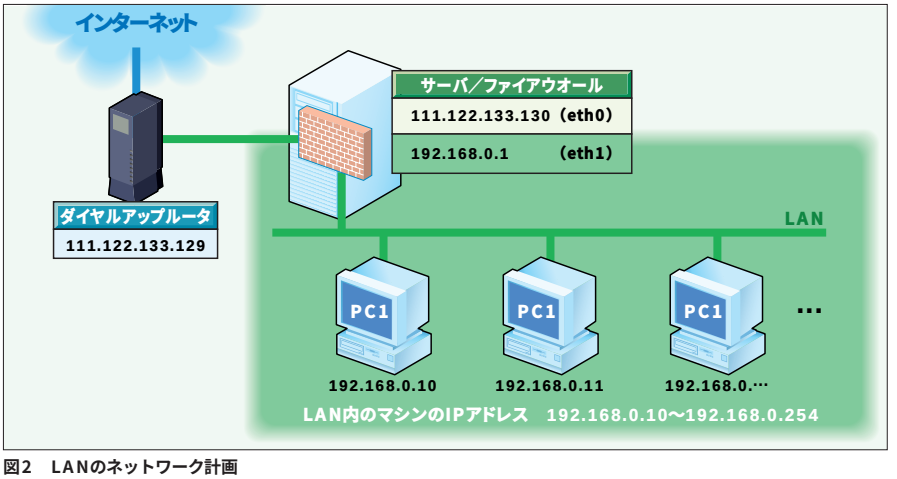


図2　LANのネットワーク計画

るので、「Hardware Added」と表示されたら、「Configure」を選択し、「Yes」を選択する。以上でハードウェアの設定は完了だ。

もし、kudzuによって認識されない場合は、/etc/modules.confにネットワークカードのドライバとネットワークインターフェイスを記述する。次の例は今回使用したFW-100TXのデバイスドライバであるrtl8139に、eth1というネットワークインターフェイスを登録した場合の例だ。

```
alias eth1 rtl8139
```

■ネットワークカードの設定

Linuxが起動したら、ネットワークカードの設定を行う。ネットワークカードの設定は、netconfコマンドが簡単だ。rootユーザー権限でnetconfコマンドを実行する。次に、「Basic host information」を選択してEnterキーを押す。ここで表示されたウィンドウをスクロールし、「Adapter 2」を表示させる。続いて、「Enabled」をチェックしてネットワークカードを有効にし、「Primary name + domain」にサーバの名前を入力する。次に、「IP address」にLAN側のIPアドレス、「Netmask」にネットマスクを入力し、「Net device」に「eth1」と入力し、「Accept」を押す（**画面3**）。設定が完了したら、「Quit」を押して、「Activate the changes」を選択して変更を有効にする。以上でネットワークカードの設定は完了だ。

次に、ネットワークインターフェイスeth1をifconfigコマンドを使って有効にしよう。

```
# ifconfig eth1 up
```

続いてパケット転送を行うために、/etc/sysconfig/networkファイルと/etc/sysctrl.confファイルにリスト1、リスト2のように変更を加える。

すべての設定が完了したらネットワークを再起動する。

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

または、

```
# /etc/rc.d/init.d/network restart
```

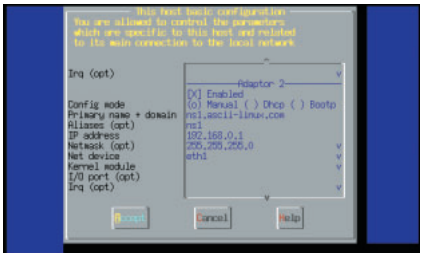
以上で2枚目のネットワークの設定作業は終了だ。ifconfigコマンドを実行して、ネットワークが正常に起動できているか確認しよう（**画面4**）。eth1に正しくIPアドレスが設定されていればOKだ。

▼リスト1　/etc/sysconfig/networkに追加する項目

```
FORWARD_IPV4="yes"
```

▼リスト2　/etc/sysctl.confに追加する項目

```
net.ipv4.ip_forward = 1
```



画面3　2枚目のネットワークカード（Adapter 2）の設定

```
# ifconfig
eth0  Link encap:Ethernet  HWaddr 00:11:22:33:44:55
      inet addr:123.123.123.123 Bcast:123.123.123.255 Mask:123.123.123.248
                                :

eth1  Link encap:Ethernet  HWaddr 22:33:44:55:66:77
      inet addr:192.168.0.1 Bcast:192.168.0.255 Mask:255.255.255.0
                                :

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
                                :
```

画面4　ifconfigの出力

LAN上のマシンの設定

LAN上のマシンの設定については簡単にだけ触れておく。今回のフレッツ・ISDNではDHCPサーバを構築していないので、LAN上の各マシンのIPアドレスは手動で設定することにしよう。今回のネットワーク計画では、192.168.0.10から順に割り当てていけばいいだろう。

デフォルトゲートウェイとDNSサーバはサーバのLAN側のIPアドレスとなる。設定すべき項目をまとめると次のようになる。

- **IPアドレス**
  `192.168.0.10〜192.168.0.254`
- **ネットマスク（サブネットマスク）**
  `255.255.255.0`
- **DNSサーバ**
  `192.168.0.1`
- **デフォルトゲートウェイ**
  `192.168.0.1`

なお、LAN側からインターネットに接続するには、次で解説するIPマスカレードの設定が必要となる。

ファイアウォールの設定

LANが構築できたところで、サーバのファイアウォールの設定と、LAN側のマシンからインターネットに接続できるように設定しよう。ファイアウォールについては、今までダイヤルアップルータのパケットフィルタリングを利用していたが、今回はサーバ側で設定するようにする。

LAN側のマシンからインターネットに接続するためには、IPマスカレードを利用する。IPマスカレードとは、簡単にいうとプライベートIPアドレスを持ったLAN側のマシンをインターネットに接続させるためのもので、1つのグローバルIPアドレスを複数のプライベートIPアドレスで共有できる便利な機能だ。IPマスカレードを使えば、LAN側のマシンを安全にインターネットに接続することが可能だ。

Linuxでのファイアウォールと、IPマスカレードはそれぞれ別の機能だが、ともにipchainsを使って設定することができる。ipchainsを使ったファイアウォールでは、ポートフィルタリングのほか、パケットフィルタリングが可能だ。

ipchainsによるフィルタの設定

ipchainsは次のような条件でフィルタリングすることができる。

- **送信元IPアドレス**
- **送信先IPアドレス**
- **送信元ポート番号**
- **送信先ポート番号**
- **ネットワークインターフェイス**

これらの条件を複数組み合わせることも可能だ。さらに、これらの条件が適用されるタイミングとして、

- **パケットを受信したとき（input）**
- **パケットがルーティングされたとき（forward）**
- **パケットを送信するとき（output）**

の3つが指定できる。それぞれの条件は、inputチェーン、forwardチェーン、outputチェーンと呼ばれ、個別にフィルタを設定できる。また、フィルタの条件に適合したパケットをどのように処理するかを細かく設定することも可能だ。それぞれのチェーンには複数のフィルタを設定することができるので、これらを組み合わせることで、柔軟性のあるファイアウォールが構築できる。

しかし逆に、この柔軟性がipchainsを複雑なものにしている。文章で説明してもちょっとわからないと思うので実際に編集部のフレッツ・ISDNサーバで設定したフィルタ（**リスト3**）を紹介しながら解説していくことにしよう。**リスト3**で設定している内容は次のようなものだ。

①現在設定されているフィルタをクリアしている。
②inputチェーンのデフォルトポリシーを拒否（DENY）に設定している。デフォルトポリシーは、設定されていないフィルタ（つまり、すべてのフィルタにマッチしない）に対して、どのような処理を行うかの設定だ。DENYに設定しているので、受信したすべてのパケットは拒否されることになる。
③これはloネットワークインターフェイス（つまり、サーバ自身）のパケットは許可（ACCEPT）するように設定している。「-i」オプションはネットワークインターフェイスの指定だ。「-s」は送信元を指定するオプション、「-d」は送信先を指定するオプションで、「0/0」は、すべてのアドレスを指す。「-j」はターゲットを指定するオプションで、ここにパケットを受信した際の処理方法を指定する。
④この指定は、インターネット上（eth0）からLAN側のプライベートIPアドレスを偽装して接続しようとするパケットを拒否する設定だ。
⑤インターネット上からサーバへ到達する各プロトコルを許可している。
⑥すでにセッションが張られた認証済みのセッションであれば、1022以上のポートを使用できるように設定している。「-y」オプションはSYNパケットと呼ばれる、接続要求のみを許可するオプションだ。オプションの前に「!」を付けることで意味が反転し、接続要求以外のパケットを許可していることになる。「1022:」という指定は1022以上のポートという意味だ。

⑦ポート番号1027番はypbindが使用するポートだが、このポートが通過できるように設定していないと、Webページが受信できないので設定している。
⑧LAN側のネットワークインターフェイス（eth1）に対しては、すべてのプロトコルを許可している。
⑨まず最初にforwardチェーンでパケットのルーティングを拒否し、LAN側のローカルIPアドレスのみIPマスカレードを許可している。「MASQ」はIPマスカレードの指定だ。
⑩outputチェーンでは、インターネット上に送信されるパケットをDENYにして、必要なものだけが送信されるように設定している。LAN側のネットワークインターフェイスは無条件ですべてのプロトコルが通過できるように設定し、インターネット側へ出力されるプロトコルはinputチェーンと逆の設定を行い、余計なパケットが送信されないように設定している。

だいたいの概要は理解できただろうか。それでは早速フィルタリングを設定しよう。

まず、**リスト3**を参考にして自分のサーバに合わせたフィルタ、setipchains.shを作成する。次に、rootユーザー権限でこのファイルを/usr/local/sbinにコピーし、実行属性とパーミッションを設定する。

```
# cp ./setipchains.sh /usr/local/sbin
# chmod 500 /usr/local/sbin/setipchains.sh
```

最後にシステム起動時に有効になるように/etc/rc.localの最後に次の1行を追加する。

```
/usr/local/sbin/setipchains.sh
```

以上でipchainsの設定は終了だ。システムを起動したままフィルタリングを有効にしたいのであれば、setipchains.shを実行すればよい。

```
# /usr/local/sbin/setipchains.sh
```

ここで紹介したパケットフィルタリングの設定は解説しやすいように作成したため、汎用性がなく、ちょっとした修正でも大幅に変更しなければならない。今月号の特集2では、汎用性の高いスクリプトも紹介しているので参照してほしい。また、セキュリティやipchains、カーネル2.4でサポートされたiptablesについても触れているので、あわせて参照してほしい。

セキュリティホールのアップデート

ipchainsを使ってフィルタリングの設定を行ったが、フィルタリングで防げないセキュリティホールが存在する可能性がある。どんなにフィルタリングでポートを閉めたとしても、提供するサービスがある限り、どこかのポートを開ける必要がある。そのサービスにセキュリティホールがあれば重大な問題となる。

たとえば、Red Hat Linux 7Jに収録されているbind-8.2.2_P5-25には重大なセキュリティホールが存在している。これは、「Ramem」や「Lion」といったワームに狙われる危険が高い。特にLionに感染するとバックドアを作成されるばかりか、メールでパスワードファイルをchina.com宛に送信されてしまう。こういったセキュリティホールに対する対策は、こまめにセキュリティ情報をチェックし、対応済みのものにアップデートする必要がある。

それではサーバのbindがセキュリティホール対策ずみのものかどうかを調べてみよう。

```
$ rpm -q bind
```

もし、「bind-8.2.2_P5-25」と表示された場合は、セキュリティホールが存在する危険なパッケージだ。すぐにRed HatのFTPサイト（ftp://ftp.redhat.com/pub/redhat/updates/7.0/i386）から最新版の、bind-8.2.3-1.i386.rpmをダウンロードしてアップデートしよう。

```
# rpm -Uvh bind-8.2.3-1.i386.rpm
```

これ以外のセキュリティホールについても日々発見される可能性がある。セキュリティに関する情報は、インターネットから入手することが可能だ。特に、日本のLinux情報（http://www.linux.or.jp/）にあるセキュリティ情報や、JPCERT/CC（http://www.jpcert.or.jp/）などをこまめにチェックし、素早くセキュリティホールを塞ぐようにしなければならない。

Linuxはセキュリティホールが発見されるのも修正されるのも速いが、セキュリティホールの情報が伝達するのも速いということを常に忘れないようにしよう。

終わりに

5回にわたり、フレッツ・ISDNでサーバを構築する方法を紹介してきた。かなり早足で説明してきたので、解説が十分でなかった点などがあったと思うが容赦してほしい。インターネットの常時接続環境はわずか5カ月で大きく変わり、今やADSLを始め新しいサ

ービスが軒並み登場してきた。そういう意味では、64kbpsという通信帯域のフレッツ・ISDNは今となっては厳しいサービスとなってしまった感がある。しかし、通信帯域の違いを除けば、サーバの構築という点ではまったく変わらない。ADSLやそのほかのサービスを選択したとしても、今回の連載特集の内容は転用できるはずだ。

どんなサービスであれ、手軽に常時接続が可能となった今こそ、知識だけでなく実際にサーバを構築してはいかがだろうか。

▼リスト3　ipchainsのフィルタ設定

```
#!/bin/sh

/sbin/ipchains -F                                                                    ①
/sbin/ipchains -P input DENY                                                         ②
/sbin/ipchains -A input -i lo -s 0/0 -d 0/0 -j ACCEPT                                ③
/sbin/ipchains -A input -i eth0 -p tcp -s 192.168.0.0/24 -d 0/0 -j DENY              ④
/sbin/ipchains -A input -i eth0 -p udp -s 192.168.0.0/24 -d 0/0 -j DENY              ④
/sbin/ipchains -A input -i eth0 -p tcp -s 0/0 -d 111.122.133.128/29 domain -j ACCEPT
/sbin/ipchains -A input -i eth0 -p udp -s 0/0 -d 111.122.133.128/29 domain -j ACCEPT
/sbin/ipchains -A input -i eth0 -p tcp -s 0/0 -d 111.122.133.128/29 www -j ACCEPT
/sbin/ipchains -A input -i eth0 -p udp -s 0/0 -d 111.122.133.128/29 www -j ACCEPT
/sbin/ipchains -A input -i eth0 -p tcp -s 0/0 -d 111.122.133.128/29 25 -j ACCEPT
/sbin/ipchains -A input -i eth0 -p udp -s 0/0 -d 111.122.133.128/29 25 -j ACCEPT
/sbin/ipchains -A input -i eth0 -p tcp -s 0/0 -d 111.122.133.128/29 ssh -j ACCEPT
/sbin/ipchains -A input -i eth0 -p udp -s 0/0 -d 111.122.133.128/29 ssh -j ACCEPT
/sbin/ipchains -A input -i eth0 -p tcp -s 0/0 -d 111.122.133.128/29 ntp -j ACCEPT    ⑤
/sbin/ipchains -A input -i eth0 -p udp -s 0/0 -d 111.122.133.128/29 ntp -j ACCEPT    ⑤
/sbin/ipchains -A input -i eth0 -p tcp -s 0/0 -d 111.122.133.128/29 pop-3 -j ACCEPT  ⑤
/sbin/ipchains -A input -i eth0 -p udp -s 0/0 -d 111.122.133.128/29 pop-3 -j ACCEPT  ⑤
/sbin/ipchains -A input -i eth0 -p tcp -s 0/0 -d 111.122.133.128/29 imap2 -j ACCEPT  ⑤
/sbin/ipchains -A input -i eth0 -p udp -s 0/0 -d 111.122.133.128/29 imap2 -j ACCEPT  ⑤
/sbin/ipchains -A input -i eth0 -p tcp -s 0/0 -d 111.122.133.128/29 113 -j ACCEPT    ⑤
/sbin/ipchains -A input -i eth0 -p udp -s 0/0 -d 111.122.133.128/29 113 -j ACCEPT    ⑤
/sbin/ipchains -A input -i eth0 -p tcp ! -y -s 0/0 -d 111.122.133.128/29 1022: -j ACCEPT  ⑥
/sbin/ipchains -A input -i eth0 -p udp -s 0/0 -d 111.122.133.128/29 1027 -j ACCEPT  ⑦
/sbin/ipchains -A input -i eth1 -s 0/0 -d 0/0 -j ACCEPT                              ⑧

/sbin/ipchains -P forward DENY                                                       ⑨
/sbin/ipchains -A forward -s 192.168.0.0/24 -d 0/0 -j MASQ                           ⑨

/sbin/ipchains -P output DENY
/sbin/ipchains -A output -i eth1 -j ACCEPT
/sbin/ipchains -A output -i eth0 -p tcp -s 111.122.133.128/29 -d 0/0 domain -j ACCEPT
/sbin/ipchains -A output -i eth0 -p udp -s 111.122.133.128/29 -d 0/0 domain -j ACCEPT
/sbin/ipchains -A output -i eth0 -p tcp -s 111.122.133.128/29 -d 0/0 www -j ACCEPT
/sbin/ipchains -A output -i eth0 -p udp -s 111.122.133.128/29 -d 0/0 www -j ACCEPT
/sbin/ipchains -A output -i eth0 -p tcp -s 111.122.133.128/29 -d 0/0 25 -j ACCEPT
/sbin/ipchains -A output -i eth0 -p udp -s 111.122.133.128/29 -d 0/0 25 -j ACCEPT
/sbin/ipchains -A output -i eth0 -p tcp -s 111.122.133.128/29 -d 0/0 ssh -j ACCEPT
/sbin/ipchains -A output -i eth0 -p udp -s 111.122.133.128/29 -d 0/0 ssh -j ACCEPT   ⑩
/sbin/ipchains -A output -i eth0 -p tcp -s 111.122.133.128/29 -d 0/0 ntp -j ACCEPT   ⑩
/sbin/ipchains -A output -i eth0 -p udp -s 111.122.133.128/29 -d 0/0 ntp -j ACCEPT   ⑩
/sbin/ipchains -A output -i eth0 -p tcp -s 111.122.133.128/29 -d 0/0 pop-3 -j ACCEPT ⑩
/sbin/ipchains -A output -i eth0 -p udp -s 111.122.133.128/29 -d 0/0 pop-3 -j ACCEPT ⑩
/sbin/ipchains -A output -i eth0 -p tcp -s 111.122.133.128/29 -d 0/0 imap2 -j ACCEPT ⑩
/sbin/ipchains -A output -i eth0 -p udp -s 111.122.133.128/29 -d 0/0 imap2 -j ACCEPT ⑩
/sbin/ipchains -A output -i eth0 -p tcp -s 111.122.133.128/29 -d 0/0 113 -j ACCEPT   ⑩
/sbin/ipchains -A output -i eth0 -p udp -s 111.122.133.128/29 -d 0/0 113 -j ACCEPT   ⑩
/sbin/ipchains -A output -i eth0 -p tcp ! -y -s 111.122.133.128/29 -d 0/0 1022: -j ACCEPT  ⑩
```

箱の中のペンギンたち

文：みわよしこ
Text : Yoshiko Miwa

第4回　組み込みLinuxベンチャー企業のあり方

「企業」とひと口に言いますが、規模も性格もいろいろです。数万人規模でがっちりした組織体制を持っている株式会社も、2〜3人の人員で柔軟な体制で動いている小さな有限会社・合資会社も「企業」です。しかしながら、業種・規模などによって、企業に求められるものは千差万別です。

今回はレーザーファイブ（東京都文京区）が行っている組み込みLinux事業を軸に、組み込みLinux事業を行っている小規模ベンチャー企業に求められるものを追ってゆきます。

レーザーファイブは1999年8月に設立された、社員数32名（2001年3月現在）の典型的な小規模ベンチャー企業です。

ベンチャー企業にはスピードが必要

レーザーファイブの代表取締役CEOである窪田敏之氏が組み込みLinux事業への進出を考え始めたのは2000年初頭のことで、1999年8月に会社を設立してから約半年が経過し、軌道に乗ってきた時期でした。

窪田氏は2000年3月にはもう資金集めを開始しました。収益を見込んで、先に事業を開始することも不可能ではありません。しかし窪田氏は、将来性は十分でも、まだ海のものとも山のものともわからない組み込みLinux事業に乗り出すにあたって、「海図のない道を行くのだから、まず兵糧が必要だった」と考えたそうです。

当時、市場に出回っている資金が比較的潤沢だったこともあり、いろいろな会社に出向いてプレゼンテーションをすると、関心を抱いてもらうことができ、何社ものスポンサーと8億円近い資金を得ることができました。

窪田氏はそれから人的体制を整備し、2000年5月には技術者2名、営業2名の事業体制を整備し、組み込みLinux事業を開始しました。

一般的に、ベンチャーにはスピードが求められますが、事業の検討、資金調達から体制構築までをわずか3カ月程度で行ったこの例も例外ではありません。

人に依存する要素

ベンチャー企業の特徴のひとつとして、「その人の個人的な個性や能力に依存する要素が比較的高い」ということが挙げられます。

大企業の場合には、「そのポスト、その職務に誰が就いても、同等のサービスを提供し続けられる」ということが要求される場面が多いため、個人の個性や能力よりは組織運営・組織体制のほうが大きな意味を持つと言えます。反面、ベンチャーの場合では、むしろ組織よりは個人といった面が大きいようです。

窪田氏はレーザーファイブのCEOですが、組み込みLinux

事業部長も兼任しています。自分は「組み込みLinux事業のディレクター」であり、組み込みLinux事業は「やっていて面白いの一言に尽きる」とのことです。ディレクターとして、時には技術者と一緒に実際の開発に関わることもあるそうです。

このようなスタイルは、NECのTK-80時代にマイコンに触れて以来、変化するアセンブラやアーキテクチャにずっと触れ続けて現在に至り、組み込みシステムに関して半可通ではない知識と豊かな経験を持っている窪田氏という人物の個性を抜きにしては考えられないでしょう。ベンチャー企業の事業が成功するためには、まずは「人」と言えそうです。

ベンチャーに求められる人物像は「ポジティブ・オタク」

ではここで、ベンチャーに求められる人物像を考えてみましょう。ベンチャーといっても多種多様ですが、ここでは組み込みLinuxシステム開発を行うベンチャーに話を限定することにします。

ベンチャー企業には専門性と技術力が必要です。何らかの専門性、独自性、それを形にするための技術力がなければ、ベンチャー企業は成立しません。これはベンチャーに限ったことではありませんが、中小ベンチャー企業の場合、専門性・独自性・技術力の必要性は、大企業よりも相対的に大きいと言えるでしょう。

さらに中小ベンチャーの場合、一人一人の社員の専門的能力や個性の重要性は、大企業社員に比べると非常に高くなります。場合によっては、「その人がいなくなったらその仕事は続行できなくなる」ということも珍しくありません。

もちろん、これは事業の運営のあり方としては、非常に望ましくないことです。誉め言葉として「余人をもって代えがたい」と言いますが、仕事を依頼する側の立場になってみれば、その仕事の代替要員がいないことは大きな不安要素となります。いかに専門性や個性が求められるからといって、ほかの人でまったく代替が利かないような専門家集団では、長期的には企業活動は成功しないでしょう。やろうと思えばほかの仕事もできる柔軟性のある専門家の集団、少しずつ専門がダブっている専門家の集団が、ベンチャー企業の理想的なあり方であろうと思われます。

図　大企業とベンチャーとの組織形態の違い

また、コミュニケーション能力の問題も無視できません。専門的能力が高く個性的な専門家の一部には、ほかの分野の人と会話が成立しないような「閉じこもった」専門性に偏るケースが見受けられます。小人数のベンチャー企業ではしばしば、技術者が営業の場に出ていったり、逆に営業担当者が技術的な話をする必要に迫られたりすることがありますので、このような「閉じこもった」専門家も、ベンチャーには適さないと言えるでしょう。

結局のところ、「協調性あるマニア」あるいは「ポジティブ・オタク」が、ベンチャーに求められる人物像と言えそうです。

技術だけでは企業活動は成立しない

組み込みLinuxシステムを扱うベンチャー企業に技術力は不可欠ですが、もちろん技術者だけでは企業活動は成立しません。営業担当者がいなかったら、誰が技術者の仕事を確保してくれるのでしょうか。経理担当者がいなかったら、誰が資金を管理してくれるのでしょうか。広報担当者がいなかったら、技術者たちの優れた仕事を誰が社会にアピールしてくれるのでしょうか。経営者がいなかったら、誰がその企業の舵を取るのでしょうか。

これらは、もちろんベンチャー企業に限ったことではありませんが、中小ベンチャー企業の場合には各人の仕事ぶりが見えやすいぶんだけ、それぞれの役割の重要性も明確になりやすいようです。

特に、少ない人員で企業活動を行っている中小ベンチャー企業の場合は、たとえば「技術者が営業的視点を持つ」「経理担当者が経営的視点を持つ」「経営者が技術的視点を持つ」といった複眼思考、また一人でいくつもの役割をこなすことが、社員一人一人に求められることが多いようです。

前回のインタビューで紹介した「りぬくす工房」の海老原祐太郎氏は、自分の役割を「ハードウェア開発・庶務・営業・企画・経理・そして社長」だと述べていました。一人でいくつもの役割をこなす必要があるということです。少なくとも、「技術者は技術のことだけ理解していればよい」という考え方は、ベンチャー企業では通用しないでしょう。

ベンチャー企業には他社とのアライアンスが不可欠

海老原氏は、「同業・非同業の数多くの企業との協業の重要性」についても述べていました。「りぬくす工房」は社員数2名のごく小規模な企業ですので、他社との協業を行わなければ、企業活動はまったくできません。

社員数32名のレーザーファイブでも、他社との協業体制を非常に重要視し、アライアンス（同盟関係）を作るための組織作りといった活動を積極的に行っています。

昨年10月に「リアルタイムLinux協議会」が設立された際に中心となって動いたのは、レーザーファイブと三技協でした。

そして現在、レーザーファイブは、同年7月に設立された「日本エンベデッドリナックスコンソーシアム（Emblix、連載第1回で会長の中島達夫氏のインタビューを掲載）」と「リアルタイムLinux協議会（RTL協議会）」の両方の理事企業となっています。

現在、Emblixは「家電など広い立場での組み込みLinux」を、RTL協議会では「リアルタイム制御ロボットなど、細かい技術的分野に特化し、Linuxのリアルタイム応用をより使いやすくするための研究」という分業・協業体制ができつつあります。このような政治的な活動が円滑に行われることは、技術者たちにとっても消費者にとっても重要なことです。なぜなら、このような政治的活動の成果や人的ネットワークは、組み込みLinux開発者によって必ず利用され、より良い製品、よりユニークな製品の開発につながってゆくからです。

もちろん政治的活動としての組織作りだけではなく、業務そのもの、技術そのものに関しても他社との協業関係は重要です。レーザーファイブは、組み込みLinux事業のコンピュータ技術の側面だけに関しても、大企業をも含む数社と協業体制を築いています。

窪田氏は「いろいろな企業のいろいろな人と話すことによって、事業を先に進めてゆくことが面白い」と語っています。アライアンスの構築や企業間政治の問題と真剣に取り組むことは、ベンチャー企業だからこそ求められることかもしれません。そしてそこには、「オープンソース」の思想のもと、全世界の協業によって発展してきたLinuxのあり方と通じるところがないでしょうか？

小規模ベンチャー企業の生きる道

ある程度以上の規模の企業の場合、「他社より優れた製品を作る」「他社より多く販売する」という競争、いわゆる「シェア競争」から自由になることは難しいでしょう。ごく最近でよく知られている例では、ソニーの「PlayStation」

とセガの「Dreamcast」のゲーム機対決が挙げられます。

しかし小規模ベンチャー企業の場合は、逆にむしろ「シェア」という発想を捨てることが重要なようです。窪田氏によれば、「自分たちは、他社と同じことをやらないから、生きのびて行ける」のだそうです。では、他社と同じことをやらずに何をやればいいのでしょうか？

これに対しては「いかにニッチを探すかが勝負」とのことです。ニッチ（隙間）の部分には、大きく成長するタネがいくつか必ずあります。それを捉えて事業化すれば、事業を大きく成長させることもできます。

シェア競争、つまり規模の大小を競う競争をやらないことの鍵は、「個性化」にあるようです。「ギャップを感じたらそれがチャンス」「あれ、おかしいな？　がビジネスチャンス」なのだそうです。何にギャップを感じ、何に「あれ、おかしいな？」と問題意識を感じるかは、まさにその個人、その企業の「個性」によるでしょう。どのようなニッチを探しあてられるかは、結局は個性の問題なのです。

りぬくす工房の海老原氏も、「これなら自分の技術でソリューションを提供できる」というものを探し、それを自分のビジネスとしていました。

結局、ベンチャー企業が自らの存在価値を高めていけるかどうかは、いかに自社が個性化し、いかに自社にふさわしい社会のニッチを探してゆけるかどうかにかかっているようです。さまざまな個性や、特技を持つ数多くの個人の協業によって開発されたLinuxのあり方と、これもまた通じているように筆者には思えます。

画面　レーザーファイブの組み込みLinuxのWebサイト（http://www.laser5.co.jp/embedded.html）

Column

レーザーファイブの組み込みLinux製品

レーザーファイブの組み込みLinux事業は、現在のところマイクロサーバ向け製品から展開されています。Linuxを利用したマイクロサーバ製品は、「需要はあるが供給はない」とCEOの窪田氏が考えたことから、そのような展開となりました。

具体的な製品としては、今年2月26日に発売された名刺サイズの小型ボードコンピュータ「L-Card＋」、近日発売予定のボードコンピュータ「L-Board」があります。いずれもOSとしてLinuxを搭載しています。

「L-Card＋」「L-Board」の特徴は、Linuxを採用しているため、カスタマイズが容易であること、採用したチップが低消費電力型であるため組み込み用途で使用しやすいことです。各種モバイル端末への応用、自販機・計測機器などへのシステム組み込みのほか、超小型サーバ向けの利用が見込まれています。

写真1　レーザーファイブ組み込みLinux事業部の作業風景

写真2　L-BoardへのLinuxのポーティング（移植）作業風景

レーザーファイブ代表取締役CEO 窪田敏之氏に聞く

Interview

**一般Linuxユーザーにとって、レーザーファイブは「LASER5 Linux」のディストリビューターというイメージが強いようですが、同社の活動はディストリビューション事業にとどまりません。
今回は、小規模ながらユニークな企業活動で高く評価されているレーザーファイブの代表取締役CEOである窪田敏之氏にインタビューをお願いしました。**

――まずは、どうして現在のような企業活動をされる方になったのかをお話しいただきたいんです。まず、レーザーファイブを設立される以前、1984年に五橋研究所を設立されたあたりの経緯を伺えますでしょうか？

窪田：まず、僕はもともと歯科医師だったんです。

――え？

窪田：そうなんですよ（笑）。東京医科歯科大学の大学院の3年目の時に、友人7人と会社を作ったんです。博士号取得後開業したんですが、歯科医院をやりながらパソコンのソフトを書いたり、大型計算機の仕事のおこぼれを貰ったりしていました。もともと、五橋研究所はそういった何か役に立つ製品やサービスを研究開発するために設立した会社なんです。現在はデジタルエックス線装置をメインに弟が経営していますが……。

――そうなんですか。

窪田：ええ、僕は凝り性で、歯科医師も結構真面目にやったつもりなんですけれども（笑）。たとえば非常に早い時期から感染防止や滅菌を徹底してやっていました。B型肝炎ウイルスとか、エイズウイルスとか。最近も、狂牛病のプリオン[*1]を非活性化するノウハウをいろいろ試してみたりしています。プリオンの非活性化自体は、濃い水酸化ナトリウムに器具を数分浸積すればできるんですけど、それをやると精密な治療器具がボロボロになってしまうんですよね。それを防ぐにはどうしたらいいかといった研究課題は尽きません。

写真　レーザーファイブ代表取締役CEO　窪田敏之氏（くぼたとしゆき）
1958年生まれ。東北大学歯学部卒。歯博。東京医科歯科大学大学院在学中の1984年、株式会社五橋研究所を設立し、コンピュータ関連事業を展開。1993年には日本初の日本語化Linuxディストリビューションを発売し、1996年には日本初のLinux専門誌「Linux Japan」を創刊。1999年、オープンソースビジネスを展開するレーザーファイブ株式会社を設立し、現在は代表取締役CEO。

――ディープな世界ですね（笑）。ところで、私にはどうも「歯科医師とコンピュータ関連ベンチャー経営者」が結びつかないのですが、どうしてレーザーファイブの設立に至られたんでしょうか？

窪田：コンピュータそのものが好きだったんです。もともと僕は子供のころからラジオ少年で、小学生のころは真空管でラジオを組み立てたりしていたんですが、トランジスタが出回り始めて、「こんなに小さいのに、ちゃんと増幅するんだ！」と感動しました。

――私が子供のころは、真空管の時代はすでに終わってトランジスタ時代でしたが、それでもICが出回り始めた時には「これ1つでラジオができてしまうんだ！」と感動しました。

窪田：ええ、僕もそこでまたもや感動しました（笑）。高校生のころには、VCOやオペアンプを使ってアナログシンセサイザを作ったり、文化祭の自主制作8mm映画用にテープレコーダとのシンクロ装置を作ったりしていましたね。卒業後、父が開業歯科医だったし、親類の一族がほとんど医師か歯科医師なので、自分も歯学部に進学したんです。ところがやっぱり電子工作が好きで。大学4年か5年のころにTK-80が発売されて、「面白そうだけど、ちょっと高すぎて買えないな」と思っていたら、たまたま回路図が手に入ったんです。それで、友人と一緒にクローンを作って、アセンブラで遊んだりしました（笑）。大学（東北大学）の工学部には当時、有名な西澤潤一先生[*2]がいらして、電子工学の講義にもぐり込んだりもしましたよ、歯学部の学生なのに（笑）。

――そういったことが現在につながっているのですね。

窪田：そうですね。五橋研究所で1993年ごろから始めた

Linux関連事業*3がふくらんで、オープンソース事業を中心とするレーザーファイブの設立につながりました。

――話は少し戻るのですが、1984年に五橋研究所を設立されてから、企業活動は順調でしたか？

窪田：変動はかなりありました。収入的には、年収が400万円以下だった年もあります。僕も家族も、お金がなければそれなりの楽しみを見つけられるタイプなのでそんなに苦にはなりませんでしたが、それでも家族4人がそれで生活していたのですから、大変ではありましたね。

――起業はその仕事が本当に好きでないとやれないことですね。

窪田：今の日本は何をやったって食えるんですから、仕事は面白いほうがいいに決まっています。

――会社生活になんとなく違和感があるけれども、起業に踏み切るにはちょっと……という方が世の中にはたくさんいますが、そういう方に相談されたら、窪田さんはどうアドバイスされますか？

窪田：「考え込むのはやめなさい。考え込むくらいなら、宮仕えのプロになりなさい」と言います。

――正論ですが、「それができれば苦労はないよ！」という反応がありそうですね。

窪田：だったら独立すればいいんです。

――ですが、簡単にできるようでできないことですから……。

窪田：そうですね。独立することになんとなく感情的なしこりがあるんだったら、独立はやめるべきです。けっこう辛いこともあるし、そういう時は好きでないと続けられません。それに、失敗したら後悔しますから、自分の感情を優先すべきです。今の勤務先に違和感があって、だけど独立には抵抗があるんだったら、宮仕え先を選び直せばいいんです。そして宮仕えするのなら、宮仕えのプロになるべきです。

――おっしゃる通りです。起業にも適性がありますしね。

窪田：ええ、歯科医として、勤務医の道を選んだ人も、開業した人もたくさん見てきて思ったのですが、開業したり独立したりして成功する人は、あまり深く悩んでいないんです。

――「くよくよ悩む人は起業には向かない」と。

窪田：そうです。それと、貧乏が怖い人も起業には向きませんね。お金は使わないと集まって来ませんから。会社を大きくできる人には、「会社のお金は丁寧に大切に使うけれども、自分のお金の使い方には太っ腹」という特徴がありますね。

――「いかにして会社の経費を自分の飲み代に使うか」といった、一般的なサラリーマンの発想とは逆ですね。

窪田：ええ、お金だけが目当てで起業すると、往々にしてうまくいきません。お金はあくまで「結果」や「おまけ」と考えるべきです。お金が目当てなら会社に勤務して、勤務先、すなわち宮仕えの相手を選ぶべきです。

――ところで話は変わりますが、Linuxに限らず、組み込みシステムの現在や将来をどのように考えていらっしゃいますか？

窪田：組み込み製品は、専用機化の時期と汎用機化の時期を繰り返してきているんです。たとえば、うどんも、そばも、パンも粉を練って作るものですよね。そしてそれぞれに専用の練り機がある。アプライアンス（特定用途向けの専用装置）です。

――そういう見方ができますね。

窪田：「粉を練る」という機能に注目すると、では汎用の練り機を作って、製品別にソフトウェアでそれぞれに必要な機能を実現してスイッチで切り替えればよいという発想になります。

――汎用機の発想ですね。

窪田：ところがあまりにも機能が多くなりすぎると、使いにくくなるわけです。そうなるとまた、「麺向けとパン向けに専用機化しよう」という流れが出てくるわけです。

――ソフトウェアではなく、ハードウェアでという方向ですね。

窪田：ええ、これと同じように組み込みシステムは、ソフトウェア主体とハードウェア主体の時期が繰り返されているんです。現在はどちらかというとハードウェア主体、アプライアンス、専用機化の時期なのかなと僕は見ています。

――Linuxによって、組み込みシステムの世界が変化するということはあるでしょうか？

窪田：あるでしょうね。組み込みシステムの世界で育ってきた人は、地に足がついているんです。仕様があって、プロトタイプがあって、求められる「一品料理」を作るという仕事を連続してやってきた人達ですから、技術やマシンに対する皮膚感覚があるんです。一方で、オープンソースコミュニティの人たちは、どのマシンかわからなくてもハッキングができます。空から地上を見下ろす鳥の目を持っています。今は過渡期ですけど、この2つが合わさると強いでしょうね。

――そうですね、数年後が楽しみです。ところで、レーザーファイブの数年後はどうでしょうか？

窪田：今は大変小さな会社ですが、規模の面でも技術面でも「立派な会社」を目指しています。具体的には上場も考えています。

――こちらも楽しみですね。本日はお忙しいところ、ありがとうございました。

*1　プリオン　狂牛病の病原体とされる特殊な蛋白質で、クロイツフェルト・ヤコブ病の病原体と考えられている。
*2　西澤潤一（にしざわじゅんいち）。東北大学名誉教授、現岩手県立大学長。静電誘導トランジスタ、光ファイバなどの発明者として知られる。
*3　株式会社五橋研究所は、1993年11月に 日本初の日本語対応Linuxパッケージ「Linux+JE」を発売した。

短期集中連載

JDK 1.3に対応したLinux対応のJavaビジュアル開発ツール

JBuilder 4 Foundation

最終回　データベースとの連携2

文：加藤大受
Text：Daiju Kato　dkato@jcom.home.ne.jp

インターネット上の通販サイトなどでは、Java技術を利用して動的なHTMLを生成しています。最終回にあたり、データベースに格納されているデータを動的にHTMLドキュメントに生成するサーブレットについて説明します。

サーブレットとは、サーバ上で動作するJavaアプリケーションです。サーブレットエンジンという環境の上で動作し、ブラウザからのリクエストに応じてHTMLドキュメントなどを生成する技術です。動的にHTMLドキュメントを生成するという意味では、Perlなどを利用したCGI（Common Gateway Interface）とよく似ていると思われます。また、技術的にはマイクロソフトの技術である、Active Server Pages（ASP）とよく似ています。

サーブレットとCGIはよく比較されますので、少しCGIとの違いについて説明していきましょう。

サーブレットは、サーブレットエンジンと呼ばれる環境の上で動作するJavaプログラムです。CGIと異なり、ブラウザからのリクエストに応じてプロセスを起動するのではありません。つまり、毎回プログラムが起動されるのではなく、いったんプログラムをサーブレットエンジン上で起動させると、明示的にアンロードされるか、あるいはサーブレットエンジンが終了するまでサーブレットエンジン上に存在します。サーブレットのライフサイクルは**図**のようになります。ブラウザからのリクエストがあると、サーブレットエンジンが対象となるクラスファイルをロードします。そのあと、ロードされたクラスファイルが動作し、クライアント向けにHTMLドキュメントを生成します。クラスファイルは動作終了後は待機状態となり、再びブラウザからリクエストがあると動作します。サーブレットエンジンが終了するか、明示

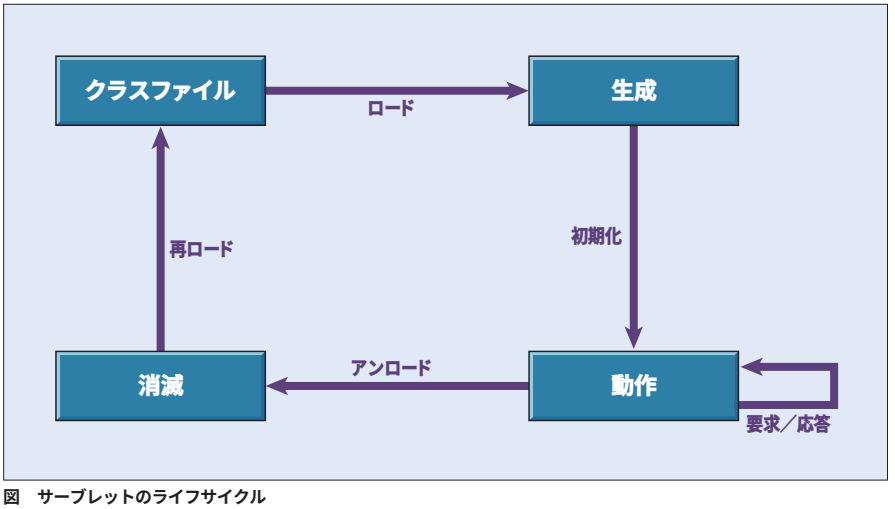


図　サーブレットのライフサイクル

的にアンロードされると、サーブレットのライフサイクルが終わります。

CGIと比較した場合、一度ロードされてしまえば、動作が終了しても待機状態になるため、レスポンスが速いという特徴があります。また、サーブレットはデフォルトでマルチセッションに対応しているため、同時に複数のリクエストがあっても対応することができます。さらにサーブレットは、Javaの提供する膨大なAPIを使用することができます。たとえば、データベースと連携できるJDBC APIや、Javaの提供する画像変換のAPIなどを利用することができ、サーブレットにさまざまな機能を追加していくことが可能になります。このため、インターネット上の通販サイトなどでは、Java技術を利用してWebシステムを構築しています。

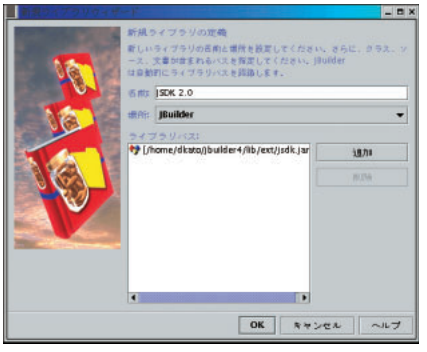
サーブレット環境の構築

サーブレットの概念を簡単に説明しました。では、具体的にサーブレットはどのように使用するかを説明していきましょう。サーブレットを作成するには、サーブレットAPIというものを使用します。このAPIは、当初、Java Servlet Development Kit 2.0（JSDK 2.0）に含まれていたのですが、Java2 Development Enterprise Edition（J2EE）の登場後は、J2EEの中に含まれています。J2EEをSunのWebサイトからダウンロードして使用してもいいですが、簡単なサーブレットを作成するのであれば、JSDK 2.0をダウンロードして使用するのがいいでしょう。また、JSDK 2.0に含まれている、Java Servlet API 2.0を使用したサーブレットをLinux上で動作させるには、Apache JServを使用するのが一番安いので、ここではJSDK 2.0を使用していくものとします。

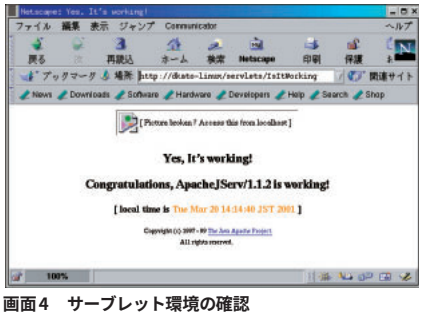
それではまず、SunのWebサイトからJSDK 2.0をダウンロードしてください。Linuxで使用するにはUNIX版をダウンロードします。このファイルは、JBuilder 4 Foundationでサーブレットを作成するときに使用します。

ダウンロードが完了したら、ファイルを解凍して、JBuilderのlib/extディレクトリにコピーしてください（**画面1**）。続いてJBuilderを起動し、JSDK 2.0のライブラリの設定を行います。［ツール］－［ライブラリ設定］で表示される「ライブラリ設計ダイアログボックス」の［新規］ボタンをクリックして、先ほどコピーした「jsdk.jar」を設定しましょう（**画面2**）。これで、JBuilderからJSDK 2.0を使用することが可能になりました。続いて、サーブレットが動作する環境を作成します。

Linuxでサーブレットを使用するには、Apache JServを使用するのが安価だと説明しました。Apache JServは、Apache Software Foundationが開発しているサーブレットエンジンで、



画面2　JSDK 2.0の設定



画面4　サーブレット環境の確認

```
$ uncompress jsdk20-solaris2-sparc.tar.Z          ← tar形式に解凍
$ tar xvf jsdk20-solaris2-sparc.tar -C /usr/local  ← /usr/localに解凍
$ cp /usr/local/JSDK2.0/lib/jsdk.jar /home/<ユーザー名>/jbuilder4/lib/ext  ← jbuilderのlib/extにコピーする
```
画面1　JSDK 2.0の解凍とコピー

```
# export PATH=/usr/java/jdk1.3.0_01/bin:$PATH   ← javaコマンドのあるディレクトリにパスを設定
# rpm -ivh ApacheJServ-1.1.2-1.i386.rpm         ← Apache JServ 1.1.2をインストール
# /etc/rc.d/init.d/httpd restart                ← Apacheが起動している場合はリスタートする
```
画面3　Apache JServ 1.1.2のインストール

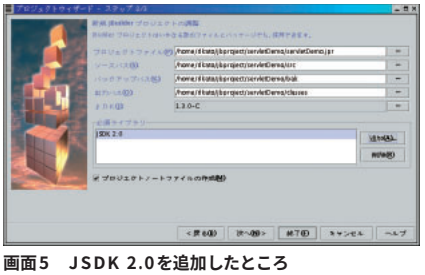
JSDK 2.0のダウンロードサイト	http://www.java.sun.com/products/servlet/archive.html
Apache JServ 1.1.2のダウンロードサイト	http://java.apache.org/jserv/dist/

表1　ダウンロードサイト一覧

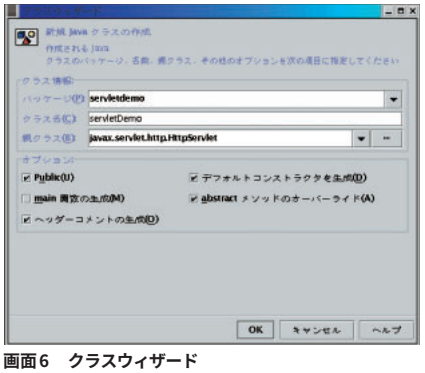
Apacheと非常に親和性が高いことが特徴です。Apache JServのWebサイトからRPM形式のApache JServ 1.1.2をダウンロードし、Linuxの環境にサーブレットの環境を作成してみましょう。ダウンロードしたApacheJServ-1.1.2-1.i386.rpmをインストールします（**画面3**）。このとき、javaコマンドのあるディレクトリにパスを設定しておくと、自動的にjavaコマンドを見つけてくれます。RPMファイルをインストールしたらApacheを再起動し、サーブレットが動作しているかを確認します（**画面4**）。確認は「http://＜サーバー名＞/servlets/IsItWorking」というURLで行います。筆者の環境ではJBuilderを使用しているマシンとは別の環境にサーブレットの環境を構築しています。

HTTPコマンド	メソッド
GET	doGet
POST	doPost
OPTIONS	doOptions
PUT	doPut
DELETE	doDelete
TRACE	doTrace
SERVICE	Service

表2　HTTPコマンドとHttpServletクラスのメソッド



画面5　JSDK 2.0を追加したところ



画面6　クラスウィザード

サーブレットAPIについて

サーブレットAPIは、HTTPプロトコルのコマンドに従い動作するように設計されています。HttpServletクラスは、サーブレットを作成するときに基礎となるクラスで、このクラスにはHTTPのコマンドが送られて来たときに実行されるメソッドが用意されています（**表2**）。つまり、クライアントからGETのコマンドが送られてくると、HttpServletクラスを継承したクラスのdoGetメソッドが実行され、何らかのHTMLドキュメントが生成されるというわけです。

JSDK 2.0ではHTTPプロトコルのみに対応していますが、最新のサーブレットAPIでは、FTPなどのプロトコルにも対応しており、クライアントのリクエストに応じて実行されるWebシステムを、すべてサーブレットで作成することが可能です。サーブレットというと難しいイメージがありますが、基本的にHTTPのGETコマンドが来たら、doGetメソッド内の処理をしていくということに留意しながらコーディングしていけば、それほど悩まずにサーブレットを書いていくことができるでしょう。

▼リスト1　サーブレットに必要なパッケージ

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
```

簡単なサーブレットの作成

それでは単純に、ブラウザからのリクエストが来たら、文字列を返すサーブレットを書いてみましょう。ブラウザでURLが入力されると、HTTPのGETコマンドがWebサーバに送信され、相手先がサーブレットの場合であればdoGetメソッドが実行されます。このdoGetメソッド内でHTMLドキュメントを生成するコードを書いてやればいいのです。

それでは早速作成してみましょう。［ファイル］－［新規プロジェクト］

▼リスト2　doGetメソッドのソースコード

```
public class servletDemo extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException
    {
            ///コンテントタイプの設定
            res.setContentType("text/html");
            //ドキュメントを生成するPrintWriterオブジェクトの作成
            //日本語を使用するためにエンコード処理が必要
            PrintWriter out = new PrintWriter(
               new OutputStreamWriter(res.getOutputStream(), "EUCJIS"));
            //HTMLヘッダーの生成
            out.println("<HTML><BODY>");
            //Hello, World!の生成
            out.println("サーブレットのデモ!<BR>");
            //HTMLドキュメントの終了
            out.println("</BODY></HTML>");
            //ドキュメント生成終了
            out.close();
    }
```

を選択し、プロジェクトウィザードを起動します。プロジェクト名を「servletDemo」に設定し、ステップ2/3に移動します。必須ライブラリの［追加］ボタンをクリックして、先ほど設定したJSDK 2.0を追加します（**画面5**）。これで、このプロジェクトでJSDK 2.0が使用可能になります。続いて、［ファイル］－［新規クラス］を選択し、クラスウィザードを起動します。クラス名を「servletDemo」にし、親クラスに「javax.servlet.http.HttpServlet」と入力します（**画面6**）。これで雛形ができました。

次に、サーブレットを使用するうえで必要なパッケージを追加します（**リスト1**）。続いて、servletDemoクラスを修正します。まずdoGetメソッドを定義します。doGetメソッドはHttpServletRequestという、Webサーバからのリクエストを受け取るオブジェクトと、HttpServletResponseという、Webサーバへのレスポンスを送信するオブジェクトの2つを引数に持ちます。doGetメソッド内では、まず最初にHTTPのコンテンツタイプを設定します。ここではHTMLドキュメントを生成しますので、「"text/html"」となります。続いて、HTMLドキュメントを生成するために必要なPrintWriterオブジェクトを生成します。なお、日本語を生成するときは必ずエンコードと呼ばれる作業が必要になりますので、いったん出力ストリームに格納し、エンコードして送信します。ここではEUCJISでエンコードします。続いて、HTMLタグを送信していきます。HTMLドキュメントの送信が終了したら、PrintWriterオブジェクトを閉じます。これで、クライアントに動的なHTMLコンテンツを送信することができます。**リスト2**はdoGetメソッドのソースコードです。Javaの文法に従いHTMLドキュメントを生成していることが理解できると思います。

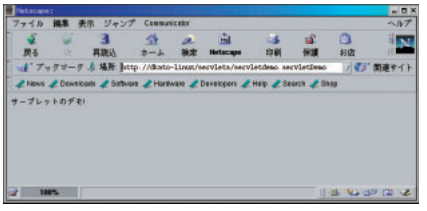
コンパイルが成功したら、サーブレットを実行してみましょう。クラスファイルを/home/httpd/servlets以下にコピーします。このとき注意しなければならないのがディレクトリ構造です。先ほど作成したservletDemoクラスはservletdemoパッケージに属します。そのため、/home/httpd/servlets/の下にservletdemoというディレクトリを作成し、servletdemo.classファイルをコピーしなければなりません。コピーしたら、ブラウザを起動し、「http://＜サーバー名＞/servlets/servletdemo.servletDemo」と入力して実行してみましょう。**画面7**のようにブラウザ上に「サーブレットのデモ！」という文字列が表示されるはずです。

▼リスト4　パッケージの追加

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
import interbase.interclient.*;
import java.sql.*;
```

フィールド	内容
MSG_ID	メッセージID
INP_NAME	名前
INP_DATE	入力日
E-MAIL	メールアドレス
MSG	メッセージ

表3　作成するデータベースのフィールド



画面7　servletDemoの実行

コメントページの作成

それでは、データベースと連動したサーブレットを作成してみましょう。作成するプログラムは、Web上でコメントを入力できるページとします。まず、コメントを格納するデータベース

```
# /opt/interbase/bin/isql -I /tmp/
schema.sql
```

画面8　isqlコマンドでスキーマファイルを実行

▼リスト5　グローバル変数の定義と、InterBaseへの接続

```
public class msgPage extends HttpServlet {

  //グローバル変数の定義
  java.sql.Connection con = null;
  java.sql.ResultSet rs = null;
  interbase.interclient.DataSource ds = new
    interbase.interclient.DataSource();

  public void init(ServletConfig config) throws ServletException {
    super.init(config);
    try {
        //InterBaseへの接続を実施
         iCInit();
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
```

▼リスト3　スキーマファイル(schema.sql)

```
/*      データベースの作成        */
CREATE DATABASE '/home/db/dbservlet.gdb'
        USER 'SYSDBA'
        PASSWORD 'masterkey'
         PAGE_SIZE 8192
         DEFAULT CHARACTER SET EUCJ_0208;

/**** 作成したデータベースへの接続  ****/
SET NAMES EUCJ_0208;
CONNECT '/home/db/dbservlet.gdb' USER 'SYSDBA' PASSWORD 'masterkey';

/*        テーブルの作成          */
/******     メッセージテーブル     *****/
CREATE TABLE MSG (
       MSG_ID   CHAR(10)  NOT NULL PRIMARY KEY,       /* メッセージID */
       NAME     CHAR(80)  ,                           /* 名前         */
       INP_DATE DATE      DEFAULT 'now',              /* 入力日       */
       E_MAIL   CHAR(80)  ,                           /* e-mail       */
       MSG      CHAR(200)                             /* メッセージ   */
       );
/******   テーブル作成のコミット処理  *****/
COMMIT;

/*    ジェネレータの作成          */
/******MSG_IDを生成するジェネレータ ***/
CREATE GENERATOR GEN_MSG_ID;

/******  ジェネレータ作成のコミット処理  *****/
COMMIT;

/*        トリガーの作成          */
/******  MSG_IDを設定するSET_MSG_ID ****/
SET TERM !!;
CREATE TRIGGER SET_MSG_ID FOR MSG
   BEFORE INSERT
   AS
     DECLARE VARIABLE MAX_ID INTEGER;
   BEGIN
     /*現在の最大値を取得*/
     SELECT MAX(MSG_ID) FROM MSG INTO MAX_ID;
     IF (MAX_ID IS NOT NULL) THEN
        BEGIN
          NEW.MSG_ID=MAX_ID+1;
        END
     ELSE
        BEGIN
          NEW.MSG_ID=1;
        END
   END
!!
SET TERM ;!!

/******   プロシージャ作成のコミット処理  *****/
COMMIT;
```

を作成しましょう。前回と同様に、InterBaseを使用します。ここではInterBaseの使い方は説明しませんので、本誌InterBase 6.0の記事、またはInterBaseのWebサイト（http://www.interbase2000.com/）を参照してください。

データベースはEUCJISをデフォルトキャラクタセットに設定し、MSGというコメントを格納するテーブルを作成します。必要となるフィールドは**表3**の通りです。

ここで、MSG_IDと入力日は自動的に入力されるものとします。MSG_IDの入力はジェネレータとトリガーを使用します。今回使用するスキーマを作成するSQL文は**リスト3**のようになります。

スキーマファイルをisqlユーテリティで実行して、/home/db以下にdbservlet.gdbというデータベースを作成します（**画面8**）。

データベースを作成しましたので、続いてサーブレットの作成に入りましょう。ここでは次の2つのサーブレットを作成します。プロジェクト名は「dbServlet」、パッケージ名は「dbservlet」とします。

- **ページを生成するサーブレット msgPage.java**
- **コメントを書き込むサーブレット writeMsg.java**

前回説明したように、JavaからInterBaseに接続するにはInterClientを使用します。今回もJDBC 2.0レベルのAPIを使用しますので、JDBC 2.0 Optional Packageを使用します。プロジェクトウィザードで今回のプロジェクトを作成するときに、JSDK 2.0とJDBC 2.0 Optional Packageのライブ

ラリを追加してください。

それではまず、コメントページを生成するmsgPageクラスを作成してみましょう。クラスウィザードを起動して、サーブレットの雛形を作成します。msgPageクラスでは、InterClientとSQL文を使用しますので、**リスト4**のパッケージを追加します。続いて、前回Javaアプリケーションからデータ処理を行ったときと同じように、データベースとの接続を管理するオブジェクトなどをグローバル変数として定義します。なお、msgPageクラスのinit()メソッドでInterBaseへの接続を行います。前回と同様に別メソッドを作成し、そのメソッド内でInterBaseへの接続を行います（**リスト5**）。

iCInit()メソッドは、前回作成したICInit()メソッドをベースにしています。異なる点は、データベース名とデフォルトキャラクタセットの指定です。今回のdbservletデータベースは、デフォルトキャラクタセットがEUCJISなので、EUCJISを指定して接続を行います。この指定を行わないと正しくデータを書き込むことができません。必ずサーバのキャラクタセットと、プログラム側のキャラクタセットを一致させてください。このiCInit()メソッドは、msgPageクラスがサーブレットエンジンにロードされると実行されるinit()メソッドから呼び出されるので、ロードされたタイミングでデータベースに接続しますので、あとは終了するまでつながったままとなります（**リスト6**）。

続いてクライアントから呼び出されたときに実行されるdoGetメソッドを作成します。このメソッド内で、HTMLドキュメントの生成を行うわけですが、今回は次の処理を行う必要があります。

- **コメントを書き込むためのHTMLドキュメントの作成**
- **すでに書き込まれたコメントをHTMLドキュメントで生成する**

これらの処理についてもwriteCommentPart()メソッドとreadComment()メソッドで受け持つ形とし、できるだけdoGet()メソッド本体は読みやすくしておきましょう。doGet()メソッドで行う内容は、HTMLコンテンツのヘッダ、タイトルの生成、writeCommentPart()メソッドの呼び出し、readComment()メソッドの呼び出し、HTMLコンテンツのフッタ部の生成となります（**リスト7**）。

▼リスト6　InterBaseへの接続

```
private void iCInit() throws Exception {
  //InterClientドライバの定義
  try {
      //サーバー名の設定
      ds.setServerName("dkato-linux");
      //パスの設定
      ds.setDatabaseName("/home/db/dbservlet.gdb");
      //キャラクターセットの設定
      ds.setCharSet("EUCJIS");
      //InterBaseに接続を実施(ユーザー名、パスワード)
      con = ds.getConnection("SYSDBA","masterkey");
      }
  catch (SQLException ex){
      }
}
```

▼リスト7　doGet()メソッド

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
{
      //コンテントタイプの設定
      res.setContentType("text/html");
      //ドキュメントを生成するPrintWriterオブジェクトの作成
      //日本語を使用するためにエンコード処理が必要
      PrintWriter out = new PrintWriter(
          new OutputStreamWriter(res.getOutputStream(), "EUCJIS"));
      //HTMLヘッダの生成
      out.println("<HTML>");
      out.println("<META http-equiv='Content-Type' content='text/html;"
                +" charset=EUCJIS'>");
      out.println("<title>サーブレットとデータベースを使用したのデモ</title>");
      out.println("</head>");
      //コメント入力部分の生成
      writeCommentPart(out);
      //既存コメント生成
      readComment(out);
      //ドキュメント生成終了
      out.println(
          "copyright(c) 2001 Daiju Kato e-mail:dkato@jcom.home.ne.jp");
      out.println("</HTML>");
      out.close();
}
```

writeCommentPart()メソッドでは、コメント入力部のHTMLドキュメントを生成します。ドキュメントを生成するためには、PrintWriterオブジェクトが必要となりますので、引数としてこれを受け取っています。ここでは、「Send」ボタンが押されたときに、コメントを書き込むwriteMsgクラスを呼び出すようにします。**画面9**は完成したmsgPageクラスのサーブレットを実行したところです。コメント入力部分が作成されていることがわかると思います。**リスト8**がこの入力部分を生成するwriteCommentPart()メソッドのソースとなります。HTMLタグを送信してコメント入力部分を作成していることが理解できるでしょう。こういった部分は、一度HTMLファイルを作成してからソースコードに埋め込むと間違いなくソースを記述することができます。

コメント入力部分が終了したら、既存のコメントを読み込むreadCommentメソッドを作成します。ここでは、SQL文を生成して、既存のコメントを1行ずつ読み込み、HTMLドキュメントとして生成します。データのある限り読み込むことになるのでwhileループを使用します（**リスト9**）。

コメントページのサーブレットができましたので、続いて書き込みのページを作成します。書き込みのページでは、受け取った名前、メールアドレス、コメントをデータベースに書き込みます。途中までの流れはpageMsgクラスと一緒になるので、送信された内容を受け取る部分から説明していきます。今回は、POSTイベントに対応することになりますので、doGetメソッドではなくdoPostイベントにコードを記載していきます。

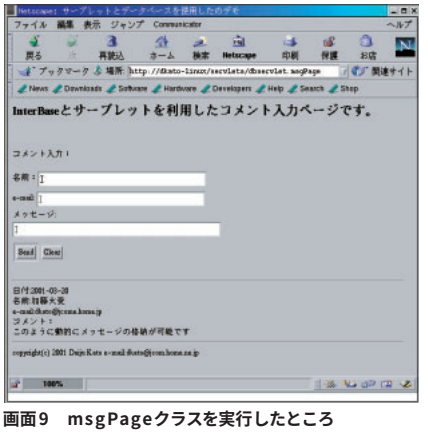
POSTイベントで送信された名前、メールアドレス、コメントはHTMLで

▼リスト8　writeCommentPart()メソッド

```
private void writeCommentPart(PrintWriter out){
  //コメント入力部分の生成
  out.println("<H1>InterBaseとサーブレットを利用したコメント入力ページです。</H1>");
  out.println("<BR><BR>");
  out.println("コメント入力：<BR>");
  out.println("<form action='/servlets/dbservlet.writeMsg' method='POST'>");
  out.println("名前：<input type='text' size='48' name='inp_name'><br>");
  out.println("e-mail: <input type='text' size='48' name='from'><br>");
  out.println("メッセージ: <br><input type 'text' size='60' name='msg'><br>");
  out.println("<input type='submit' value='Send'>");
  out.println("<input type='reset' value='Clear'>");
  out.println("</form><HR>");
}
```

▼リスト9　readComment()メソッド

```
private void readComment(PrintWriter out) {
  try {
        //SQL文の発行の準備
        java.sql.Statement stmt = con.createStatement();
        rs = stmt.executeQuery("SELECT * from MSG");
        while (rs.next()) {
            out.println("日付:"+rs.getString("INP_DATE")+"<BR>");
            out.println("名前:"+rs.getString("NAME")+"<BR>");
            out.println("e-mail:"+rs.getString("E_MAIL")+"<BR>");
            out.println("コメント：<BR>");
            out.println(rs.getString("MSG")+"<BR><HR>");
        }
        rs.close();
  } catch (SQLException ex) {
    ex.printStackTrace();
  }
}
```



画面9　msgPageクラスを実行したところ

▼リスト10　コメントの受け取り

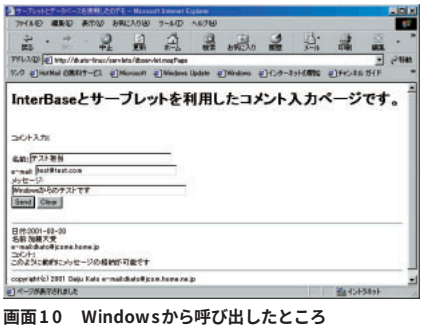
```
      //送信されたコメントの受け取り
       inp_name = new String(req.getParameter("inp_name").getBytes("8859_1"),"EUCJIS");
       inp_email= new String(req.getParameter("from").getBytes("8859_1"),"EUCJIS");
       inp_msg  = new String(req.getParameter("msg").getBytes("8859_1"),"EUCJIS");
```

指定された変数の中に格納されます。それぞれの変数から取り出してやればいいのですが、そのまま取り出すと文字化け状態となってしまいますので、必ずエンコード処理が必要となります。送信ページからEUCJISで送信されてきますので、EUCJISで取り出してやります。取り出すには、PrintWriterオブジェクトでいったん出力ストリームに入れる場合と同様に、バイト単位で文字列を取り出して、EUCJISにエンコーディングします（**リスト10**）。

送信ページから送られた内容をベースにSQL文を作成し、データベースに格納します。doPostイベントからデータの書き込みを行うwriteComment()メソッドを呼び出して書き込みを行います。**リスト11**はwriteCommentイベントのソースコードです。送信された内容を出力したあと、SQL文を作成してデータベースに対してSQL文を実行します。今回は結果セットを伴わないINSERT文を送りますので、使用するメソッドはexecuteUpdateメソッドとなります。書き込みに失敗した場合を想定してエラー処理を入れておきましょう。

サーブレット環境へのコピー

コメント入力ページの作成と、コメントの書き込みページの作成が終了したら、サーブレット環境を作成します。



画面10　Windowsから呼び出したところ

今回のプログラムでは、さまざまなパッケージを使用しています。JServでは、サーブレットAPI以外のクラスおよび、jarファイルなどを使用する場合には環境設定が必要となります。クラスパスの設定はjserv.propertiesというファイルで行います。このファイルは、/etc/httpd/conf/jserv/以下にインストールされています。こちらのファイルを開いて、CLASSPATHの設定部分にJDBCを使用するためのパッケージ、InterClientを使用するためのパッケージ、JDBC 2.0を使用するためのパッケージを追加します（**リスト12**）。

この設定を行わないとInternal Errorとなり、サーブレットがまったく起動しなくなりますので注意してください。

jserv.propertiesの設定が終了したら、/home/httpd/servlets以下に作成したクラスファイルをコピーします。パッケージ名と同じ名前のディレクトリを作成し、その下に格納してください。サーブレットのコピーが終わったら、Apacheを再起動します。これで環境設定は終了です。ブラウザからアクセスして動作を確認してください。Windowsから使用しても正しく動作します（**画面10**）。

▼リスト11　writeComment()メソッド

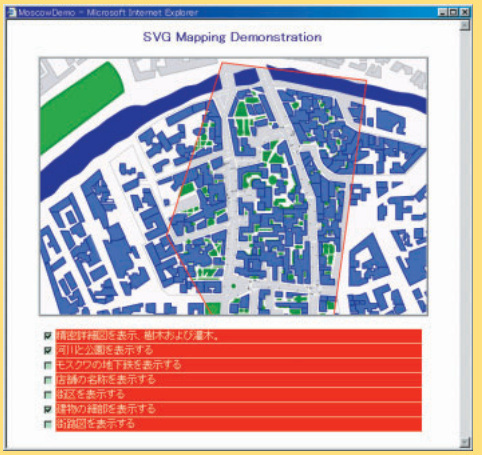
```
private void writeComment(PrintWriter out)  {
  out.println("<B>名前:</B>"+inp_name+"<BR>");
  out.println("<B>e-mail:</B>"+inp_email+"<BR>");
  out.println("<B>メッセージ:</B>"+inp_msg+"<BR>");
  //interBaseへの書き込むためのSQL文の作成
  String sql="INSERT INTO MSG (NAME,E_MAIL,MSG) VALUES ("
             +"'"+inp_name+"','"+inp_email+"','"+inp_msg+"')";
  try {
        //SQL文の実行
        stmt.executeUpdate(sql);
        out.println(
        "<H2>メッセージの書き込みありがとうございました</H2><BR>");
  } catch (SQLException e) {
       out.println("<H2>書き込みに失敗しました</H2><BR>");
       out.println(e.getMessage());
  }
  out.println("<HR>");
  out.println("<A HREF='/servlets/dbservlet.msgPage'>[戻る]</A>");
}
```

▼リスト12　/etc/httpd/conf/jserv/jserv.propertiesファイル（抜粋）

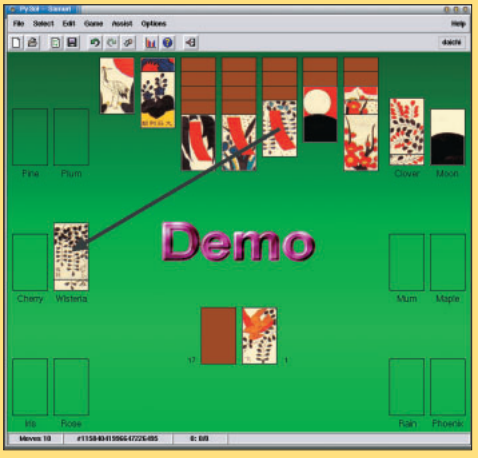
```
# CLASSPATH environment value passed to the JVM
<..........途中略..........>
#
wrapper.classpath=/usr/lib/apache/ApacheJServ.jar
wrapper.classpath=/home/httpd/classes/servlet-2.0.jar
#JDBCを使用するための設定
wrapper.cpasspath=/usr/local/jdk1.3.0_01/jre/lib/rt.jar
#InterClientを使用するための設定
wrapper.classpath=/usr/interclient/interclient.jar
wrapper.cpasspath=/usr/interclient/interclient-core.jar
#JDBC 2.0を使用するための設定
wrapper.classpath=/home/db/jdbc2_0-stdext.jar
```

Free Application Showcase

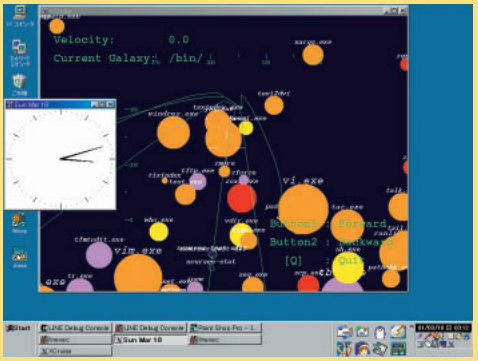
文：出井 一
Text : Hajime Dei



Sodipodi　P.137



PySol　P.140



LINE　P.144

※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対応の次世代ドローソフト

Sodipodi

バージョン：0.22　　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ており、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ライブラリが必要となる。

●SVGとは

スケーラブル・ベクタ・グラフィックス（SVG）は、ベクタ図形やイメージ、テキストで構成される2次元グラフィックを表現する言語だ（**画面1**）。W3Cが提唱するオープンな規格であることや（**画面2**）、XMLで定義されていて可読性や可搬性が高いことから、次世代のWebページで広く使われることが期待される。

現在は、W3Cにより勧告候補が出されている段階で、仕様が完全に固まっているわけではない。それでも、SVGに対応したドローソフトやコンバータ、IE／Netscape用のプラグインなどがいくつか登場している。MozillaにSVG表示機能を内蔵させる試みも行われているようだ。

なお、現在のSodipodiがサポートするのはSVGのサブセットに過ぎず、アニメーションやフィルタ、スクリプトなどは含まれない。また、ビットマップ画像による塗りつぶしなど、ダイアログにボタンが用意されているものの、まだ実装されていない機能もある。このため、他のソフトで作成したSVGファイルをSodipodiで読み込むと、正常に表示や編集できないことがあるので気をつけよう。

●ビルドとインストール

最初にライブラリをインストールする。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なら、日本語対応版がインストール済みのはずだ。Sodipodiのビルドには、gnome-print-develパッケージも必要なので、こちらも忘れずにインストールしておこう。

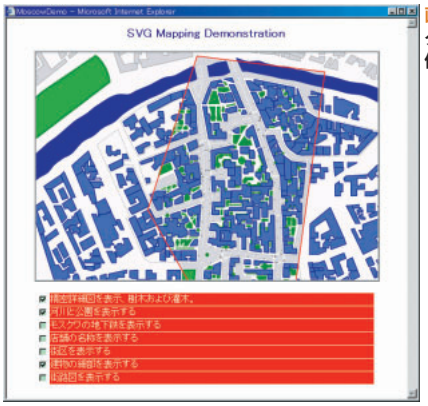
GNOME Applicationライブラリ（GAL）については、GNOMEのFTPサイトで配布されるtarボールからRPMパッケージを作成しよう。「su -」としてrootになってから、「rpm -ta gal-0.5.tar.gz」とすると、/usr/src/redhat/RPMS/i386にgalとgal-develパッケージが作られる。これらを「rpm -Uvh gal-＊0.5-1.i386.rpm」としてインストールすればいい。

これで準備が整ったので、Sodipodiのビルドとインストールを行う。tarボールを展開後、「./configure」→「make」でビルド、「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。

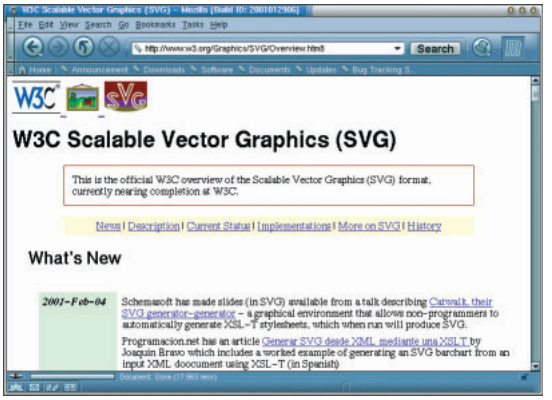
なお、configureスクリプトによって作成されるSPECファイル（sodipodi.spec）は不完全なので、そのままではRPMパッケージを作成できない。「%changelog」セクションの日付を「＊Fri Mar 9 2001」に変更し、「%files」セクションに「%{prefix}/share/locale/＊/LC_MESSAGES/sodipodi.mo」を追加する必要がある。

●Sodipodiの起動と画面構成

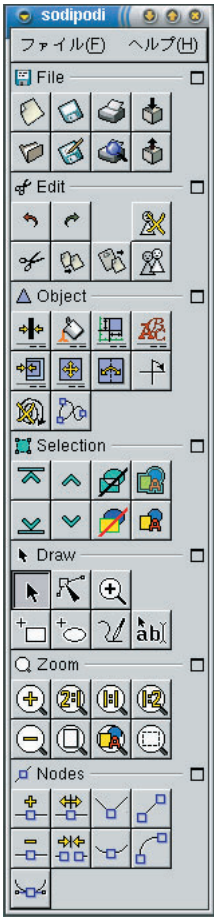
ktermなどのコマンドラインで「sodipodi&」と入力するか、GNOMEメニューの[プログラム]－[グラフィック]－[Sodipodi]を選択すると、たくさんのボタンが並んだメインウィンドウが開く（**画面3**）。日本語カタログが付属するため、日本語環境ではメニューやダイアログ、チップヘルプのほとんどが日本語で表示される。



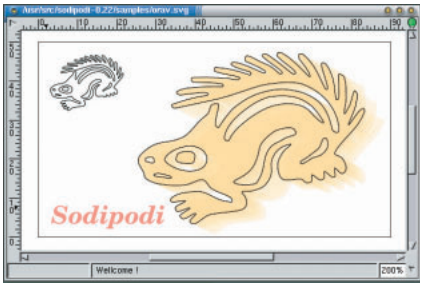
画面1　IE用Adobe SVG Viewerプラグインでモスクワの地図を表示した例。



画面2　SVGはW3Cで標準化が行われているオープンな規格だ。



画面3　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。



画面4　付属のサンプル画像（リスのOrav）はSodipodiのシンボルだ。

画面5　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。

メインウィンドウのインターフェイスは洗練されている。各ボタンはFile、Edit、Objectなど機能別のグループに分類されており、境界線のクリックで表示の有無を切り替え可能だ。また、境界線の右端にある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。

Fileグループ左上の[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可能だ。試しに、付属のサンプル画像を読み込んで操作してみるといいだろう（**画面4**）。

メインウィンドウのボタンのほか、画像上での右クリックでポップアップするメニューからもさまざまな操作を行える。たとえば、表示倍率の変更は、Zoomグループのボタンの代わりに、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。

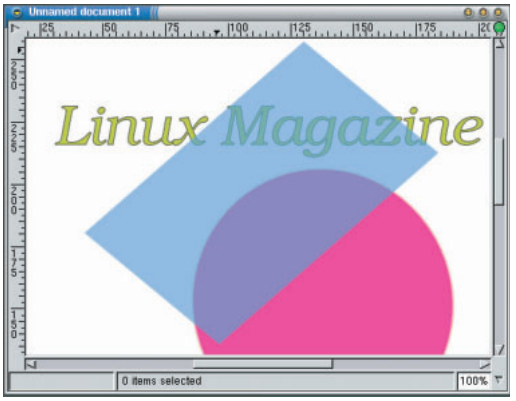
なお、表示を拡大したために画像全体がウィンドウ内に収まらなくなったら、スクロールバーの操作や、画像上での中ボタンドラッグにより、表示する範囲を変更できる。

●オブジェクトの描画と選択

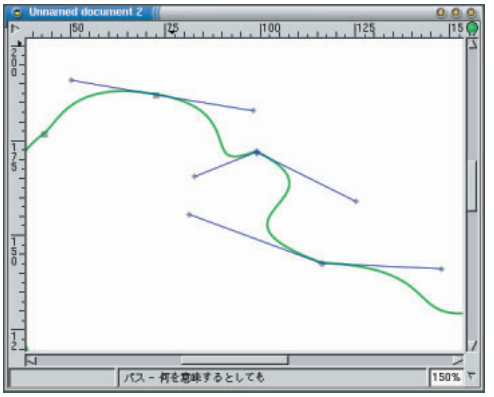
Sodipodiで描画できるオブジェクトは、矩形・楕円・フリーハンド線・テキストの4種類のみ。Drawグループ下段の4つのボタンでオブジェクトの種類を選択し、画像上でのドラッグにより描画を行う（テキストのみクリック後に文字列を入力）。複数のオブジェクトの連続描画も可能だ。

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[ノード編集]ボタンを押すと、オブジェクトをクリックで選択できるようになる。また、ドラッグやShift－クリックによる複数オブジェクトの選択もサポートされる。

選択したオブジェクトに対しては、内部のドラッグによる移動や、周囲に表示される8つのボックス（■）のドラッグによるサイズ変更が可能だ。再



画面6　塗りつぶしのAlphaパラメータを変更すれば半透明表示も可能。



画面7　フリーハンド線の各ノードのハンドルで曲がり具合を調整できる。

度オブジェクトをクリックすると、今度は中心を含む9つのボックスに切り替わり、これらをドラッグすることで回転や変形を行える。

●オブジェクトのプロパティを変更

オブジェクトの太さや色などを変更するには、Objectグループ上段の4つのボタンでプロパティ設定用のダイアログを開く。

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」（**画面5**）、位置などを設定する「レイアウト」、テキストのフォントや内容（日本語不可）を設定する「フォントスタイル」が用意されている。

たとえば、塗りつぶしのプロパティダイアログで、不透明度を表わすAlphaパラメータを1未満の数値に変更してみよう。設定した数値に応じて、背後のオブジェクトが透けて見えるようになる（**画面6**）。

オブジェクトの前後関係は、Selectionグループのボタンで変更可能だ。もちろん、複数のオブジェクトをひとまとめに扱うグループ化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4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、フリーハンド線（SVGの「パス」に相当）を使えば、ほとんどの図形を実現できる。フリーハンド線には適当な間隔で「ノード」が設定されており、描画後にこれらを操作することで、形を整えたり、大きく変形させることが可能になる。

対象となるフリーハンド線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（◇と□）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7**）。

さらに、Nodesグループのボタン（**画面8**）により、ノードの追加や削除、分割などの操作も行える。なお、端点の結合や直線／曲線への変換といった操作では、隣接する2つ以上のノードを、あらかじめShift－クリックで選択しておく必要がある。

このほか、矩形／楕円／テキストをフリーハンド線に変換することも可能だ。対象となるオブジェクトを選択してから、Objectグループの[選択したオブジェクトを曲線状に変換]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簡単に作成できる（**画面9**）。

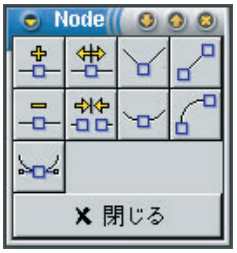
●印刷とファイルへの出力

Sodipodiで作成した画像は、Fileグループの[描画の保存]ボタンでSVG形式で保存され、後で読み込むことができる。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。

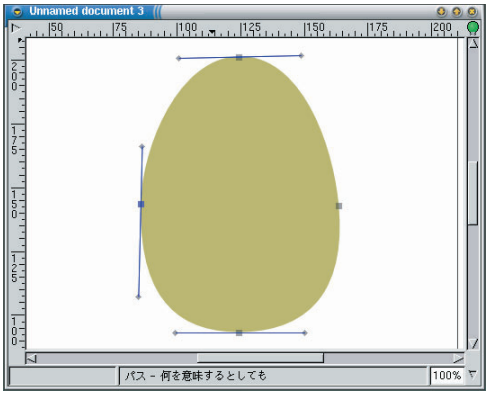
Fileグループの[描画印刷のプレビュー]ボタンを押すと、gnome-printのプレビューウィンドウが開いて、印刷イメージを確認できる（**画面10**）。表示の拡大・縮小や、印刷ダイアログの呼びだしも行える。

印刷ダイアログでは、PS形式に変換し、lpr経由でプリンタに出力するほか、いったんPS形式やPDF形式でファイルに保存し、gvやXpdfなどで表示・印刷することも可能だ。

また、Fileグループの[エクスポート]ボタンを使うと、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、この方法を利用するとよいだろう。



画面8　ノードの追加や削除、分割などの操作に使われるNodesグループ。



画面9　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。

画面10　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。

200種類以上ものソリティアを集めたカードゲーム

PySol

バージョン：4.70　　ライセンス：GPL

http://pysol.tsx.org/

PySolは、バリエーション豊かなカードゲームだ。バージョンアップのたびにゲームの種類が増え、最新版の4.70ではなんと202種類にも及ぶトランプや花札の一人遊び（ソリティア）をプレイできる。BGM演奏用のサウンドサーバや、さまざまなデザインのカードを追加するカードセットもオプションで用意されている。実行にはPythonとTcl/Tkが必要だ。

●ビルドとインストール

PySolは、ファイル一式をtar＋gzipしたtarボールで配布されている。インタプリタ言語Pythonで記述されているため、ビルドの必要はない。インストールは、tarボールの展開先で「su」としてrootになってから、「make install」とする。

続いて、別途配布されているサウンドサーバと音楽データ、カードセットの導入を行う。なお、これらがなくてもPySolは実行できるので、以下の処理を飛ばしても構わない。

サウンドサーバを導入すると、MODやMP3によるBGM再生が可能になる。ビルドには、SDLとsmpegライブラリが必要だ。tarボールの展開先でsrcディレクトリに移動後、「./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。

SDLなどのない環境では、ビルド済みのスタティックリンク版サウンドサーバを利用しよう。tarボールを展開後、「su」でrootになってから、「cp pysol soundservermodule.so /usr/lib/python1.5/site-packages」としてコピーすればいい。

BGM用の音楽データは、サウンドサーバとは別に配布されている。tarボールを展開後、「su」でrootになってから、「cp -p data/music/＊ /usr/local/share/pysol/4.70/music」としてコピーしよう。

最後はカードセットだ。PySol本体にもいくつか付属するが、別途配布のtarボールには、さらに多くのカードセットが含まれている。tarボールを展開後、「su」としてrootになってから、「cp -af data/cardset-＊ /usr/local/share/pysol/4.70」としてコピーすればいい。

●ゲーム中の操作

「pysol&」として起動すると、BGMの演奏とともに、クロンダイクのカードが配られてプレイできる状態になる（**画面1**）。次回からは、終了前にプレイしていたのと同じ種類のゲームが自動的に選択される。

別の種類のゲームに切り替えるには、[Select]メニューから選択する。200種類以上のゲームが含まれるため、さらにいくつかのサブメニューで分類されている。実際にプレイすることのできるプレビュー画面から選択することも可能だ（**画面2**）。はじめのうちは、[Popular games]以下にある「クロンダイク」「フリーセル」「スパイダー」などのメジャーなゲームをプレイするとよいだろう。

プレイ中の操作には主にマウスを利用する。置札（タロン）をクリックして新しいカードを引き、場札（タブロー）や台札（ファウンデーション）までドラッグでカードを移動させるのが基本的な操作だ。また、台札へのカードの移動にダブルクリックや右ボタンクリック、他のカードに一部隠されているカードの確認に中ボタンクリックが使えることも覚えておこう。

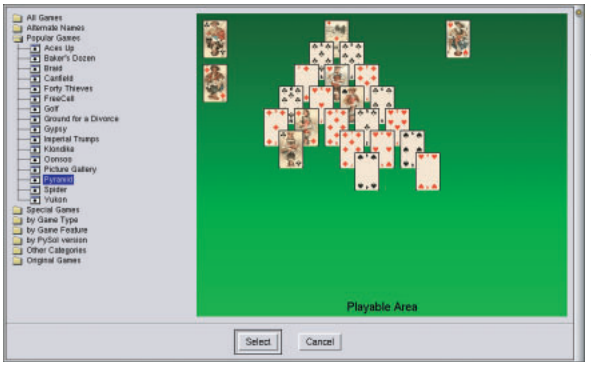
さらに、ツールバーのボタンやショートカットキーを活用すると、プレイの手間を大幅に省ける。たとえば、ゲームの終盤、残りのカードを台札に移動すればクリアできる場合は、ツールバーの[Auto drop cards]ボタン（またはAキー）を押せばいい。台札に移動可能なすべてのカードをPySolが自動的に移動してくれる。

また、同じ配置でゲームをやり直したり、現在の状態をファイルに保存／再開することも可能だ。クリア時に表示されるCongratuation画面（**画面3**）を目指してがんばってほしい。

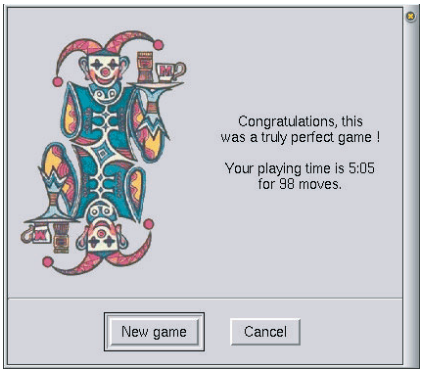
●よく知らないゲームをプレイする

ルールがよくわかっていないゲームに挑戦するには、ツールバーの[Rules for this game]ボタンを押して、ルールの解説を参照する（**画面4**）。簡潔な英語なので読むのはそれほど難しくない。なお、ポピュラーなゲームのバリ

画面1　クロンダイクなど200種類以上のゲームをプレイできる。



画面2　試しにゲームをプレイしてみることも可能なプレビュー画面。



画面3　みごとにクリアすると、このCongratuation画面が表示される。

エーションの場合、基本ルールとの相違点だけが書かれている。

ルールはうろおぼえでも、とにかくプレイしてみたいという人は、操作をやり直せるアンドゥ機能（ツールバーのボタンかZキー）、移動できるカードを教えてくれるヒント機能（[Assist]－[Hint]かHキー）、コンピュータが代わりにプレイするデモ機能（[Assist]－[Demo]かCtrl－Dキー）などの機能を使うとよいだろう（**画面5**）。

ただし、ヒントやデモで示される手は、単にカードを動かせるというだけで最善手ではない。特に、難易度の高いゲームでは手詰まりになりやすいので注意されたい。

●豊富なカードセットを楽しもう

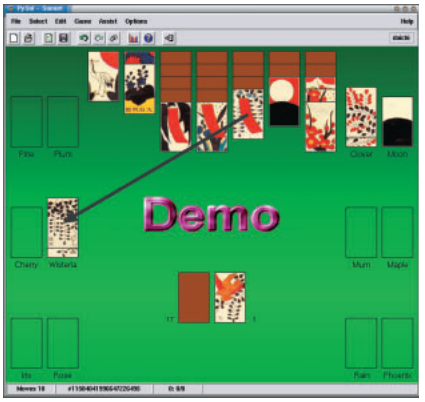
[Options]メニューの項目により、カードの裏面のデザインやテーブルの色などを変更できる。たとえば、裏面のデザインを変更するには、[Options]－[Card background]の一覧表示中から選択すればいい。変更した設定は自動的に保存される。

カードのデザインをまとめて変更するには、[Options]－[Cardset]（またはEキー）でカードセットを選択する（**画面6**）。さまざまなカードが大きさや年代などによって分類されており、プレビュー画面を眺めながら選択可能だ。たとえば、画面の狭いノートパソコンでは、サイズ別の「Tiny」や「Small」に分類された、小さめのカードを使うとプレイしやすい。

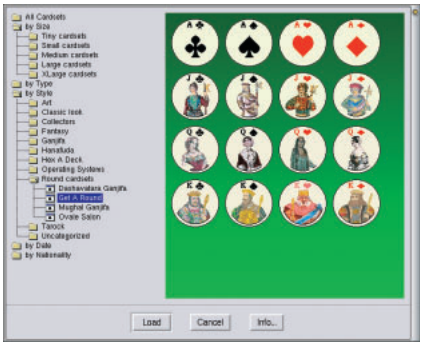
PySol本体には8種類、別途配布されるものを含めると80種類ものカードセットが用意されている。丸いカードの「Get A Round」など、バリエーション豊かなカードセットは見ているだけで飽きない。ただし中にはデザイン重視で、読みやすさはいまひとつのカードもあり、プレイ中に数字などが見にくいこともあるので注意しよう。

なお、花札のカードセット（「Kintengu」など）は、「Matukiri」（松桐）をはじめとする花札ソリティア（8種類）専用だ。通常のゲームで使うと、当然のことながらカードの数字がわからないためプレイできない。

画面4　よく知らないゲームは、まずルールの確認からはじめよう。



画面5　デモ機能を利用してゲームを自動的に進めることも可能だ。



画面6　プレイに使用するカードセットをプレビューしながら選択できる。

日本語に対応したPDFビューア

Xpdf

バージョン：0.92　　ライセンス：GPL

http://www.foolabs.com/xpdf/xpdf.html
http://www.neuroinformatik.ruhr-uni-bochum.de/ini/PEOPLE/rmz/t1lib/t1lib.html (t1lib)

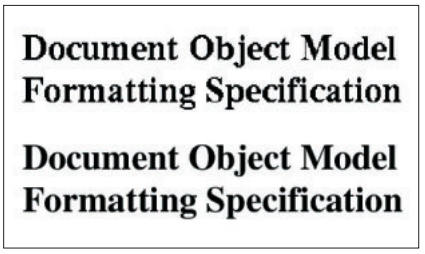
Xpdfは、アドビシステムズの提唱するポータブル・ドキュメント・フォーマット（PDF）で書かれたPDFファイルを表示するビューアだ。ビルド時の設定によって、日本語を含むPDFファイルにも対応する。表示の拡大／縮小はもちろん、テキストの検索やリンクにも対応しており、PS形式のファイルを出力することも可能だ。実行にはフォント描画用ライブラリのFreeTypeとt1libが必要となる。

●ビルドとインストール

まずは、ライブラリをインストールする。TrueTypeフォントの描画処理を行うFreeTypeは、最近のディストリビューションならXと一緒にインストールされるはずだ。なお、Xpdfのビルドにはfreetype-develパッケージも必要となるので、忘れずにインストールしておこう。

次に、Type1フォントの描画処理を行うt1libを導入する。Webサイト（上記URLを参照）で配布されているtarボールを展開後、「./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。

Xpdfはtarボールのみ配布されている。日本語を含むPDFファイルに対応するには、「./configure --enable-japanese --enable-a4-paper」とする必要がある。その後の手順はt1libとまったく同じだ。



画面1　スケーラブルフォントを利用することでジャギーが解消される。

●スケーラブルフォントの設定

そのままの状態でXpdfを使うと、文字の表示にビットマップフォントが使われるため、表示を拡大すると（あるいは大きなサイズの文字があると）、斜め線のギザギザ（ジャギー）がめだってしまう（**画面1上**）。

これに対し、スケーラブルフォントであるType1フォントやTrueTypeフォントを使うようにXpdfに指示すると、拡大してもジャギーのない滑らかな表示が可能だ（**画面1下**）。

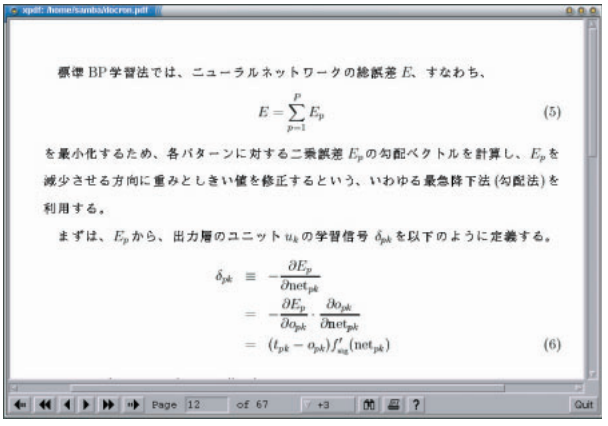
フォントの指定は、Xのリソースデータベースで行う。具体的には、各ユーザーのホームディレクトリの.Xdefaults（または.Xresources）に、**リスト1**の内容を追加すればいい。英数字はGhostscript付属のType1フォント、日本語はTrueTypeの渡辺明朝フォントを指定している。このほか、最後の行では、URLクリックに対するブラウザ設定も行っている。

なお、.Xdefaultsや.Xresourcesの内容は、Xの起動時にリソースデータベースに追加される。Xを起動した後で変更した内容は、「xrdb -merge .Xdefaults」とするまで有効にならないので注意しよう。

●Xpdfの起動と基本操作

ktermなどのコマンドラインで「xpdf&」とすると、Xpdfのウィンドウが開く。右クリックメニューの[Open]かO（オー）キーでダイアログが開くので、PDFファイルを選択しよう。

また、起動時のコマンドライン引数で、「xpdf hoge.pdf&」のようにPDFファイルを指定したり、「xpdf



画面2　Xpdfでは、日本語を含むPDFファイルも表示可能だ。

画面3　内部・外部リンクは青い矩形で表示される。右下の情報にも注目。

hoge.pdf 15&」のように最初に表示するページを指定することもできる。

ウィンドウには、PDFファイルの最初のページ（あるいは指定したページ）が表示される。説明通りにビルドしていれば、日本語を含んだPDFファイルも表示可能だ（**画面2**）。

操作にはマウスとキーボードの両方を利用できる。たとえば、ページのスクロールはカーソルキーか中ボタンのドラッグ、ページの切り替えはPageUp／PageDownキーかウィンドウ左下のボタングループのクリックで行う。

ページを順番に読んでいくにはスペースキーを使うといい。現在のページの末尾までスクロールし、末尾に達すると自動的に次のページへの切り替えを行ってくれる。

また、ウィンドウ下部のZoomリストから拡大率（-5～+5の11段階）を選択することで、ページの表示を拡大／縮小できる。ウィンドウの横幅に合わせたり（fit width）、ページ全体を表示する（fit page）ことも可能だ。

●リンクや検索、印刷にも対応

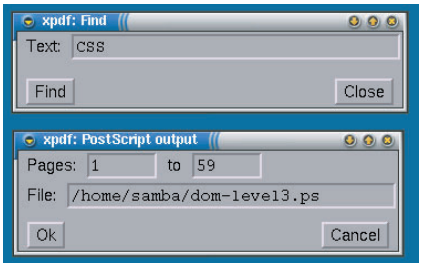
PDFには、文書の内部やインターネットのWebページへのリンクを埋め込むことができる。Xpdfは、こうしたリンクを青い矩形で表示し、リンク先の情報（URLなど）をウィンドウ右下に表示する（**画面3**）。

文書内部のリンクは、目次ページの章タイトルなどに設定されており、クリックすると、Xpdfの表示がリンク先のページに切り替わる。

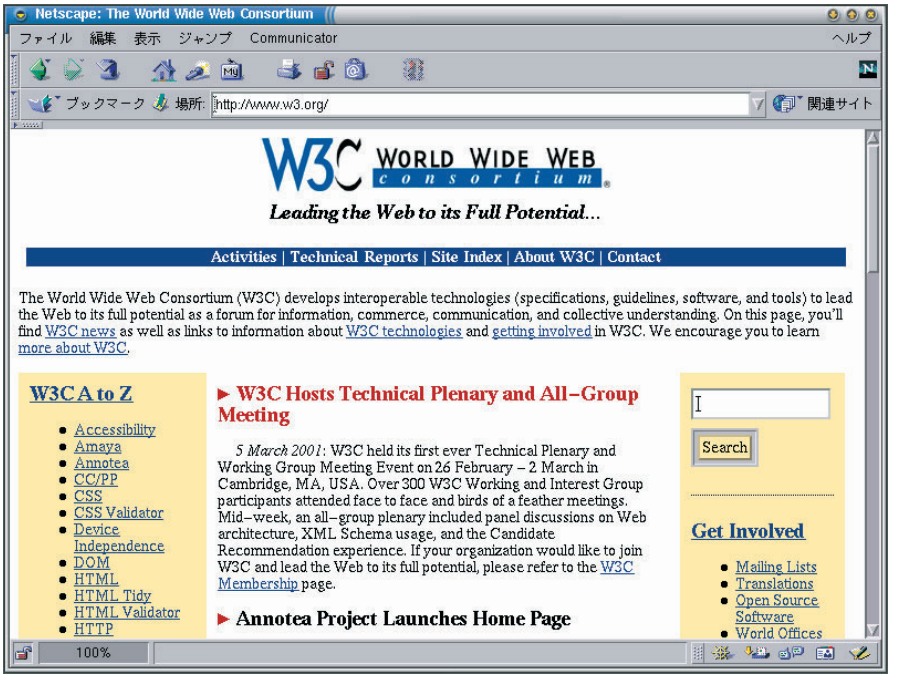
外部リンクの場合は、Xのリソースデータベースで指定したコマンドにURLが渡される。**リスト1**の設定では、Netscape（4.x系）を指定しているので、リンク先のWebページがNetscapeで表示される（**画面4**）。

このほか、文字列の検索機能も用意されている。ウィンドウ下部の双眼鏡ボタン（あるいはFキー）を押して、ダイアログに検索文字列を入力すればいい（**画面5上**）。なお現在のバージョンでは、日本語の文字列を検索することはできない。

Xpdf本体は印刷機能を持たないが、PS形式でのファイル出力機能があるため、対象となるページ範囲を指定して（**画面5下**）PSファイルを作成し、その後gvなどを利用して印刷すればいいだろう。



画面5　文字列の検索を行ったり、PSファイルを出力することも可能だ。



画面4　外部リンクをクリックすると、NetscapeにWebページが表示される。

▼リスト1　.Xdefaults（または.Xresources）に追加する内容。

```
xpdf.t1TimesRoman: /usr/share/fonts/default/Type1/n021003l.pfb
xpdf.t1TimesItalic: /usr/share/fonts/default/Type1/n021023l.pfb
xpdf.t1TimesBold: /usr/share/fonts/default/Type1/n021004l.pfb
xpdf.t1TimesBoldItalic: /usr/share/fonts/default/Type1/n021024l.pfb
xpdf.t1Helvetica: /usr/share/fonts/default/Type1/n019003l.pfb
xpdf.t1HelveticaOblique: /usr/share/fonts/default/Type1/n019023l.pfb
xpdf.t1HelveticaBold: /usr/share/fonts/default/Type1/n019004l.pfb
xpdf.t1HelveticaBoldOblique: /usr/share/fonts/default/Type1/n019024l.pfb
xpdf.t1Courier: /usr/share/fonts/default/Type1/n022003l.pfb
xpdf.t1CourierOblique: /usr/share/fonts/default/Type1/n022023l.pfb
xpdf.t1CourierBold: /usr/share/fonts/default/Type1/n022004l.pfb
xpdf.t1CourierBoldOblique: /usr/share/fonts/default/Type1/n022024l.pfb
xpdf.t1Symbol: /usr/share/fonts/default/Type1/s050000l.pfb
xpdf.t1ZapfDingbats: /usr/share/fonts/default/Type1/d050000l.pfb
xpdf.japaneseFont: -watanabe-mincho-medium-r-*-*-%s-*-*-*-*-*-jisx0208.1983-0
xpdf.urlCommand: netscape -remote 'openURL(%s)'
```

Linuxの実行ファイルそのままWindows上で動かす

LINE

バージョン：0.3a　　ライセンス：GPL

http://line.sourceforge.net/
http://sources.redhat.com/cygwin/（Cygwin DLL）

LINEは、Linux用ソフトをWindows 98／2000で実行可能にするソフトだ。有名なWINEとは逆に、Linuxの実行ファイル中のシステムコールを、Win32 APIやCygwin APIにすりかえることでWindowsで動作させる。ライブラリがスタティックリンクされたソフトはもちろん、共有ライブラリを利用するソフトも実行可能だ。現時点では、コンソール系ソフトや一部のXアプリのみ動作する。実行にはCygwinが必要だ。

●LINEのインストール

LINEは、ソース一式とビルド済みの実行ファイルの両方を含んだZIPファイルのみ配布されている。Windows上でLhasaなどの展開ソフトを使って、適当なインストール先フォルダ（たとえば「C:\LINE」）に直接展開しよう。

このZIPファイルに含まれるLine.exeには、一部の環境で起動時にエラーが表示される不具合があるため、別配布のline-0.3a.exeで置き換える。line-0.3a.exeをLINEのインストール先にコピーした後、コマンドラインから、

```
C>ren Line.exe Line.old
C>ren line-0.3a.exe Line.exe
```

とすればいい。ファイル名を変えずにline-0.3a.exeのまま使うこともできるが、起動の際に「.exe」まですべて入力する必要がある。

LINEの実行には、GNUツールをWindows用に移植した「Cygwin」が必要だ。Cygwinについては、CygwinのWebサイト（上記URLを参照）や、「Project HeavyMoon」（http://www10.u-page.so-net.ne.jp/fa2/riue-s/）の日本語解説などを参考にしてほしい。

Cygwinのパッケージにはさまざまなツールが含まれるが、LINEの実行に必要なのは、「Cygwin DLL」と呼ばれる共有ライブラリだけだ。このDLLは、LinuxなどのUNIX系OSに必須のシステムコールのうち、Win32 APIに欠けているものを補う役目をしている。

WindowsでCygwin環境を使っていない人は、LINEのWebサイトで配布されているCygwin DLLのみのZIPファイルを利用するのが簡単だ。ZIPファイルに含まれるcygwin1.dllを、LINEと同じフォルダに展開すればいい。

●テスト用ソフトで動作を確認

LINEの実行にはLine.exeを利用する（システムコールのすり替えを行うLinexec.exeは直接実行できない）。使い方は簡単で、コマンドライン引数にLinux用ソフトのファイル名を指定すればいい。ソフトの引数を後ろに並べることも可能だ。

インストール先のtestフォルダには、「hello」や「testargs」など、LINEの動作を確認するためのLinux用実行ファイルが用意されている。手始めに、LINEを使ってこれらをWindowsで実行してみよう。

DOSウィンドウ（MS-DOSプロンプト）を開いて、LINEのインストール先フォルダに移動し、

```
C>line test/hello
```

とすると、Line.exeが起動してLinexec.exeを呼び出し、システムコールのすり替えを行いながらtestフォルダのhelloを実行する。実行結果は別のDOSウィンドウに表示される。「Hello Windows」という文字列が表示されればテストは成功だ（**画面1**）。他の実行ファイルについても同様にテストを行ってみよう。

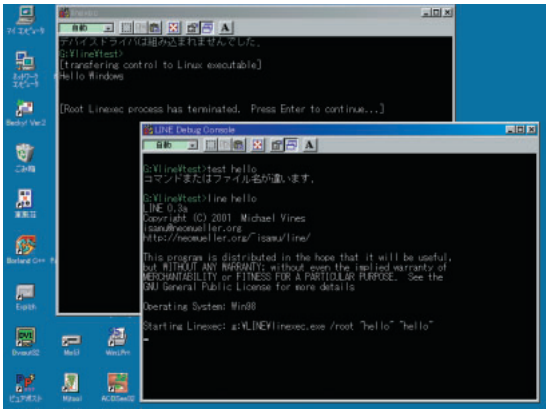
なお、C:\AUTOEXEC.BAT中のPATH設定に、LINEのインストール先フォルダを追加すれば、どのフォルダからでもLINEを実行できる。

●実行可能なバイナリを作成する

利用するライブラリをすべて内蔵した「スタティックリンク」方式の実行ファイルは、実行ファイルとLINEの組み合わせだけで実行できる。LINEに付属するテスト用ソフトもスタティックリンクされたものだ。

こうした実行ファイルをLinuxでビ

画面1　インストール後は、テスト用のソフトで動作を確認する。



ルドするには、Makefileで設定されるgccのリンク用オプションに「-static」を付ければいい。

Makefileをエディタなどを使って直接書き換えてもいいが、configureスクリプトを利用するソフトでは、「./configure --enable-static-link」とすると、スタティックリンク用の設定になるものが多い。

たとえば、bash 2.04の場合は、「./configure --enable-minimal-config --enable-static-link --enable-help-builtin」とすることで、最小構成・スタティックリンク・ヘルプ内蔵の設定が行われる。

この設定でビルドしたbashをWindowsにコピーし、「line bash」としてLINEを実行したところ、DOSウィンドウにbashのプロンプト「#」が表示され、ワイルドカードによるファイル名の展開や、組み込みコマンドforによる繰り返し処理などが動作することを確認できた（**画面2**）。このほか、コンソールで動作するファイルユーティリティ類も動作するようだ。

●X用アプリの一部も動作する

LINEでX用アプリも実行することもできる。ただし、実行結果をWindowsのデスクトップに表示するには、Windows用のXサーバソフト（ASTEC-Xなど）が別途必要だ。

LINEを起動するDOSウィンドウでは、あらかじめ環境変数DISPLAYを設定しておこう。Windowsマシンのホスト名とディスプレイ番号（普通は0）を指定して、

```
C>set DISPLAY=ホスト名:0
```

などとすればいい。

X用アプリの場合は、LINEでサポートしていないシステムコールが含まれる確率が高く、動作しないことが多い。それでも、ファイルシステムを宇宙に見たてて表示する「xcruise」や、rxvtに付属する時計「rclock」など、いくつかのX用アプリが動作することを確認できた（**画面3**）。

●共有ライブラリが必要な場合は

現在のLinuxディストリビューションに含まれる実行ファイルは、さまざまな共有ライブラリ（lib.so.6など）を実行時に参照する「ダイナミックリンク」方式でビルドされている。

このため、実行ファイルだけをWindowsにコピーしてもLINEでは動作しない。あらかじめ、実行時に参照される共有ライブラリをLinux上でlddコマンドを使って調べ、それらをCygwin環境のlibディレクトリにコピーしておく必要がある。

つまり、Cygwin DLLだけではなく、その他のツール類も含めてちゃんとしたCygwin環境を構築しなくてはならない。説明が複雑になるため、今回は省略させていただく。

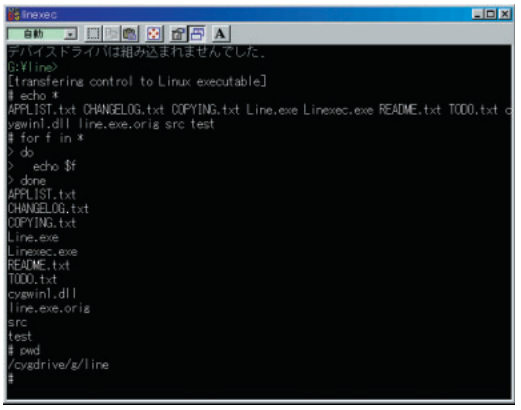
●まだ未完成な部分も多い

Windows用ソフトをLinuxで動作させるというLINEとは正反対の機能を持ったWINEというソフトがある。WINEの開発に際しては、ソースが未公開なうえに、膨大な数にのぼるWindows APIを、Linuxのシステムコールで実現しなければならない。

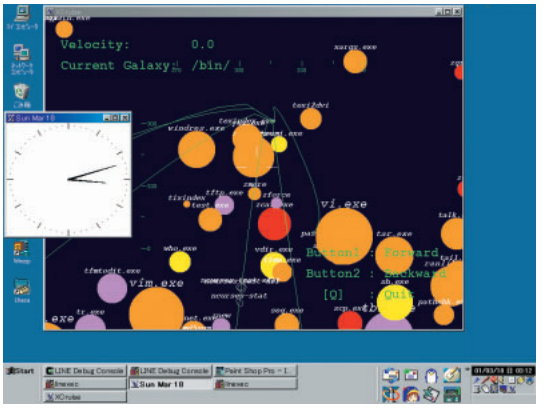
一方LINEの場合は、POSIX準拠でオープンソースのLinuxカーネルが対象なので作業は楽だ。実際、LINEのソースには、カーネル2.2のソースが一部流用されている。

とはいえ、バージョン番号が0.3であることからもわかるように、現時点でのLINEの完成度は決して高くはない。LINEでサポートしていないシステムコールを利用するソフトを動かすと、エラーメッセージを表示するメッセージボックスが開く。

さらに、「Bad syscalls」というメッセージの後で、Linexec.exeが「不正な処理（ページ違反）」で異常終了することもある。このケースはWindows 98だと回復できないため、リセットするしかない。一方、より堅固な作りのWindows 2000では、Linexec.exeのみ強制終了できる。



画面2　LINEを使ってLinux用のbash 2.04をDOSウィンドウで実行中。



画面3　Xサーバと組み合わせればデスクトップにXアプリを表示できる。

洗練されたファイル転送ソフト

lftp

バージョン：2.3.8　　**ライセンス：GPL**

http://lftp.yar.ru/

lftpは、FTP／HTTPなど6種のプロトコルに対応したコマンドラインベースのファイル転送ソフトだ。bash風のジョブ制御機能を持ち、複数のコマンドを同時に実行できるのが特長だ。中断した箇所からのレジューム転送や、巨大なファイルを複数に分割しての受信、サーバのディレクトリ構造を再現するミラーリングなどにも対応している。また、日本語カタログが付属しているため、日本語環境ではメッセージやヘルプが日本語で表示される。

●ビルドとインストール

lftp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ボールは、「./configure」→「make」でビルドし、「su」でrootになって「make install」でインストールという一般的な手順だ。

一方、配布されているRPMバイナリパッケージはRed Hat Linux 7用なので、Red Hat 6ベースのディストリビューションでは、rpmを3.0.5以降に更新しないと利用できない。以下の手順に従って、tarボールからRPMパッケージを各自作成するとよいだろう。

「su -」としてrootになってから「rpm -ta lftp-2.3.8.tar.gz」とすると、/usr/src/redhat/RPMS/i386にバイナリパッケージが作成される。これを「rpm -Uvh lftp-2.3.8-1.i386.rpm」としてインストールしよう。

●lftpの起動とサイトへの接続

ktermなどのコマンドラインで「lftp」とするとlftpが起動して、コマンド入力待ちのプロンプト「lftp :~>」が表示される。

lftpには、通常のftpよりもコマンドが豊富に用意されている（**表1**）。各コマンドの詳細は、「help コマンド名」で表示される日本語ヘルプを参照してほしい（**画面1**）。

まずは、openの引数にURLを指定して接続する。たとえば、

```
lftp :~> open ftp://ring.asahi-net.or.jp/pub/linux/Vine
```

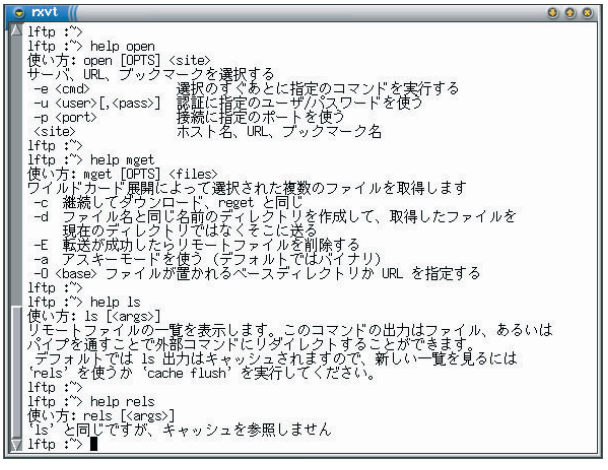
とすると、FTPを利用してring.asahi-net.or.jpに匿名（anonymous）ログインし、/pub/linux/Vineディレクトリに移動する。現在接続しているサーバ名やディレクトリ名は、プロンプトの表示で確認可能だ。

上のような長いURLを毎回入力するのは面倒なので、ブックマーク機能を活用しよう。「bookmark add ブックマーク名」として現在のURLをブックマークに登録すれば、「open ブックマーク名」で接続できる。「朝日Vine」のように、日本語のブックマーク名を付けることも可能だ。

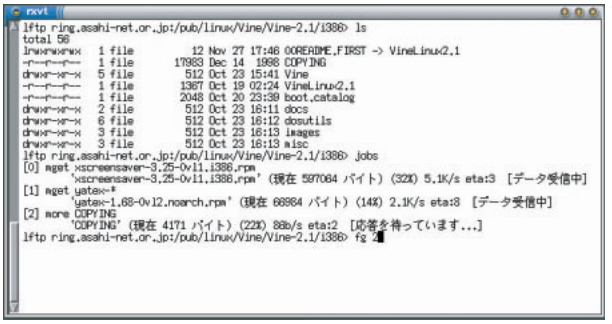
●bashと同様のコマンドライン操作

ログインした後は、通常のftpと同様に、lsなどを使ってリモート（接続先）の情報を取得する。一方、ローカル（接続元）の情報を確認するには、「!ls」のように、!の後ろに外部コマンドを指定すればいい。

lftpのコマンドラインは、bashと同レベルの編集／履歴機能を備えている。中でも、リモートのファイル名の一部を入力するだけで残りを補完してくれるTabキーは便利だ。



画面1　メッセージやコマンドのヘルプが日本語で表示される。



画面2　同時に複数のコマンドをバックグラウンドで実行できる。

なお、リモートのファイル情報は自動的にキャッシュされるため、lsを繰り返し実行しても2回目以降はデータが転送されない。ファイルを送信した後の状態を確認するには、キャッシュを参照しないrelsを使うか、「cache flush」としてキャッシュの内容を破棄する必要がある。

●ファイル転送とジョブ制御

ファイルの送受信に関するコマンドには、通常のftpに用意されているget／putやmget／mput（ワイルドカード対応）のほか、転送を中断した箇所から再開するreget／reput、ファイルを複数の接続で分割受信するpgetが用意されている。詳しい使い方はヘルプを参照されたい。

また、mirrorを使うと、リモートのすべてのファイルをディレクトリ構造ごとローカルディスクに構築するミラーリングを行える。逆にローカルディスクから、リモートへのミラーリングも可能だ。

毎日1回ずつミラーリングする、あるいは指定時刻にミラーリングを開始するなどの処理も、一定間隔で繰り返しコマンドを実行するrepeatや、指定時刻にコマンドを実行するatとの組み合わせで実現できる。

●バックグラウンドでファイル受信

lftpでは、bash風のジョブ制御により複数のコマンドをバックグラウンドで実行できる。ファイルを受信しながら、別のファイルの内容を画面に表示する、といった操作を簡単に行えるわけだ。

コマンドをバックグラウンドジョブとして実行するには、シェルと同様にコマンドラインの末尾に「&」をつければいい。複数のコマンドをカッコで囲んでグループ化することも可能だ。

現在のジョブ一覧はjobsで参照できる（**画面2**）。実行中のコマンドをバックグラウンドに移行したり（Ctrl－Zキー）、その逆の操作をfgで行うことも可能だ。

なお、ファイル転送中にexitで終了すると、lftp自体が自動的にバックグラウンドに移行して、転送完了まで処理を続行する。

(1) 接続関係のコマンド	
open	URLなどを指定して接続
close	アイドル状態の接続を閉じる
exit	終了、またはバックグラウンドに移行
(2) ファイル一覧、内容表示など	
ls、rels	ファイル情報の一覧表示（relsはキャッシュを無視）
nlist、renlist	ファイル名一覧表示（renlistはキャッシュを無視）
find	ディレクトリの内容を再帰的に表示
cat、zcat	ファイルの内容を標準出力に表示（zcatは圧縮対応）
more、zmore	ファイルの内容をページャで表示（zmoreは圧縮対応）
pwd、lpwd	カレントディレクトリを表示（lpwdはローカルが対象）
cd、lcd	カレントディレクトリを変更（lcdはローカルが対象）
mv	ファイル名を変更
chmod	ファイルのパーミッションを変更
rm、mrm	ファイルを削除（mrmはワイルドカード対応）
mkdir	ディレクトリを作成
rmdir	ディレクトリを削除
(3) ファイル転送コマンド（get系）	
get	リモートのファイルを取得
mget	ワイルドカードに対応
pget	複数接続に対応
reget	レジュームに対応
(4) ファイル転送コマンド（put系）	
put	ローカルのファイルを送信
mput	ワイルドカードに対応
reput	レジュームに対応
mirror	ディレクトリ構造ごとミラー
(5) ジョブ制御とコマンド実行	
jobs	実行中のジョブ一覧を表示
kill	指定したジョブを削除
fg、wait	指定したジョブが終了するまで待機
scache	セッションの一覧表示、または切り替え
!	外部コマンドを実行
at	指定時刻にコマンドを実行
command	別名を無視してコマンドを実行
glob	ワイルドカードを展開して指定のコマンドを実行
queue	指定したコマンドをサイトごとのキューに追加
repeat	コマンドを繰り返し実行
source	ファイルに書かれたコマンドを実行
site	サイトのコマンドを実行し、結果を表示
(6) その他のコマンド	
alias	別名（エイリアス）の指定、解除、一覧表示
set	変数の設定、削除、一覧
user	リモートログイン時に使うユーザー名などを設定
anon	ユーザーを匿名（anonymous）に設定
bookmark	ブックマークの制御（サブコマンドあり）
cache	ローカルキャッシュを制御（サブコマンドあり）
help	コマンド一覧、または各コマンドのヘルプ表示

表1　lftpのコマンド（一部抜粋）

ヘリコプターを操る戦略アクションゲーム

Copter Commander

バージョン：1.0　　ライセンス：Artistic

http://www.speakeasy.org/~morse/copter-commander/
http://www.student.oulu.fi/~jlof/gtkglarea/（GtkGLArea）

Copter Commanderは、ヘリコプターを操って敵のヘリや車両などと戦う横スクロールタイプのアクションゲームだ。単純なドンパチではなく、兵士や車両を支援して敵を攻略する戦略性の高いゲームだ。複数のプレイレベル（面）が用意されており、コンピュータを相手に1人で遊ぶソロプレイのほか、最大4人で行うネットワークプレイも可能だ。実行にはMesaとGtkGLArea、GNOME／GTK＋が必要となる。

●ビルドとインストール

まずは、ライブラリをインストールする。OpenGL互換の3Dライブラリ Mesaについては、インストール済みのディストリビューションが多いはずだ。Copter Commanderのビルドには Mesa-develパッケージも必要なので、忘れずにインストールしておこう。

次に、OpenGL用GTK＋ツールキットのGtkGLAreaをインストールする。配布されているtarボールからRPMパッケージを作成しよう。「su -」としてrootになってから、「rpm -ta gtkglarea-1.2.2.tar.gz」とすると、/usr/src/redhat/RPMS/i386にgtkglareaとgtkglarea-develパッケージが作成される。これらを、「rpm -Uvh gtkglarea-＊1.2.2-1.i386.rpm」としてインストールすればいい。

Copter Commanderは、tarボールのみ配布されている。configureスクリプトは付属しないので、そのまま「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。

●起動からプレイ開始まで

ktermなどのコマンドラインで「copter-commander&」とすると、Copter Commanderのウィンドウが開いてタイトルが表示される（**画面1**）。

ソロプレイの場合や、ネットワークプレイの最初のプレイヤーである場合は、[ゲーム]－[新ゲーム]でダイアログを開いて、ポート番号とプレイレベル（面）を指定する（**画面2**）。通常、ポート番号は変更する必要はない。また、プレイレベルの「Balloon Run」（初期値）と「Micro」はネットワークプレイ専用なので、ソロプレイでは「Flashpoint」などを選択しよう。

一方、ネットワークプレイの残りのプレイヤーは、[ゲーム]－[Join game]を選択し、最初のプレイヤーのホスト名とポート番号をダイアログで指定するだけでいい（**画面3**）。

どちらの場合も、ゲームに参加するプレイヤー（最大4人）の一覧がダイアログに表示される（**画面4**）。参加チーム（LeftまたはRight）の選択やチャットも可能だ。だれかが[OK]ボタンを押すとプレイ開始となる。

●操作はマウスとキーを併用

しばらく待つと、プレイヤーの操縦する戦闘ヘリ（ガンシップ）がヘリポートから離陸する（**画面5**）。ヘリの操作にはマウスを使用し、左ボタンが向きの反転、中ボタンが機銃の発射、右ボタンが爆弾の投下だ。

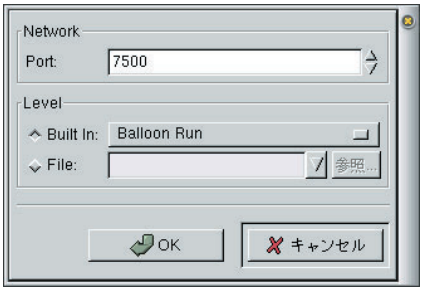
戦闘ヘリには、このほかにも空対空ミサイルや空対地ロケットなどの武器が用意されている。こうした武器の操作はキーボードで行う（**表1**）。機銃と爆弾も含め、左Shiftキー（機銃）、zキー（爆弾）、xキー（ミサイル）、cキー（ロケット）と順番に並んでいるので、左手で操作するといい。

戦闘ヘリの機数は限られており、敵

画面1
一瞬のイントロのあとで表示されるタイトル画面。

Esc	マウスの外部移動を許可
Tab	マウスの外部移動を禁止
Shift	機銃を発射
z	爆弾を投下
x	追尾型ミサイルを発射
c	ロケット弾を発射
Space	（搭載中の）歩兵の降下
t	戦車を要請
m	歩兵を要請
e	技師を要請
a	対空車両を要請
v	爆破車両を要請
h	戦闘ヘリを要請
Enter	トークモード
Ctrl－S	サウンド切り替え
Ctrl－B	αブレンディング切り替え
Ctrl－N	ヘリ画像の再ロード（nVidiaチップ用）

表1　プレイ中のキー操作一覧



画面2　最初のプレイヤーはポート番号とプレイレベル（面）を選択する。

画面3　残りのプレイヤーはホスト名とポート番号を指定するだけでいい。

画面4　プレイ前のダイアログ。プレイヤー同士のチャットも可能だ。

に破壊されたり、燃料切れで墜落したりして残機が0になると、丸みを帯びた形の民間ヘリ（シビー）に切り替わる。民間ヘリは無制限に使える代わりに武器が貧弱で、機銃と爆弾しか持っていないし、弾数も少なめだ。

なお、弾薬や燃料はヘリポートに着陸すると補給される。上部のインジケータを見て、弾薬や燃料が残り少なくなったらヘリポートに戻ろう。

●味方ユニットの出動を要請

このゲームの目的は、敵陣営の移動司令部（HQC）を破壊することにある。なぜか亀の形をしている移動司令部は、それぞれの陣営（プレイエリアの左右端）から、ゆっくりと相手陣営に向かって移動する。

注意してほしいのは、プレイヤーの操縦するヘリでは、敵の移動司令部を絶対に破壊できないという点だ。移動司令部を破壊できるのは、破壊車両（デモリッションバン）と呼ばれるユニットに限られている。

破壊車両はvキーで出動を要請でき、味方陣営から敵の移動司令部に向かって自動的に移動する。ただし、破壊車両はとても脆弱なので、敵の掩蔽壕（バンカー）や砲台などの建造物や、歩兵や戦車といったユニットと戦闘すると、簡単に破壊されてしまう。

破壊車両を確実に敵の移動司令部まで到達させるには、こちらも歩兵や戦車などを出動させて、敵の建造物やユニットを破壊する必要がある。出動要請のキー操作については、**表1**を参照されたい。

出動にはユニットごとに一定のコストがかかり、手持ちの資金がそれに応じて減少する。資金を増やすには敵の建物やユニットを破壊すればいい。

やみくもに出動要請を出すのではなく、「戦車は他の地上ユニットに強い」「歩兵は一部の建造物を接収できる」といった、ユニットごとの特性を生かせるように、出動させるユニットを決める必要がある。もちろん、プレイヤーの操作するヘリでも敵のユニットや建造物を破壊し、味方の破壊車両を援護しよう（**画面6**）。

こうして、爆破車両を先に敵の移動司令部に到達させた陣営が勝利の栄冠を得る。

画面5　戦闘ヘリがヘリポートから離陸する。左下は味方の移動司令部。

画面6　戦車や歩兵などの地上戦力と協力して敵を攻撃しよう。

直感的な操作で使えるテキストビューア

lookat／bekijk

バージョン：1.2　　ライセンス：フリー

http://staf.digibel.org/topic.php?lang=eng&top=lookat

lookat／bekijkは、端末画面で動作するオランダ生まれのテキストビューアだ。英語版のlookatとオランダ語版のbekijkが用意されている。カーソルキーなどによる直感的な操作や、画面上部に用意されたメニューなど、使い心地はDOSのビューアに近い。ソースを修正することなく日本語表示が可能だが、検索時の日本語入力には対応していない。実行にはncursesライブラリが必要となる。

●ビルドから起動まで

lookat／bekijk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スクリプトは含まれていないが、たいていのLinuxのディストリビューションではMakefileを修正する必要はない。そのまま「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しよう。

以下では、lookatについて説明するが、bekijkもメニューなどの表示が異なるだけで機能は同じだ。なお、X上でlookat／bekijkを利用する際、表示色の関係でktermでは文字が見にくくなってしまうので、代わりにrxvtを利用したほうがいいだろう。

●テキストファイルを快適に閲覧

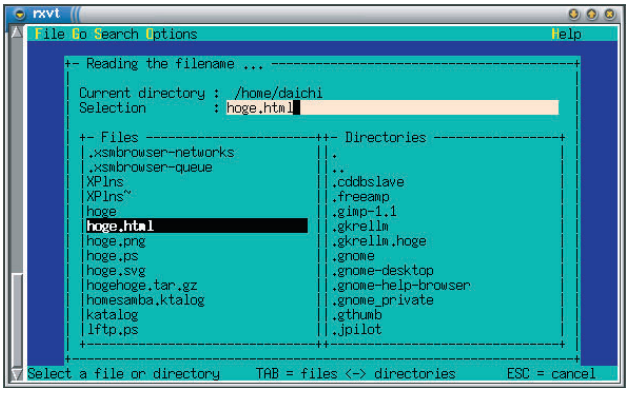
Linuxのコンソールか、X上の端末ソフト（rxvtなど）のコマンドラインで「lookat」とすると、lookatが起動してファイル選択ダイアログが開く（**画面1**）。「lookat ファイル名」のように、コマンドライン引数でファイル名を指定することも可能だ。

いずれにせよ、ファイルの内容が端末画面のほぼ全体に表示される。ただし、画面の最上行はメニューの表示、最下行はファイル名や行番号などの表示に使われる（**画面2**）。

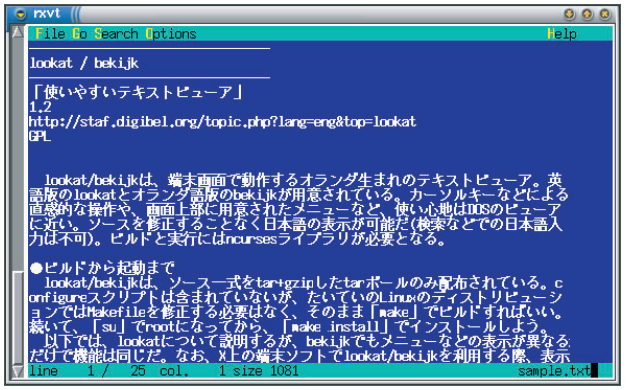
スクロールなどの操作はキーボードで行う。カーソルキーやPageUp／PageDownキー、Home／Endキーなど、直感的に使い方がわかるキーのほか、一部の操作はlessやEmacs風のキー割り当てになっている。

最上行のメニューは、Alt－英字キーで開ける。英字キーはメニュー先頭の文字で、FileメニューならAlt－Fキー、HelpメニューはAlt－Hキーだ。たとえば、lookatのキー操作を一覧表示するには、Alt－HキーでHelpメニューを開き、[Keys]を選択すればいい。メニュー項目の選択には、カーソルキーとEnterキーを使う。

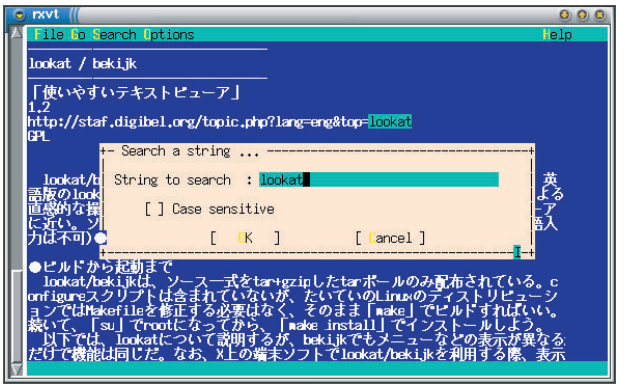
／キーを押すか、[Search]－[a string]を選択すると、検索用ダイアログが開いて、指定した文字列を検索できる（**画面3**）。¥キーによる再検索も可能だ。なお、検索文字列に日本語は入力できないので気をつけよう。



画面1　起動時にファイル名を省略した場合は、この画面で選択する。



画面2　青い背景に白い文字のビューア画面。日本語も表示可能だ。



画面3　文字列検索などは上部のメニューから選択する。

Webページのダウンロードとミラーリング

HTTrack

バージョン：3.00RC1　　ライセンス：GPL

http://www.httrack.com/

HTTrackは、Webサイトを構成するHTMLファイルや画像ファイルなどをダウンロードし、オフラインでブラウズできるようにするミラーリングソフトだ。Windows用のビジュアル版とUNIX系OS用のコマンドライン版が用意されている。単なるダウンロードソフトではなく、相対リンクを書き換えてくれるほか、追加、変更されたファイルだけを取得する更新ミラーや、レジューム転送にも対応している。

●URLを指定するだけの簡単操作

HTTrackは、ソース一式をtar＋gzipしたtarボールのみ配布されている。tarボール展開先のsrcディレクトリで、「./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。

基本的な使い方は簡単で、コマンドライン引数にWebサイトのURLを指定すればいい。リンクをたどりながら、Webサイトを構成するHTMLや画像、その他のファイルがすべてダウンロードされる（**画面1**）。複数のURLを並べて指定することも可能だ。

ファイルの保存先はカレントディレクトリが初期値だが、同じディレクトリにURLなどのキャッシュ情報などのデータファイルも作成されるので、－Oオプションで保存先ディレクトリを明示したほうがいいだろう。

たとえば、「httrack "http://www.hoge.com/" -O ~/webdata」とすると、ホームディレクトリのwebdata以下に、ミラーリングされたwww.hoge.comのファイルやキャッシュ情報などがすべて格納される。

2回目以降のミラーリングは、保存先ディレクトリに移動後、「httrack --update」とするだけでいい。キャッシュ情報に基づいて更新ミラーが行われ、追加、更新されたファイルだけダウンロードされる。

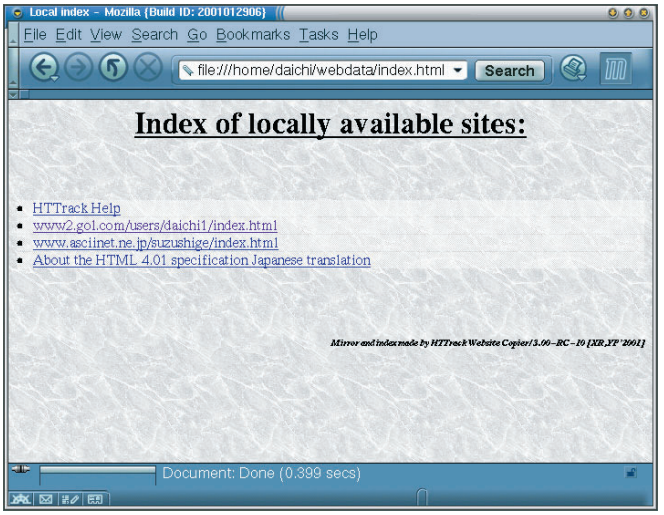
●ミラーリングしたページを表示

保存先ディレクトリには、ダウンロードしたファイルやキャッシュ情報のほか、保存したWebサイトの一覧を表示するindex.htmlが用意されている（**画面2**）。このページを基点として、ダウンロードしたWebサイトをオフラインでブラウズすればいい。

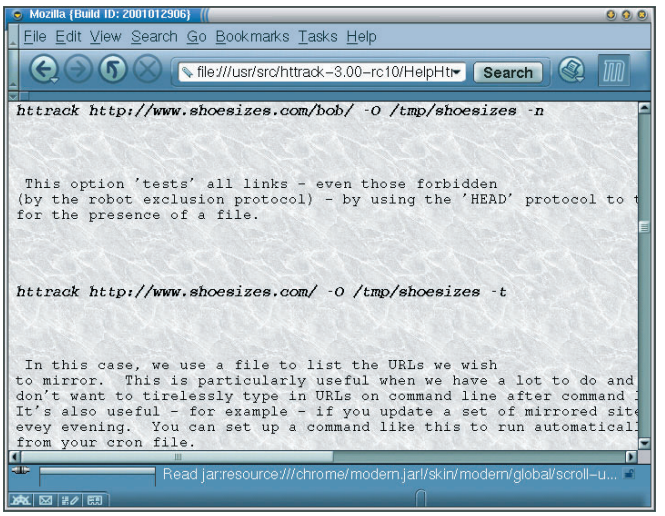
なお、今回紹介した方法では、指定したURL以下のリンク先のファイルしかダウンロードされない。外部リンクをたどる場合は、ダウンロードを許可／禁止する「＋パターン」や「－パターン」、リンクの深さを指定する－rオプションなどを組み合わせて指定する必要がある。

こうしたオプションについては、tarボールに含まれているHTML形式のユーザーズガイド（英語版）に詳しく説明されているので、こちらを参照されたい（**画面3**）。

画面1
指定したURL以下のファイルをまとめて取得できる。



画面2　ダウンロードしたWebサイトの一覧がリンクとして表示される。



画面3　豊富な例題を丁寧に解説したユーザーズガイドは一見の価値あり。

隠喩としてのコンピュータ

デフレ時代のITを考える

文：豊福 剛
Text : Tsuyoshi Toyofuku

いま、世の中のかなりの人が鬱であるに違いない。過去における失敗、あのとき、やらねばならなかったことが、きちんとできていれば、このような現在にはなっていなかったのに、という後悔の念が、人を鬱に陥れる。

精神病の本などを読むと、鬱というのは、過去に対する負債の感情と関係しているらしい。すると鬱は、個人的な病というだけでなく、集団の病としてもありえるのだろう。日本の'90年代は「失われた10年」として語られているのだけれども、「失われた〜」という形容そのものが、鬱の表明にほかならない。しかも、現在が「失われた10年」を脱したわけではなく、依然として出口なしの状態である以上、10年どころか、20年も30年も失われ続けるのかもしれない。

デフレスパイラルの鬱

もちろん、鬱に陥る前は、鬱でない状態、あるいは鬱をさほど意識しないでやりすごせた状態が存在する。過去の負債によって、未来の展望が否定された状態が鬱だとすると、バブルは躁である。未来の展望が過剰に肯定され、現在の負債が未来に先送りされるのだ。

'80年代は、バブルというだけでなく、ポスト産業化社会の到来が宣言された時代でもあった。それはおそらく、経済における供給と需要の関係が逆転した状況に対応しているのだろう。

物が乏しい状況においては、供給が主導で、供給に需要が従属するのに対して、物が豊かな状況においては、需要が主導で、需要に供給が従属する。生存に不可欠な物は、もはや十二分に供給されていて、供給が需要をはるかに上回っている。供給が過剰になると、供給は需要を創出しなければならない。必要なものはすでに十分に足りているから、必要ではなく、欲求を掘り起こさなければならない。それは、「欲しいものが欲しい」ということでもある。そのような状況で欲求として機能したのが、たとえばブランドという価値なのだろう。

ブランド品というのは、それを購入することで、その商品の機能ではなく、その商品の象徴的意味が価値として消費される、そのような商品である。その商品の良し悪しではなく、その商品の好き嫌いが、選択の基準になる。ブラ

ンド品は、非ブランド品に比べて、一般に価格がきわめて高いのだが、それは単に高いだけではなく、そのブランドの象徴的意味を買うのだ。

バブル期の'80年代日本は、まさにそのような時代だったわけだが、需要が主導でありえたのは、貨幣が過剰に流通していたからである。現在は、デフレの時代であり、貨幣の流通が滞り、商品の価格は低下していく。貨幣の流通が滞るために、商品よりも貨幣が求められる。供給が需要に従属している関係は変わらないので、需要が縮小した以上、供給は価格を下げることでしかそれに対応できなくなる。

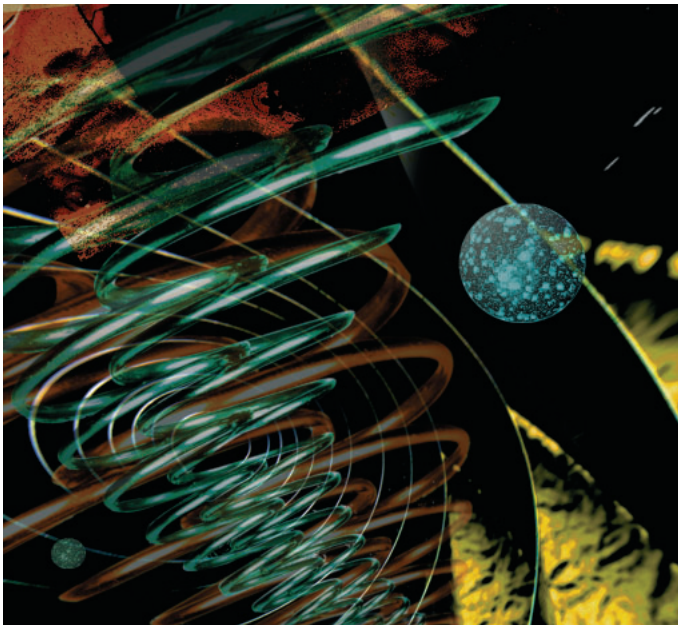
これは悪循環である。需要が供給に従属している場合には、価格の低下はより多くの需要を産み、そこで流通した貨幣は、賃金として還元されるが、供給が需要に従属している場合には、価格が低下しても、需要は拡大しない。それでも価格を下げることでしか供給を維持できないとなると、価格を下げる圧力は、賃金を下げ、雇用を減らすことに、いずれは帰結する。そうなると、所得が減少するから、需要は縮小し、ますます物が売れない状況になる。

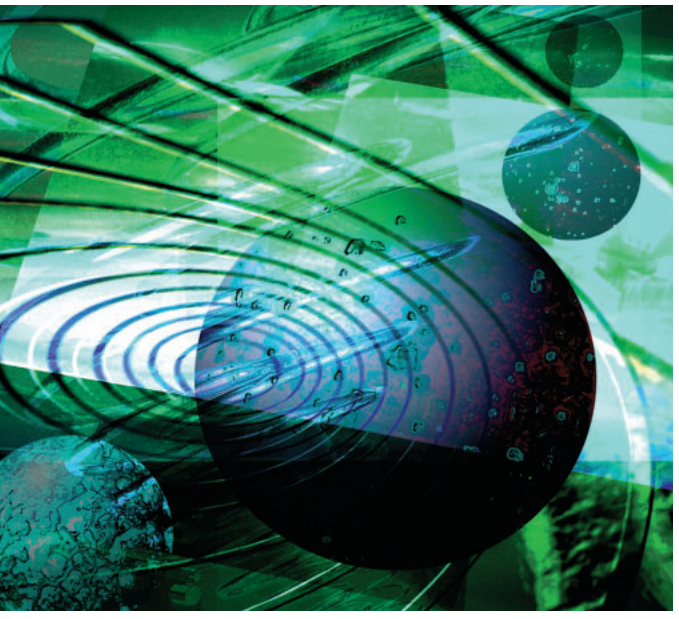
ニューエコノミーの失速

最近、気になるのは、もしかして、このデフレスパイラルに、コンピュータやインターネットがかなり関係しているのではないかということだ。コンピュータの性能向上と価格低下はこれからも続いていくだろうし、同様に、インターネットの高速化と価格低下も続いていくだろう。それは誰でも良いことだと考えるし、そのことを否定するつもりはない。ただし、それはあくまで、コンピュータやインターネットが、供給が主導で、需要が供給に従属している前提での話だったのではないだろうか。

1995年から2000年くらいまでは、Windows 95の発売もあって、パソコンが一般家庭に普及していった時期であり、同時にインターネットの普及が拡大していった時期でもある。しかし、その後、iMacやらVAIOやらが流行し始めたことは、ブランド品としてのパソコンが欲求される段階に入ったことの徴候であり、供給が需要に従属する関係に逆転したことを意味するとしたら…。

もちろん、この考えを積極的に認めたくはないし、パソ





コンとインターネットの普及は、新しい需要を創出する大きな要因として機能するのだと考えたい。たぶん、そのような願望のひとつの頂点が、ニューエコノミー論だったのだろう。アメリカはITによって、新しい産業を創出し、恒久的な成長を続けていくと主張する、あのニューエコノミー論である。しかし、デフレスパイラルの現実は、そのような願望をすら懐疑させてしまうように思う。

ドット・コム企業に対する期待というのは、新しい産業の創出に対する期待だったのだろう。日本でいわれるIT革命というのは、おそらく、新しい産業の創出に対する期待の部分が半分と、既存産業のITによる効率化、それによる国際競争力の獲得を目指す部分が半分なのだろう。しかし、昨年来のアメリカでのIT関連株の下落傾向をみると、新しい産業の創出というのは、やはり、かなりの部分、実体以上に水脹れしたバブルだったように思えてしまう。

コンピュータやインターネットを戦略的に活用できた企業が、市場での競争に勝ち残れるというのは、確かだとしても、それは、企業の活動における諸々のコストが削減されたり、それまでの諸々のオペレーションがより効率的に再編成された場合にのみ成り立つのだろう。しかし、それは、いままでやってきたことを、より安く、より効率的に実行することではあっても、それまでになかったような、何か新しい産業や経済の形態を誕生させるものであるのか、そこのところが、よくわからないのだ。

市場主義経済にとってITとは何なのだろう？

市場主義経済の根本には、商品の売買がある。そこでは、物やサービスを商品として売ることによって、貨幣を手にし、その貨幣でもって、物やサービスを買う。それでは、情報と商品の関係はどう考えればいいのだろうか。

市場主義経済にとっての情報には、おそらく3つのカテゴリーがあると思われる。ひとつは、商品の生産を効率化するための情報。もうひとつは、商品の流通を効率化するための情報。そして最後に、商品そのものとして成立する情報。ITはこの3つのカテゴリーのいずれにも大きく関連する。

物やサービスの生産と流通をITによって効率化することは、資本主義において生産様式の高度化が要請される以

上、不可欠の条件である。しかし、それは同時に、ITによって労働の質と量が変ることも意味する。それまで人間がやっていた情報処理の作業を、ITに置き換えることによって、より短い時間でその作業が完了するようになれば、その分、人が不要になってしまうということでもある。

生産や流通に必要な労働力が減れば、その分、商品の価格は低下する。しかし、失業が増大し、賃金の水準が低下するので、需要の絶対量は縮小する。需要の絶対量を拡大するためには、既存の産業ではなく、何か新しい産業が創出されなければならない。

そう考えると、商品そのものとして成立する情報が、その突破口にならないだろうかと考えたくなる。しかし、情報が商品として成立するためには、その情報が希少性を持たなければならない。メディア産業について考えてみよう。本や雑誌、あるいはCDやビデオは、立ち読みしたり、試聴したりできたとしても、それを物として所有するためには、それを貨幣で買わなければならない。情報は、物にパッケージされることで商品として成立しているのだ。しかし、ITは、情報を物から剥離し、情報そのものの流通を可能にする。いまさらNapsterの例を持ち出すまでもなく、ITは情報の希少性の根拠を突き崩してしまう。

インターネットのホームページは、ある意味でテレビに似ている。テレビ番組の多くは貨幣で買うことなく消費されている。テレビ番組は広告によって成り立っている。同様に、新聞や雑誌も、広告がなければ、それを発行し続けることはできない。インターネットの本格的普及から5年以上が経過したわけだが、ホームページに関していえば、いまだに広告による収益しか成り立たないように思える。

広告が、物やサービスをアピールし、その購買を促すことに目的があるとしたら、商品として販売される物やサービスの絶対量が増えない以上、広告のパイそのものがインターネットによって増えることはなく、そのパイをめぐって他の既存メディアとの競争になる。インターネットや携帯電話の普及によって、テレビやラジオを試聴する時間は相対的に減るだろうし、本や雑誌が読まれる時間も減るだろう。商品としての情報を、有料コンテンツや広告といった従来の枠組みで考えるかぎり、ITはそれらを発展させるものというより、むしろ逆の効果をもたらすもののように思えるのだ。

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。

労働運動にもオープンソースを

文：安田幸弘
Text : Yukihiro Yasuda

オープンソースのアプリケーションサーバ

Zopeというオープンソースのアプリケーションサーバ*がある（http://www.zope.org/）。しばらく前までは出来合いのアプリケーションもなく、バグやら仕様変更やらでイマイチ使いにくかったのだが、この半年ぐらいで急速に充実してきた。ちょっと重いのが欠点だけど、チョイチョイとメソッドを作ってやるだけで、わりと簡単にいろんなことができてしまう。

市販の高性能なアプリケーションサーバと比べると、Zopeはまだ荒削りなところもあるけど、なんてったってオープンソース。市販のアプリケーションサーバとは自由度が違う。特別な開発ツールもいらないし、不可解なAPIと格闘する必要もない。もちろん、ZopeにはZopeのうっとうしさはある。まず日本語でちゃんと使えるようにするまでにひと苦労させられるのは、Zopeに限らず洋モノの宿命だとしても、バグなんだか仕様なんだか、しばしば得体の知れない挙動に悩まされたり、独自のオブジェクトデータベースに飲み込まれたコンテンツを外部から利用しにくいとか、まあ、いろいろある。

それでもZopeの面白さは、そのへんのさまざまなオブジェクトを組み合わせたり、足りないものは適当に作ればダイナミックなWebパブリッシングができちゃうというところ。これで個人サイトを作ってもいいんだけれど、こういう面白いツールは、自分一人でいじっているだけではつまらない。Zopeみたいなサーバがあれば、ちょっと大掛かりなサイトもサクサク簡単に作れるのだが、仕掛けを作るのは簡単でもコンテンツを入れるのはまた別の話である。かといって、いきなり商用サイトってのもんだしなあ、なんてことを考えていたときに、日本に労働運動のサイトを作らないか、という話が持ち上がってきた。

サイトの計画段階で、情報サイトとコミュニティサイトの側面を持ち、会員や第三者がWebを通じて情報の提供と共有を目指すということになったのだが、これを手作業でやると専属の編集スタッフが必要だろう。ぼくが「それではZopeでやりましょう」と提案したのは言うまでもない。

「マイク・ライボネッ！」

そのサイトはそのものずばり「レイバーネット」。サンフランシスコに初めて「レイバーネット」を名乗るネットワークが誕生したのは1991年、まだパソコン通信の時代だった。1994年ごろ、サンフランシスコでレイバーネットを運営していたスティーブ・ゼルツァー氏の話を聞いたことがあるのだけど、彼は「世界を結ぶネットワークが労働運動を変える」と力説していたのを思い出す。

その後、1996年にイギリスのリバプールの港湾ストをきっかけに、イギリスに誕生したのがインターネットのWebを使った「レイバーネット」だった。このリバプールの港湾ストは、インターネットを使った国際的な支援のネットワークを背景として勝利を収め、労働運動の活動家たちにインターネットの威力を思い知らせたできごとだったのだが、そのときの支援ネットワークを基礎に一般的な労働運動のネットワークとして「レイバーネット」を立ち上げたのがクリス・ベイリー氏である。昨年、インターネットの権利プロジェクトの一環として彼を日本に呼んだとき、クセの強いロンドン訛で毎日「マイク・ライボネッ！」（レイバーネットを作れ）と言われたのが、まあ、今回の日本のレイバーネットの直接のきっかけだったりするわけだが、それはともかく、インターネット上のレイバーネットは、リバプールの後、各国に続々と誕生、盛んに労働運動の情報の交換が行われている。

ちなみに、レイバーネットという名の看板をかけてはいても、各国で運営されているそれぞれのレイバーネットは、別に組織的なつながりがあるわけでも人の交流があるわけでもない。まあ、言ってみればレイバーネットの名前は赤い旗みたいな一種のシンボルだ。政治的な立場や思想・信条

を超えて、労働運動に関わる人が集まるノードにつけられたラベルなのだ。

クリス・ベイリーは世界中のすべての国に「レイバーネット」を作りたいと言っていたけど、中でも日本は重要だと言う。なにしろ、世界中、どこに行っても日本企業の看板を見ない国はないほど、日本の企業活動は国際化している。それに伴って、日系企業がらみの労働争議も各地で頻発しているのに、同じ日系企業で働く労働者の連帯どころか情報交換もない状態なので、日本に国際的な労働運動情報の交換基地を作ることはとても大事だという。そんなクリスの働き掛けや、新しい労働運動を作っていこうとする人たち、そして日本で労働運動に関わる人々を中心に、「レイバーネット日本」の構想が現実化し、以前から主にNGO関連のつながりで各国のレイバーネットのスタッフとコンタクトを持っていたぼくが技術方面を引き受けることになった、というのがコトの次第である。

パソコンはめんどくさいだけのワープロ？！

もっともぼくは、労働運動といえば菜っ葉服を着たオジサンたちが集まり、赤い鉢巻を締めて「団結ガンバロー！」なんて叫んでる程度のイメージしかない門外漢なので、コンテンツ的な協力はほとんどできない。主に技術部分でのサイト構築やZopeアプリケーションの日本語化や機能改善なんかをやっているのだが、労組のオジサンたちからは「ログインって何の意味だ？　めんどくさいからログインしなくていいようにしろ」とか「カタカナ、横文字は画面から追放してくれ」みたいに常識を超える注文が出てきたりする。ログインをなくせってのはさすがに危ないけど、負担の少ない認証方法を考える価値はありそうだ。表記の変更なんて、ソースがあれば何てことはない……もちろん適当な日本語表記があれば、だけど。

米国などの労働組合では、「.union」を労組専用のトップドメインに認めろなど、ITの活用が進んでいる。ログインに抵抗を感じているようでは心もとないけど、日本の労組もITに目を向けるようになったのは大きな前進だろう。あとは、どうすればネットやコンピュータを実際の彼らの運動の武器として使えるようにするかというところが問題だ。

まずはパソコンを使うところから始まるのだけど、彼らには「パソコンなんて、めんどくさいだけのワープロだ」なんて言われたりする。そんな話を聞きながら、最近Linuxあたりをベースにして、ワープロ、メール、ネットに特化したシンプルなシステムを組めないもんだろうかなどと考えているんだけれど、どうだろう？　単機能マシンは売れないというのが通説だけど、商売じゃないんだから売れなくてもかまわない。慣れてきたら、商用システムやLinuxのフルシステムに入れ替えればいいだけの話だ。また、ちょっと古めのマシンにLinuxを入れて、労組の事務所のネットワークサーバにしても面白いかも知れないと思う。わかりにくい複雑な機能は全部取り払って、必要最低限の機能で気軽に使えるサーバがあれば、面白い使い方ができるかもしれない。

これまでLinuxをはじめとするオープンソースのシステムは、商用システムに負けるなとばかりに、最先端の機能で張り合ってきた。しかし、商用システムにはマネできないフレキシブルな構成が可能というオープンソースの特性を生かした使い方は、もっと研究の価値があるんじゃないだろうか、なんてことを考えているのである。

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。

＊アプリケーションサーバ
CGIなどに代わり、Webサーバとデータベースなどバックエンドのサーバとの仲立ちをするミドルウェア。

ドクターShiodaの

ギョーカイSnapShot

PC業界今月の話題

文：塩田紳二
Text : Shinji Shioda

2・13 Napsterに違法判決
2・22 汎用JPドメイン優先登録受け付け始まる
2・26 Intel Developer Forum開催
3・8 Yahoo!業績の下方修正とCEO辞任
3・13 Transmeta MIDORI Linux公開

なんだか、最近、急にPDAが増えちゃって……、実際には、いろいろ買ってしまったのはいいが、同時にいくつも使うことはできないので、それがあちこちに転がっている状態である。Palm系が多いのだが、とりあえず、スケジュールや住所録を入れるのは1つにして、それぞれのPDAを周辺機器と組み合わせて専用機化して使うことにした。たとえば、VISORにはカメラモジュールをつけてデジカメにして、PalmVxにはGPSアダプタをつけてNavigator化するなどである。だけどね、やっぱり、いくつもPDA持ち歩くのってなんか変な感じ。

IntelのIDFは地味になった？

2月26日から3月1日まで、米国でIntelのIDF（Intel Developer Forum）が開催された。これは、年2回開催されるIntelの顧客向けの技術コンファレンス。ここでは、その時点でのIntelの方向性などが公開され、まあ業界では有名なイベントのひとつ。

さて、前回（昨年8月）は、米国の景気が良かったこともあって、かなり派手なイベントという印象だったが、今回は景気が下向きになったこともあり、かつてのハデさがなくなった。

具体的には、Intelはメインのビジネスであるプロセッサビジネスに回帰という印象だ。行われたキーノートスピーチでは、自社製品を4つのアーキテクチャに分類してみせた。去年までは、「インターネットのビルディングブロックを提供する会社」とかなり広い範囲のビジネスを行うような表現だったのに対し、今回はCPUビジネスを全面に持ってきた。簡単にいえば、CPUビジネスに注力するということだろう。

4つのアーキテクチャとは、IA-32（PentiumIIIと4）とIA-64（Itanium）、StringARM/Xscale、そしてネットワークプロセッサである。これらにそれぞれ「IA-32」、「Itanium Processor Family」、「Internet Personal Client」（StringARM）、「Internet Exchange Architecture」と名付けている。

ただ、だからといってIntelの活動が止まったわけではない。今回は、次期ItaniumであるMcKinleyのデモが行われ、Pentium4版のXeonや次期Pentium4についても話が出た。Intelとしては、AMDとの競争もあるので、世の中が不景気だからといって、停滞してもいられないわけだ。

詳細なスペックの公開はなかったが、次期拡張バスについても話が出た。これは、現在のPCIの後継となる新しいI/Oアーキテクチャである。CPUが高速化し、ネットワークがブロードバンド化しつつある現在、PCIバスはそろそろ飽和状態になりつつある。AGPを使ってディスプレイデバイスを別にして多少帯域を空けたが、たとえば、IEEE1394やUSB2.0から画像を取り込み、ハードディスクにセーブしつつ、表示を行うなんて用途では、PCIも手一杯になり、さらにここで100Base-Tを使うなんてことになると、ちょっと苦しい。また、ライバルであるAMDは、Hyper transportという新しい高速バスを提案している手前、Intelも何かやらざるを得ない状態である。

そこで今回は、やるという話だけして、次回のIDFまでに急いでスペックをまとめようというねらい。高速のシリアルバスを使い、最大10GHz程度までスケーラブルに拡張可能というのが基本スペックなのだが、果たして今年の夏（次回IDFは8月末に予定されている）にどんなものが出てくるのか、今年の楽しみのひとつである。

低消費電力CPUをサーバ用に

さてIntelは、現在デスクトップ系でAMDと、モバイル系でTransmetaと競合関係にあるが、TransmetaのCrusoeは、サーバにも使われ始めたようだ。低消費電力でファン不要ということは、サーバで考えれば、ファンなどのメンテナンスが不要で、発熱が小さいためコンパクトにできるということである。このため、いわゆるThinサーバなどとは意外に相性がいいわけだ。

Intelはこれに対抗して、Ultra Denseというサーバを開発している。これは、ブレードと呼ばれる小型のCPU基板にモバイル用の低消費電力CPUを載せ、HDDと組み合わせて1つのサーバを構成、これを匡体に複数格納できるようにして多数のサーバをより少ない設置スペースで実現しようというもの。こちらは、このブレードのボードサイズやコネクタなども規格化（実際には、Compact PCIの仕様をベースにしている）し、匡体やブレードを別々に製品化できるようにしてある。つまり、単にモバイル用CPUをサーバに使いましょうというだけでなく、さまざまな規格化と一緒にして、サードパーティを動かしていこうという考え方なのだ。

一方、Transmetaは、いままでMobile Linuxと言われていたものを公開した。なんと名前は、Midori Linuxで、Midoriは、日本語の「緑」つまり、グリーンで地球環境に優しいというイメージを表すという。なんだか、日本のアニメファンの米国人が描いたようなイメージキャラクタが使われているが、はたして、こっちはどうなるんでしょうか。Intelも対抗して「ki-midori」なんてLinuxを出す……はずはないか。

WindowsをXPにしたMicrosoftは？

Microsoftは、次期Windowsの名前をWindows XPとした。このWindows XPは、コードネームWhistlerと呼ばれているWindows 2000の後継バージョンをベースにしたWindows 2002といってもいいものである。これを整理すると、

Windows ME→Windows XP Personal
Windows 2000 Professional→Windows XP Professional
Windows 2000 Server→Windows2002 Server ？

という感じになりそう。つまり、デスクトップマシン用のOSは、NTカーネルでWindows 2000の後継だが、名称は、Windows XPとなるわけだ。

また、サーバ系は、Windows 2002など、現在のWindows 2000の直接後継であるような名前が使われるという。さらにこれに合わせてOfficeもOffice XPという名称になる予定。なんだか、XPだらけである。本当は、Windows 2000ですべてのOSのカーネルをNTカーネルにしようとしていたのに、それができずにWindows MEになったわけで、こんどは、本当にNTカーネルになるから、じゃあ、MEに合わせて次はXPだ、ってことなんでしょうね。

Napsterに違法判決

前から揉めているNapsterとレコード業界だが、控訴審でNapsterが著作権侵害行為を意図的に助長したとし、違法の判断を下した。ただし、サーバの停止は命令されてはいない。

いろいろと話題にはなるが、技術としてみれば、Napsterは、インターネットで個人ベースの情報交換という仕組みをつくり、それはいまではピアtoピア技術として認められるまでになっている。かのIntelも、この技術に注目しているし、6月に開かれるJavaOneで、SUNもピアtoピア技術をJavaでサポートするための仕組みを発表する予定だという。

従来のサーバ・クライアント技術では、サーバへの負荷の集中やネットワークのトラフィックの集中が問題になった。特にファイルのダウンロードは、ユーザー側のネットワーク接続速度がアナログモデムであることがほとんどのため、長時間にわたって、サーバに負荷をかける結果となった。Napsterはある意味、これを解決したのである。ただし、その結果生まれたのは、違法コピーされた音楽ファイルの交換である。これでNapsterは違法音楽ファイルを流すことはできなくなったが、ユーザーはファイル名を変えたり、暗号化するなどしておそらくしばらくはNapsterを使い続けるだろうし、サーバのないGnutelaというピアtoピアシステムもある。そういえば、Gnutellaに悪質なウィルスGnumanが出たとか。じつはレコード会社の陰謀だったりして。

TransmetaはMobile Linux改めMidori Linuxを発表した。
http://midori.transmeta.com/

正しきタコへの道

初級Linuxer養成講座

第20回　シェルスクリプトをかじってみる

バックグラウンドでコマンドを実行するということは、ユーザーの目の届かないところで処理を行うことにほかならない。これは、複雑な処理を簡単なコマンド1つで実行できるようにする「シェルスクリプト」でも同じことである。Linuxが「得意」とする「バッチ処理」のもっとも一般的な形態である「シェルスクリプト」の使い方を習得してみよう。

文：竹田善太郎
Text : Zentaro Takeda

数カ月前、自宅にやっと「フレッツ・ISDN」が開通して、インターネットに**つなぎ放題**の環境が手に入った。しかし、通信速度があまりにも遅い。同じ64Kbpsのはずなのに、通常のアクセスポイントに接続する場合に比べて、フレッツ・ISDNで接続したほうが、1〜2割ほど実効速度が落ちてしまう。ひどいときなど、20Kbps程度にまで落ちることがある。フレッツ・ISDNに切り替えたときに、同時にプロバイダも乗り換えたことが原因のひとつかもしれないが、どうも**地域IP網**というのがくせものらしい。

テレホーダイタイムになると、ぐぐっと速度が落ちるのは相変わらずなのだが、関係ない時間帯に**いきなり重く**なることもあって、どうも安定しないのだ。DNSへの問い合わせの応答が返ってこなくて、Webサイトにつながらなくなってしまうこともよくあるし、大きなファイルのダウンロード途中に、みるみる速度が落ちていって、**ついにタイムアウト**してしまうことも少なくない。こんな症状が出たときには、**tracerouteコマンド**を使って調べるのが常套手段だが、どうも地域IP網から外に出るゲートウェイのところで**詰まって**しまっているようだ。

遊びで使っているだけなら問題ないのだが、インターネットを仕事の道具として使っている身としては、ちょっとつらい。かといって、**毎月数万円**もの出費が強いられる専用線を導入できるほど**事業規模は大きくない**。

そんな中で、やっとフレッツ・ISDNが通ったばかりだというのに、今度は某社のADSLサービスが自宅エリアでも使えるようになったらしい。個人用の常時接続サービスといってもいろいろあって、たとえば**ルータ禁止**などという**前時代的な制限**を設けているところもあるようだが、ここのサービスではルータもOKということなので、さっそく申し込むことにした。

「ADSLが速い」というのは、知人の仕事場や「高速インターネット装備」のホテルなどで体験してはいるのだが、本当のところを言うと**あまり期待はしていない**。毎日、ほぼ12時間近くもインターネットを使い続けている経験から言わせてもらうと、どうも日本のインターネットは、**根幹の部分が朽ちかけている**ような気がするのだ。

こんなことを書くと、ネットワークの構築や維持を仕事にしている人から怒られるかもしれないのだが、バックボーンのネットワークをきちんと整備しないまま、**株価の吊り上げ**を目

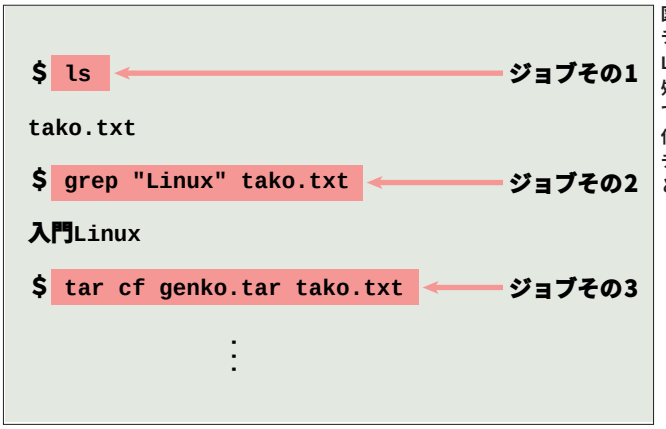


図1　コマンドラインも「バッチ処理」の一種
Linuxのコマンドラインでは、処理させたい仕事を1行単位で入力する。これも、1行単位の小さな「ジョブ」をバッチ処理させていると考えることもできる。

的に、やれADSLだ、やれ光ファイバだと、**見た目のアクセス回線だけ速くしている**事業者が少なくないのでは、と疑心暗鬼になっている。

実は、今回申し込んだADSLサービスは、現在フレッツ・ISDNで利用しているのと同じプロバイダへのアクセス回線になるのだが、ISDNのバンド幅すらまともにカバーできない状況なのに、ADSLになってもそれほど速くなるとは思えない。一応**準大手**で**古株**のプロバイダなので、筆者の思い過ごしなのかもしれないが、いろいろ調べてみる限り、ADSLや光ファイバのアクセス回線も、フレッツ・ISDNの地域IP網と同じように**ローカルな**ネットワーク接続が速くなるだけで、そこから外部へアクセスしようとすると、とたんに性能が落ちるらしい。

結局、LANがいくら速くなっても、インターネットの使い勝手が変わらないのと同じで、やれ10Mbpsだ100Mbpsだといっても、プロバイダがローカルに提供している**娯楽コンテンツ**を見るのが快適になる程度で、インターネットそのものの使い心地はあまり変わらないのではないだろうか。ましてや、一時的なブームに浮かれて、バックボーンの容量に見合わない高速アクセス回線を無闇に増やしてしまうと、ネットワーク全体のバランスが崩れて**目も当てられない惨状**になるのでは、と心配している。

「バッチ処理」の正体

さて、本誌を含め、Linux関連のいろいろな記事を読んでみると、「**Linuxはバッチ処理が得意**だから云々…」という記述をいたるところで見かける。かくいう筆者自身も、このフレーズは便利に使わせてもらっている。ところが、バッチ処理という言葉を各種の用語集で調べてみると、「前もって決められた処理を、まとめて実行する」という、わかったようなわからないような解説がついていることが多い。

では、「バッチ処理」と対になる意味を持つ言葉は何かといえば、「リアルタイム処理」、「時分割処理」、「オンライン処理」、「対話型処理」など、さまざまな言葉が思いつくのだが、ソフトウェアの専門家でもないかぎり、どんなものなのか見当もつかないだろう。ましてや、WindowsやMacintoshなどのグラフィカルなOSが主流の現在では、「バッチ処理」がなんであるかなど知らなくても、**ヘビーユーザー**を名乗ることができるのだ。

「バッチ処理」の意味を先にあげた「反対語」から探ってみると、「リアルタイムでなく、オンラインでなく、対話型でない処理」ということになる(「時分割」については、ちょっと話が複雑になるので省略する)。言葉をもう少し変えると、「いつ終わるかわからず、実行中に何が起こっているかわからず、実行時にユーザーが介入できない処理」ということで、さらに噛み砕いてしまえば、「**ユーザーの目や手の届かないところで実行される処理**」ということになるだろう。一言で言えば**無人処理**や**自動処理**という言葉がぴったりくる。

いったん実行してしまったら、結果が出るまでユーザーはじっと待ち続けるしかない、という意味からすると、コマンドラインで1行ずつコマンドを実行するスタイルも「バッチ処理」の一種である(**図1**)。実行中のコマンドラインのことを**ジョブ**と呼ぶことは、何回か前に説明したと思うが、この「ジョブ」という用語は「バッチ」とほぼ同じ意味で使われることがあるのだ。

●ちょっと昔話

そもそも「バッチ処理」とは、**大昔**(といっても十数年くらい前まで)の**メインフレームコンピュータ**において、大勢の利用者が1台のコンピュータを共同利用する方法の一種だった。コンピュータを利用するユーザーは、あらかじめコンピュータ上で実行させたいプログラムやデータを、**パンチカード**や**磁気テープ**に記録しておいて、それを抱えてコンピュータセンターに出向き、係員に渡したり、自分でカードリーダーやテープドライブに読み込ませる。プログラムは、原則として受付順に1つずつ実行されるのだ(**エラいセンセイ**や、締め切り間際のプロジェクトなどは優先的にジョブを**投入**してもらえるようだったが……)。

コンピュータセンターの壁には、ジ

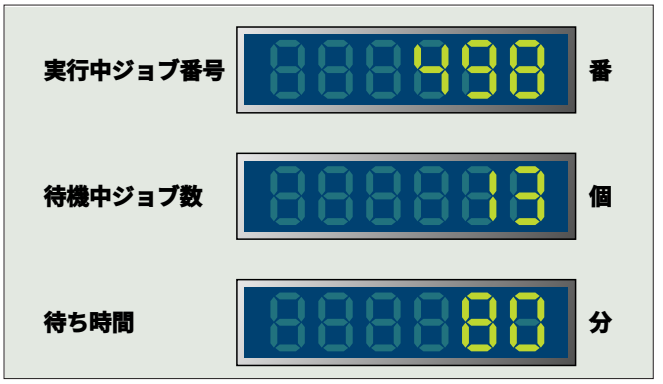


図2 「ジョブ進行状況表示器」の再現図
昔の「計算機センター」には、大病院の薬局や銀行の窓口にあるような「待ち時間表示装置」があった。この表示器を見て、「今日は混んでいるな」とか「今なら待ちなしで済む!」と一喜一憂したものである。

ョブ進行状況というような名前の電光掲示板（図2）があって、その時点で読み込ませたジョブが、何時間後に終了するかが表示されている。そして、ジョブの終了予定時刻まで、近くの喫茶店で時間をつぶすことが多かったのだ。

いったん登録してしまったジョブは、実行されるまで結果がどうなるかわからない。終わったころを見計らってセンターに行ってみると、エラーが起こっていて、それまでの努力が水の泡になることもある。無限ループに陥るようなプログラムを実行させてしまって、プリンタ用紙を大量に無駄にして、センターの係員から大目玉をくらう、というようなこともある。

コンピュータの性能が上がり、価格が安くなってくると、このような原始的なバッチ処理はごく特殊な場面、たとえば、スーパーコンピュータ（これすらも、現在では死語になりつつあるようだが）を利用するような場合に限られてきて、1台のコンピュータに端末をたくさん接続して、複数のユーザーが同時に利用する「時分割処理」とか「マルチタスク処理」が普通になってきた。1台1台の端末を使って、まるで自分専用のコンピュータであるかのように使えるようになったのだが、プログラムを実行させるときは、端末上でエディタを使うなどしてプログラムやデータを作成し、さらにジョブ制御言語なる難解なプログラミング言語を使って、プログラムを実行させるためのプログラムを記述して、初めて目的のプログラムを動かせるようになる。

なんとも面倒くさそうな話だが、Linuxの祖先のUNIXは、このような時期に生まれたOSなので、古い時代のコンピュータの文化が色濃く残っている。実行中のコマンドラインのことをジョブと呼ぶあたりが、その顕著な例だろう。しかし、UNIXには対話型シェルが備わっているのが、それまでのコンピュータとの大きな違いだった。面倒なジョブ制御言語を使わなくても、コマンドラインにプログラムファイルの名前とデータファイルの名前を入れてやるだけで、すぐにプログラムを実行できるようになったのだ。

●それでも便利なバッチ処理

しかし、決まりきった作業をさせるのに、毎回同じコマンドラインを入力するのは面倒だし、ちょっと複雑な作業をさせる場合には、コマンドラインを間違えると、もう一度最初からやり直さなければならなくなる。

そこで、UNIXには、旧式のバッチ処理と同じように、プログラムの実行手順をあらかじめ記述しておいて、それをまとめて実行できるような仕組みも残されたのだ。これが、いわゆるシェルスクリプトの始まりである。

スクリプトとプログラム

シェルスクリプトに限らず、コンピュータを使っているとスクリプトという言葉を頻繁に耳にするだろう。最近ではJavaスクリプトやVBスクリプトという単語をよく聞く。これらはWebページの作成にからんでくるものなのだが、スクリプト（script）という言葉を辞書で引くと、台本とか筋書きという言葉と並んで、行動予定という言葉がある。

結局、スクリプトとは、コンピュータの行動予定を書き記したもの、というのがぴったりきそうだ、それならプログラムも同じ意味では、と思うかもしれない。まったくそのとおりで、スクリプトはプログラムの一種と考えておけばよい。「シェルスクリプト」のことを「シェルプログラム」と呼ぶこともあるのだ。

厳密には、C言語やBASICなどの「プログラミング言語」で書かれたものが「プログラム」、それ以外の、たとえばアプリケーションの「マクロ言語」のようなもので書かれたものをスクリプトと区別することもあるようだが、それではPerlで書いたものはどうなるのかとか、「Visual Basic for Application」はプログラミング言語といえるのか、という問題もあるので、このような区別にはあまり意味はない。「プログラム」のことを気取っていうと「スクリプト」になる、とでも考えてもらって結構だ。

画面1　単純なシェルスクリプトその1
実行したいコマンドを、1行ずつ順にテキストファイルに書いておけば、その順にコマンドが実行される。

```
kterm
[zen-t@zen06 zen-t]$ cat simple
echo "aaa"
echo "bbb"
echo "ccc"

[zen-t@zen06 zen-t]$ bash simple
aaa
bbb
ccc
[zen-t@zen06 zen-t]$
```

ちょっと脱線してしまったが、ともかくスクリプトとは、コンピュータに実行させたい作業を書き付けたものであり、シェルスクリプトとは、ふだんシェル上で実行しているコマンドラインを、実行する順番に書き出したもの、と考えていてほしい。実際には、単に順番に書くだけではなく、もう少し複雑なこともできるのだが、それについてはまたあとで触れる。

シェルスクリプトを「書く」

コンピュータに作業手順を示す、といっても、まさか**紙に書いたもの**を読ませるわけにはいかない。通常は、テキストファイルにコマンド文字列を順番に書いたものを作るのだ。

Linuxのシェル（bash）の場合、シェルスクリプトの先頭には、次のような行を入れる**きまり**になっている。

```
#!/bin/bash
```

これは、「このファイルは/bin/bashで実行するために書かれたもの」であることを意味する。

あとは、この下に、実行させたいコマンドラインを次々に書いてゆけばよい。ファイルの1行がコマンドライン1回分に相当する。たとえば、

```
echo "aaa"
echo "bbb"
echo "ccc"
```

と書けば、これが1行ずつ順番に実行されて、画面には、

```
aaa
bbb
ccc
```

と表示されるようになるのだ（**画面1**）。

シェルスクリプトの中に書くコマンドラインは、ふだんシェルのコマンドラインから入力しているものと同じである。使えるコマンドの種類も、ほとんど変わらない。たとえば、「shimatsusho.txtというファイルをviで編集して、その中身をmyboss@mycompany.co.jpに送信する」というような手順を、シェルスクリプトに書くこともできる。

```
#!/bin/bash
vi shimatsusho.txt
mail myboss@mycompany.co.jp
< shimatsusho.txt
```

このスクリプトを「daily」というような名前で保存しておけば、日々の始末書の提出作業がコマンド1発でできるようになる（**画面2**）。

シェルスクリプトを実行する

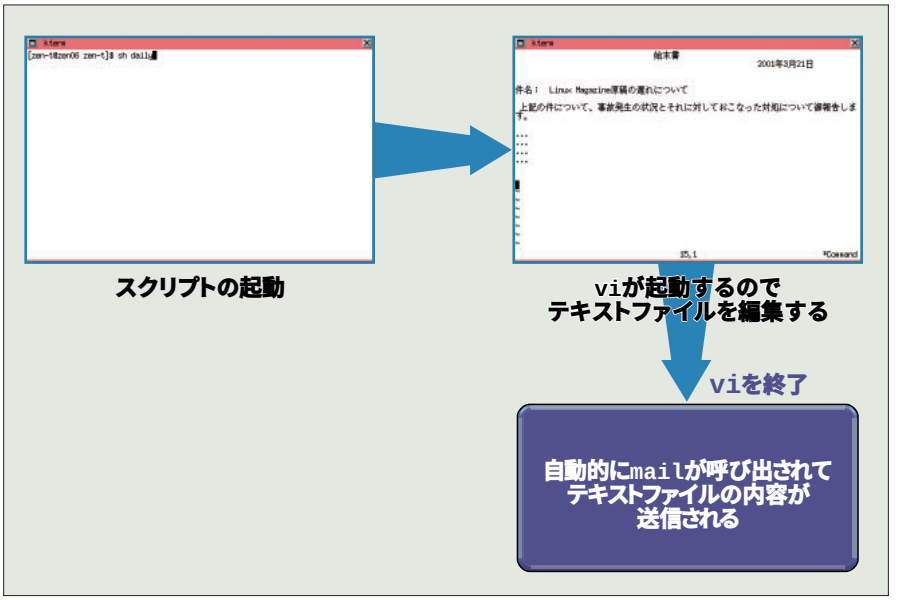
さて、シェルスクリプトの中身に何を書けばよいのかは後回しにすることにして、記述したシェルスクリプトを実行する方法は何通りかある。まず、次のようにしてbashを新たに起動して、シェルスクリプトの内容を直接実行させる方法だ。

```
$ bash daily
```

すると、ファイル「daily」の中身に従って、viが起動して編集画面になり、編集作業を終えてviを終了させれば、自動的にmailコマンドが実行されて、テキストファイルの中身が送信されるようになる。

スクリプトを実行するのに、**いちいちbashと入れるのは面倒**と思うのであれば、**スクリプトファイル自体をコマンドファイルにする**方法もある。シェルスクリプトのファイルに**実行可能属性**をつけてやるのだ。

ファイルの属性を変更するのが**chmodコマンド**だ。これについては、本連載の第9回にも説明したが、スクリプトファイルを実行可能状態に



画面2　単純なシェルスクリプトその2（始末書提出スクリプト）
viのようなアプリケーションをスクリプトから起動することもできる。アプリケーションを終了させれば、自動的に次のコマンドの実行に移るわけだ。

設定するには、次のようにchmodコマンドを使う。

```
$ chmod +x daily
```

ここで、

```
$ ls -l daily
```

を実行してみると、

```
-rwxrwxr-x  1 user1 user1  31 Mar
27   daily
```

などと表示されて、実行可能属性（x属性）が追加されたことがわかるだろう。

スクリプトのファイルに実行可能属性をつけると、いちいち「bash」と入力しなくても、シェルプログラム（これもbashなのだが）がファイルの先頭行を調べて、そのプログラムを起動し、ファイルの内容を実行させるように仕向けてくれる。コマンドラインでは、次のように入力してやればよい。

```
$ ./daily
```

ここで、ファイル名の前に./と付いているのは、**カレントディレクトリにあるファイルを実行しろ**という意味なのだが、先ほどの「bash daily」ではこのようなものは必要なかった。**なんだ、結局面倒じゃないか**と思うだろうが、このようなものが必要な理由について話すと長くなるので、ここでは、カレントディレクトリのスクリプトを実行するには**「./」を頭につけなければならない**、というように覚えておいてほしい。

ところで、ここまでに説明したシェルスクリプトの2通りの実行方法では、一方では「bash」を起動し、もう一方では起動していないように見えるのだが、実際はどちらもコマンドラインの入力を受け付けているbash以外に、シェルスクリプトを読み込んで仕事を遂行する**別のbash**が起動されていることに変わりはない（**図3**）。もし、実行するスクリプトの先頭に、

```
#!/usr/bin/perl
```

という行があれば、bashではなくPerlが起動されて、スクリプトの内容を処理することになる。逆にこのように記述されたスクリプトを、

```
# bash perlscript
```

というように実行しようとすると、

```
perlscript: command not found
```

というようなエラーメッセージが表示されることになる。そういう意味でも、スクリプトに実行属性をつけて、

```
# ./perlscript
```

というように実行させたほうが、そのスクリプトがどのようなプログラムで実行するように書かれたものなのかをいちいち意識しなくてもすむので、楽である。

ちなみに、「#!……」のような行を省略すると、その時点で使用しているシェルプログラムのスクリプトであるとみなして、実行しようとする。

シェルスクリプトを実行する際に、新しいbashが起動されていることを確認する方法があるので、ちょっと実験してみよう（**画面3**）。

bashには**$$**というちょっとへんな名前の**シェル変数**がある。シェル変数とはなにか、どのように使うのかについては、次回に説明することにするが、この「$$」は、現在コマンドを読

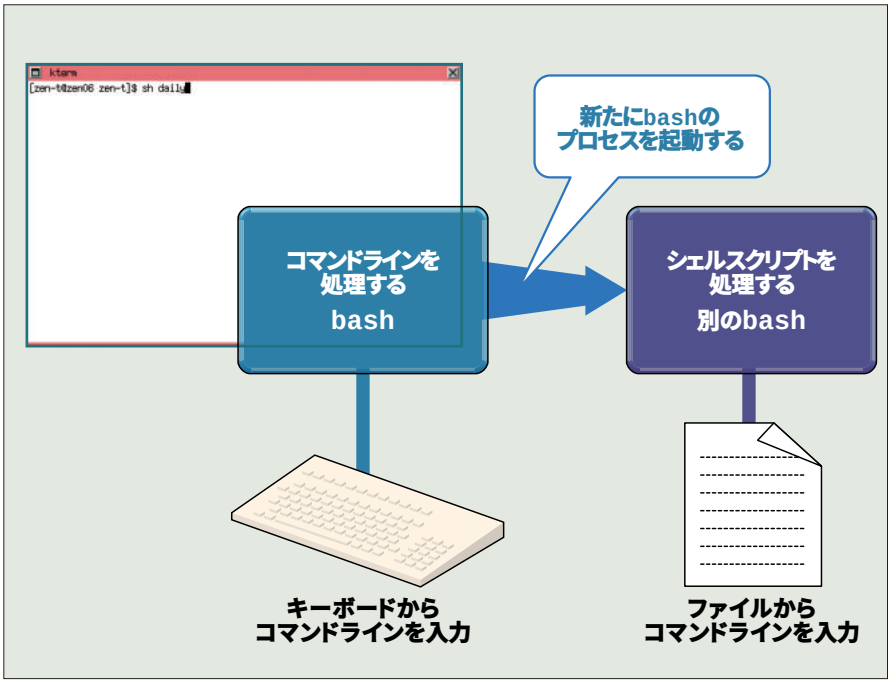


図3　シェルスクリプトを実行する「シェル」
シェルスクリプトを実行すると、それまでコマンドラインを処理していたシェルとは別に、別のプロセスとしてシェルが起動される。この仕組みを覚えておかないと、あとで「シェル変数」などを扱うときに混乱することになる。

み込んで実行しているbash自身のプロセス番号（プロセスID＝PID）を表わすものだ。たとえば、echoコマンドで$$の値を表示させてみると、現在コマンドラインを処理しているbashのPIDが表示されるはずだ。

```
$ echo $$
607
```

次に、このシェル変数を表示するだけのシェルスクリプト「subshell」を作成してみる。subshellの内容は、次の2行だけである。

```
#!/bin/bash
echo $$
```

これに実行属性をつけて、実際に動かしてみる。

```
$ chmod +x subshell
$ ./subshell
1284
```

先ほどとは別のPIDが表示されると思う。これは、次のようにシェルスクリプトを実行しても同じである。

```
$ bash subshell
1285
```

このように、シェルスクリプトを起動するたびに、新しいbashのプロセスが起動されていることがわかってもらえただろうか。

ちなみに、何度シェルスクリプトを実行しても、大元のコマンドラインを処理しているbashのPIDは変わらない。

```
$ echo $$
607
```

MS-DOSの「バッチファイル」（Linuxのシェルスクリプトに似たようなもの）に慣れている人だと、Linux（UNIX）のシェルのこのような性質に気づかずに、とんでもない間違いをしてしまうことがある。マルチタスクではないDOSのバッチファイルでは、特別なコマンドの書き方をしない限り、コマンドラインを実行しているシェルそのものがバッチファイルを処理しているからだ。

シェルスクリプトの実行結果は？

先ほどの「始末書提出スクリプト」は、スクリプトの例としてはあまり一般的でないかもしれない。しかし、このような目的でスクリプトを使うことに、何の問題もない。要は、**面倒な定例作業**を手順書にして、自動的に実行させることがスクリプトのもっとも大きな利点だからだ。

しかし、一般的なスクリプトでは、viのように、ユーザーが直接操作するようなプログラムを起動することは少ない。それよりも、grepとかsedとかfindといったように、いったん実行してしまったら、あとは結果が出るのを待つようなタイプのコマンドを使うことが多い。

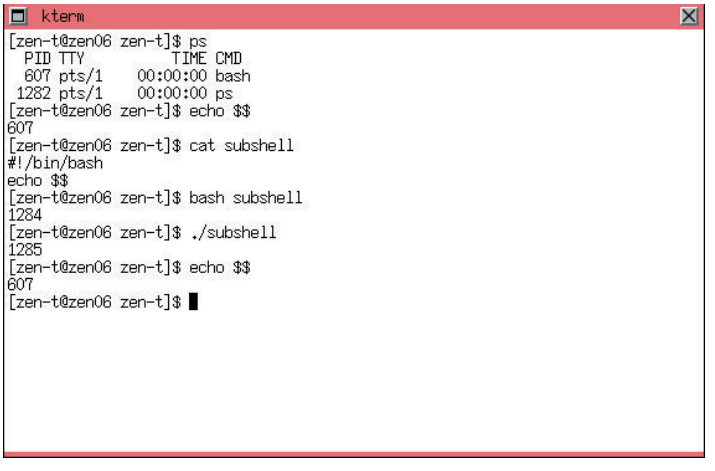
たとえば、カレントディレクトリ以下にある、「.txt」という拡張子をもつファイルを探して一覧表を作り、さらにその一覧の中から「Linux」という文字列を含むファイルを検索して一覧を作る、というような作業を簡単なスクリプトで記述すると、次のようになる。

```
#!/bin/bash
find . -name "*.txt" > a.txt
grep "Linux" < a.txt > b.txt
```

このスクリプトを実行すると、カレントディレクトリにa.txtとb.txtの2つのファイルが作られて、それぞれの中にコマンドの実行結果が保存されることになる。

スクリプトを使って複雑な作業を行う場合、このように**リダイレクト**を活用することが多い。また、シェルスクリプトは1回実行してしまえば、あとは終了するまで何もしないで待つ、ということで**バックグラウンドジョブ**にも似ている。そういう意味で、シェルスクリプトをうまく記述するには、リダイレクトとバックグラウンド実行の知識が必要だったのだ。

ここで誌面が尽きたようなので、次回はもう少し具体的なシェルスクリプトの作成方法を解説しよう。



画面3　bashのプロセス番号を表示させる
bashの「$$」というシェル変数には、現在のプロセス番号が格納されている。この値を表示するシェルスクリプトを実行させてみると、親のシェル（PID 603）とは別のシェル（PID 1284〜5）によってスクリプトが実行されているのがわかる。

InterBase 6.0

前回は見積書発行システムのスキーマを設計しました。クライアントの作成に入る前にInterBaseのセキュリティとユーザー管理について考えてみましょう。さらに、ロールと特権についても解説します。そして、InterBaseでのより細かいセキュリティ管理の実現方法を解説します。

第6回　セキュリティとユーザー管理

文：加藤大受
Text:Daiju Kato　kato@jcom.home.ne.jp

Database

前回、データベースを使用するシステムでは、データベースサーバのユーザー機能を使用せずに、独自にユーザー管理を行うほうが好ましいと説明しました。データベースサーバは非常に使いやすいユーザー管理機能を持っていますが、複数のシステムで1つのデータベースサーバを使用する場合や、システムの画面ごとや機能ごとにセキュリティを設定していきたいと考えると、独自のセキュリティシステムをシステムに組み入れるほうが使いやすくなるでしょう。今回は、InterBaseが持っているセキュリティ機能について説明し、独自にセキュリティシステムを作成する場合の方法などについて解説していきます。

InterBaseのユーザー管理の方法

リレーショナルデータベースを使用するには、データベースサーバにログインするためのユーザーが必要になります。InterBaseの場合では、「SYSDBA」という管理者用のユーザーが用意されており、すべての権限を持っていま

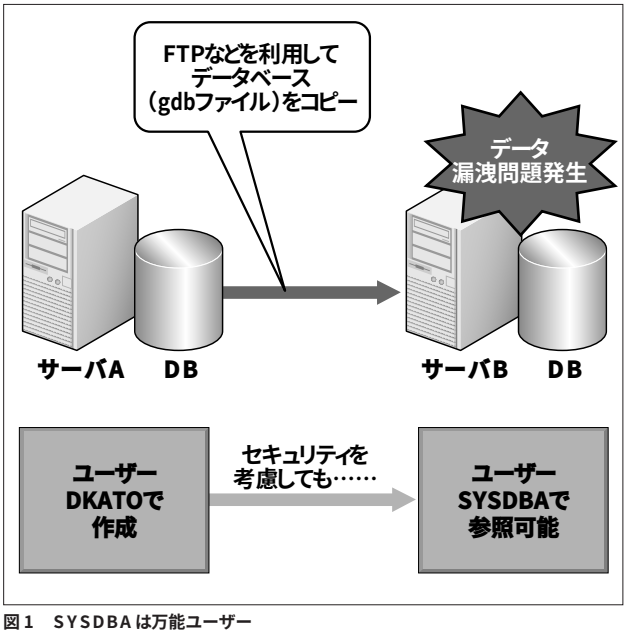


図1　SYSDBAは万能ユーザー

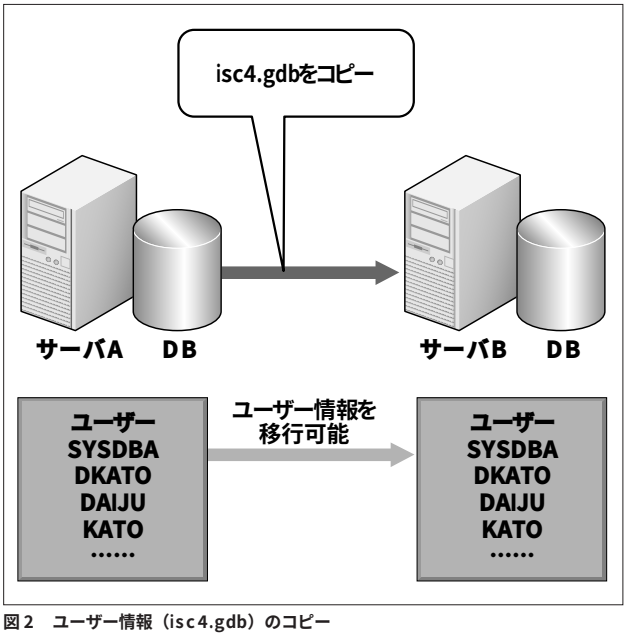


図2　ユーザー情報（isc4.gdb）のコピー

す。では、すべての権限を持っているとはどういうことなのでしょうか？

たとえば、サーバAにInterBaseをインストールし、そこで何らかのシステムを稼働させたと仮定します。セキュリティを考慮し、SYSDBAではないユーザーでデータベースを作成し、SYSDBAのパスワードをデフォルトの「masterkey」から変更しておきました。通常であれば、これでセキュリティは守られているのですが、もし誰かがサーバBを立て、サーバAからInterBaseのデータベースファイルをコピーし、サーバBにInterBaseをインストールしてSYSDBAの管理者用のユーザーを使うと、データベース内のデータをすべて操作できることになります。

コマンド	説明
di[splay]	isc4.gdb のすべての行を表示
di[splay] name	ユーザーnameの情報だけを表示
a[dd] name -pw passwd [option argument] [option argument ...]	ユーザーnameをパスワードstringとともにisc4.gdbに追加。optionとargumentでユーザーのほかのデータを指定
mo[dify] name	a[dd]と同様。nameがisc4.gdbにすでに存在する場合に使用
de[lete] name	isc4.gdbからユーザーnameを削除
h[elp]	gsecコマンドと構文を表示
?	gsecコマンドと構文を表示
q[uit]	対話形式を終了

表1　gsecの対話形式でのコマンド一覧

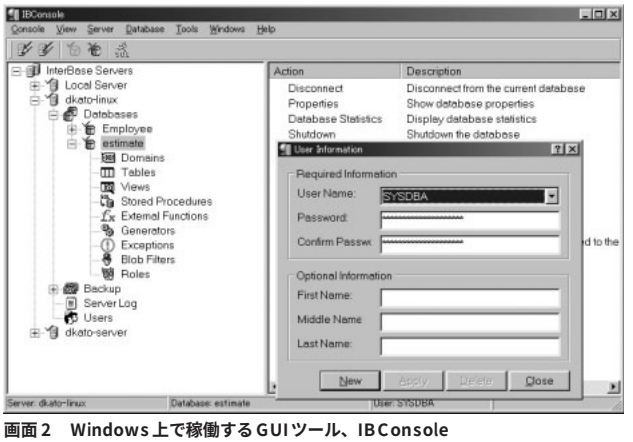
オプション	説明
-password 文字列	変更を行うユーザーのパスワード
-pa 文字列	変更を行うユーザーのパスワード
-user 文字列	変更を行うユーザー
-pw 文字列	ターゲットユーザーのパスワード
-uid 整数	ターゲットユーザーのID
-gid 整数	ターゲットユーザーのグループID
-fname 文字列	ターゲットユーザーのフルネーム
-mname 文字列	ターゲットユーザーのミドルネーム
-lname 文字列	ターゲットユーザーの姓

表2　gsecのオプション

InterBaseの場合、データベースが通常、gdbという拡張子を持つファイル1つにまとめられるため、ファイルをコピーするだけでもバックアップを実行できるという特徴を持っていますが、この機能があだになってセキュリティホールになる可能性があるわけです（**図1**）。

InterBaseのユーザーは各サーバごとに管理されています。InterBaseのルートディレクトリ（デフォルトは/opt/interbase/）のisc4.gdbというファイルでユーザー情報を管理しています。このファイルもInterBaseのデータベースで、SYSDBAであれば開くことができます（**画面1**）。なお、パスワードは暗号化されていますので、このファイルを開いてもパスワードが盗まれることはありません。

つまり、isec4.gdbをコピーすることで、InterBaseサーバのユーザー情報をサーバAから、サーバBに移行することができるのです（**図2**）。この仕組みは、組み込み用途で使う場合などでは非常に便利な特徴ですが、基幹システムで使用する場合にはセキュリティの甘さが問題になる可能性があります。データベースはほかのユーザーに参照されないディレクトリに置き、コピーなどによってデータの



画面2　Windows上で稼働するGUIツール、IBConsole

```
$ /opt/interbase/isql -U SYSDBA -P masterkey
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect '/opt/interbase/isc4.gdb';
Database:  '/opt/interbase/isc4.gdb', User: SYSDBA
SQL> show database;
Database: /opt/interbase/isc4.gdb
        Owner: SYSDBA
PAGE_SIZE 4096
Number of DB pages allocated = 151
Sweep interval = 20000
SQL> show tables;
       HOST_INFO
SQL> quit;
```

ISQLでユーザー情報データベースを参照
ユーザー情報のデータベースに接続
データベース情報
テーブル情報

画面1　ユーザー情報が管理されているisc4.gdbファイルの表示

漏洩が起きないように注意してください。

■ユーザーの作成

すでに解説したように、InterBaseのユーザーを作成するには、Windowsプラットフォームで稼働するGUIツールであるIBConsole（**画面2**）を使用するか、gsecと呼ばれるコマンドラインツールを使用します。gsecはコマンドラインオプションを使用してユーザー管理を行うこともできますし、対話形式で行うことも可能です。**表1**は対話形式で使用できる場合のコマンドの一覧で、**表2**はコマンドラインで使用する場合のオプションです。

それでは、DAIJUというユーザー名を作成する場合を、対話形式とコマンドラインオプションの両方で行ってみます。対話形式の場合では、GSEC>というコマンドプロンプトが表示され、コマンド待ち状態になります。ここで、addコマンドを使ってユーザーを作成します。作成したユーザーは、displayコマンドで参照することができます。コマンドラインオプションを使用する場合は、ユーザー名を作成するには、-addというオプションを使用し、ユーザー情報の参照には、-displayオプションを使います。このように、gsecユーテリティは対話形式でも、コマンドラインオプションでもどちらでも使用可能なユーザー管理ツールです（**画面3・4**）。

ロール機能

InterBaseには、ロールという機能が提供されています。ロールはSQL-99にも規定されており、ある特定のセキュリティを持つオブジェクトの名前を作成できる機能です。ロールはCREATE ROLEというSQL構文で作成することができます（**表3**）。たとえば、administratorというロールを作成する場合は**画面5**のように作成します。

これで、administratorというロールの使用が可能になります。作成したロールを使用するには、データベースに接続するときにロールをオプションで指定します。たとえば、isqlを使用するときは、「ROLE」オプションの後にロ

構文	引数	説明
CREATE ROLE	rolename;	rolenameロールに関連付ける名前。データベース内のロール名間で一意でなければならない

表3　CREATE ROLE構文

```
#/opt/interbase/bin/gsec ―― gsecの起動
GSEC> add daiju -pw daiju ―― ユーザー'DAIJU'の追加
GSEC> display ―― 追加したユーザーの表示
     user name                       uid   gid     full name
------------------------------------------------------------------------------
     SYSDBA                          0     0
     DAIJU                           0     0
GSEC>QUIT
```

画面3　gsecの対話形式

```
#/opt/interbase/bin/gsec -add DAIJU -pw DAIJU ―― gsecのコマンドラインオプションでユーザー'DAIJU'を作成
#/opt/interbase/bin/gsec -display ―― 作成したユーザーの表示
     user name                       uid   gid     full name
------------------------------------------------------------------------------
     SYSDBA                          0     0
     DAIJU                           0     0
```

画面4　gsecのコマンドライン

```
$/opt/interbase/bin/isql -U SYSDBA -P masterkey
Use CONNECT or CREATE DATABASE to specify a database
SQL> CONNECT '/home/db/estimate.gdb';
Database:  '/home/db/estimate.gdb'
SQL> CREATE ROLE administrator; ―― ロールの作成
SQL>QUIT;
```

画面5　administratorというロールを作成

```
$/opt/interbase/bin/isql -U SYSDBA -P masterkey
Use CONNECT or CREATE DATABASE to specify a database
SQL> CONNECT '/home/db/estimate.gdb' ROLE
administrator;
Database:  '/home/db/estimate.gdb', Role:
ADMINISTRATOR
SQL>QUIT;
```

画面6　作成したロールの使用

ールの名前を追加します（**画面6**）。

作成したロールには、いろいろな権限を持たせることができます。ちょうど、管理を行いたいときだけスーパーユーザーでログインして処理を行う場合などに似ています。SYSDBAを使うことなく、特定のロールにテーブルの作成・更新機能を追加しておくことで、データのメンテナンスを実現することができるでしょう。

権限について

すべてのユーザーがデータを更新できてしまうと、多くの問題が発生します。これらの問題を解決する方法として、リレーショナルデータベースには権限という概念があります。この権限によって、どのユーザーが、データの閲覧、

特権	実行できる動作
ALL	SELECT、DELETE、INSERT、UPDATE、EXECUTE、REFERENCESのすべての権限を持つ
SELECT	テーブルまたはビューから行を抽出できる
DELETE	テーブルまたはビューから行を削除できる
INSERT	テーブルまたはビューに行を挿入できる
UPDATE	テーブルまたはビューの列の現在値を変更する。指定した一部の列だけに限定することができる
EXECUTE	ストアドプロシージャを実行できる
REFERENCES	外部キーで指定される列を参照できる。少なくとも1次キーに含まれる、すべての列にREFERENCES特権を設定しなければならない

表4　InterBaseの特権

構文	
GRANT <特権> ON [テーブル名｜ビュー名｜列名] TO [オブジェクト名｜ユーザーリスト｜グループ名] TO WITH GRANT OPTION GRANT <特権> EXECUTE PROCEDURE <プロシージャ名> TO [オブジェクト名｜ユーザーリスト] TO WITH GRANT OPTION GRANT <ロール名> TO [PUBLIC｜ロールユーザーリスト] WITH ADMIN OPTION	
引数	**説明**
特権	許可する特権の名前のリスト。指定できる値はSELECT、DELETE、INSERT、UPDATE、REFERENCES
列名	許可される特権が適用される列
テーブル名	許可される特権が適用される既存のテーブル名
ビュー名	許可される特権が適用される既存のビュー名
プロシージャ名	許可される特権が適用される既存のプロシージャ名
グループ名	/etc/groupに定義されているグループ名（UNIXのグループ）
オブジェクト	既存のプロシージャ、トリガー、またはビューの名前
ユーザーリスト	isc4.gdbに登録されているユーザー名、またはCREATE ROLE文で作成したロール名
WITH GRANT OPTION	GRANT文で指定された特権のGRANT特権をユーザーリストに与えることが可能
ロール名	CREATE ROLE文で作成された既存のロール名
PUBLIC	全ユーザーを指すキーワード
ロールユーザーリスト	ロールが与えられるユーザーのリスト。ユーザーはisc4.gdbに登録されていなければならない
WITH ADMIN OPTION	ロールユーザーリストに指定されているロールにGRANT特権を与えることが可能

表5　GRANT構文

構文	
REVOKE [GRANT OPTION FOR] <特権> [テーブル名｜ビュー名｜列名] FROM [オブジェクト｜ユーザーリスト｜ロール名｜グループ名] REVOKE <特権> EXECURE PROCEDURE <プロシージャ名> FROM [オブジェクト｜ユーザーリスト] REVOKE <ロール名> FROM[PUBLIC｜ロールユーザーリスト]	
引数	**説明**
特権	取り消す特権の名前のリスト。指定できる値はSELECT、DELETE、INSERT、UPDATE、REFERENCES
GRANT OPTION FOR	REVOKE文で指定された特権を取り消すとともに特権を付加する権限を外す
列名	取り消す特権が適用される列
テーブル名	取り消す特権が適用される既存のテーブル名
ビュー名	取り消す特権が適用される既存のビュー名
プロシージャ名	取り消す特権が適用される既存のプロシージャ名
グループ名	/etc/groupに定義されているグループ名（UNIXのグループ）
オブジェクト	既存のプロシージャ、トリガー、またはビューの名前
ユーザーリスト	isc4.gdbに登録されているユーザー名、またはCREATE ROLE文で作成したロール名
ロール名	CREATE ROLE文で作成された既存のロール名
PUBLIC	全ユーザーを指すキーワード
ロールユーザーリスト	ロールが与えられるユーザーのリスト。ユーザーはisc4.gdbに登録されていなければならない

表6　REVOKE構文

更新、削除、追加などができるかどうかを管理していきます。権限は一般的に特権と呼ばれます。

通常、スキーマの作成者とスーパーユーザー（InterBaseではSYSDBA）だけが、作成したスキーマをフルコントロールできるすべての特権を持っています。InterBaseでは特権は**表4**のように分類されます。これらの特権を付加していくことで、作成者以外のユーザーがデータを閲覧したり、更新したりすることができるのです。

特権を設定するには、GRANT構文を使用します（**表5**）。特定のユーザーに特権を設定することも、PUBLICというキーワードを使用して、すべてのユーザーに対して設定することもできます。また、ロールに割り当てることもできます。特権を割り当てるだけでなく、特権を割り当てる権限をユーザーやロールに与えることも可能です。

見積もり作成システムで使用するユーザーを作成し、すべてのテーブルを閲覧、更新、追加できる特権と、プロシージャの実行特権を与えてみましょう。まず、システムで使用する「ESTIMATE」というユーザーをgsecで作成します。パスワードはユーザー名と一緒にしておきます。メンテナンス効率を考えて、SQLファイルとしてまとめておき、作成したSQLファイルをisqlで実行します。**リスト1**が作成したSQLファイルです。GRANT構文を使用して特権をESTIMATEユーザーに付加しています。

特権の削除はREVOKE構文で行います（**表6**）。使い方

▼リスト1　特権の設定

```
/*****************************************************************/
/*                                                               */
/*              見積書発行システム                               */
/*                        特権の設定のファイル                   */
/*                                                               */
/*              ファイル名:previlege.sql                         */
/*                                                               */
/*  2001/03/15  Daiju Kato                                       */
/*****************************************************************/

/**** データベースへの接続  ****/
SET NAMES EUCJ_0208;
CONNECT 'dkato-linux:/home/db/estimate.gdb' USER 'SYSDBA' PASSWORD 'masterkey';

/******** 会社マスターの更新権 ***************
GRANT SELECT, UPDATE, INSERT ON COMPANY TO ESTIMATE;

/********  製品マスター更新権  ***************/
GRANT SELECT, UPDATE, INSERT ON PRODUCTS TO ESTIMATE;

/********  見積書マスター更新権  **************/
GRANT SELECT, UPDATE, INSERT ON ESTIMATE_MASTER TO ESTIMATE;

/********  ログファイル更新権  ****************/
GRANT SELECT, UPDATE, INSERT ON LOGS TO ESTIMATE;

/********  CHECK_LOGINプロシージャの実行権 ****/
GRANT EXECUTE ON PROCEDURE CHECK_LOGIN TO ESTIMATE;

/**** 特権設定のコミット  ****/
COMMIT;
```

▼リスト2　特権の削除

```
REVOKE INSERT ON COMPANY FROM estimate;
```

▼リスト3　実行権の削除

```
REVOKE EXECUTE ON PROCEDURE CHECK_LOGIN FROM estimate;
```

はほとんどGRANT文と同じです。たとえば、会社マスターのINSERT特権をESTIMATEユーザーから外す場合は**リスト2**のように行います。

同様に、CKECK_LOGINプロシージャの実行権を外す場合は**リスト3**のように行います。

このように、GRANTとREVOKEをうまく使用していくことで、スキーマの特権を管理していくことができます。

より細かいセキュリティ管理を行う場合

InterBase自身でも非常に細かいユーザー管理やセキュリティ管理を実現することができますが、システムの各画面ごとや、機能ごとに権限を追加することは非常に難しいのが現状です。前回、簡単に説明したように、細かいセキュリティについては独自に作り込むのが理想です。どのような方法があるのかを少し説明してみましょう。

販売管理などでは、権限がないユーザーにはメニューを表示させないようにします。この場合は、メニューと権限に対応したテーブルを作成します。ユーザーがログインしたタイミングで、ユーザーと権限の対応テーブルと、メニューと権限の対応テーブルのリンクを取り、ログインしたユーザーがアクセスできる機能を絞り込んでメニューを動的に変更します。このような仕組みを作成することで、各メニューから呼び出される機能にはセキュリティを考慮する必要がなくなり、権限を追加する管理メニューだけを用意しておくだけですみます。

もっと細かく権限を管理したい場合は、さらに機能と権限の対応テーブルを作成し、どの機能を使用するのにどの権限が必要なのかを管理します。機能と権限を結びつけることで、メニューからある機能を呼び出したときにユーザーの持っている権限と比較することで、それらの機能を呼び出すことができるかを判断することができます。メニューは全ユーザー共通とし、ユーザーの権限によって機能を呼び出せるかどうかを決定していきたい場合には、この機能と権限の対応テーブルを作成する方法がよいでしょう。

どちらの方法であっても、どのようなセキュリティを行いたいのかをシステムの基本設計のタイミングでまとめておく必要があります。また、ユーザーのログイン回数を管理しておくなどのシステムが必要な場合は、どのテーブルにログイン回数を格納しておくかを決めておく必要があります。システムの基本仕様を決定し、スキーマを作成するタイミングで、将来も含めたシステムのセキュリティについて検討しておき、できるだけ将来拡張が可能なようにスキーマを設計しておくことが大切です。システムができあがってからスキーマを変更することは非常に大変です。将来行う可能性がある部分については、改良が可能なように考えておくとよいでしょう。

イベントやトリガーをうまく活用することも忘れずに覚えておいてください。

独自のセキュリティシステムを作成する方法は、ほとんどのシステムを作成する場面で直面する問題となります。ある程度のパターンを持っておくことで、それらのルーチン、または仕組みを汎用的に使用していくことも可能だと思いますので検討されてみるといいでしょう。

●

次回はIBPerlの使い方を説明し、Perlからの呼び出し方法を説明します。

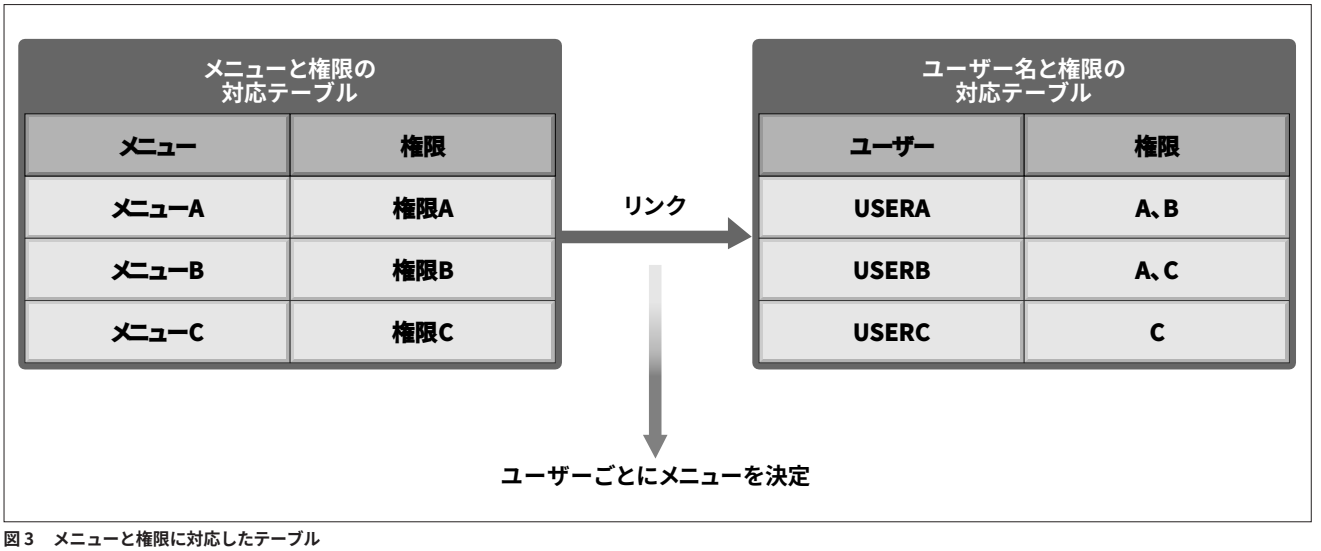


図3　メニューと権限に対応したテーブル

プログラミング工房

今月は、等距離暗号文字列検索を題材とした文字列検索プログラムの第3回目である。先月までに検索をする部分ができ上がったので、それを表示する部分とランダムテキスト生成について解説して、このプログラムを完成させてみたい。

第17回　文字列処理プログラム（3）

文：藤沢敏喜
Text : Toshiki Fujisawa

1次元配列の2次元空間への表示

コンピュータのメモリは、ハードウェア的に見れば単なる1次元の配列である。そのため、ファイルに格納されたテキスト文書も1次元のメモリにロードされ、1次元の文字列配置を2次元に表示するプログラムが必要になる。

ここで、2次元配列の表示を行うプログラムを作成するために、先月号で使った次の1次元文字列を考えてみる。

```
ALLAISTUDYISNULLCANSELSENDUSABYUNIX
```

この文字列が、char buf[35];という文字配列に格納されているとして、これを先月号で示した横（col）7文字、縦（row）5文字の2次元配列として表示するには、次のようなプログラムを作成すればよい。

```
#define MAX_COL  7
#define MAX_ROW  5

int row, col;

for(row = 0; row < MAX_ROW; row++){
    for(col = 0; col < MAX_COL; col++){
        putchar( ' ' );
        putchar( buf[ row * MAX_COL + col ] );
        putchar( ' ' );
    }
    putchar( '\n' );
    putchar( '\n' );
}
```

このプログラムを実行すると、

```
 A  L  L  A  I  S  T
 U  D  Y  I  S  N  U
 L  L  C  A  N  S  E
 L  S  E  N  D  U  S
 A  B  Y  U  N  I  X
```

という結果が得られる。しかし、このままではどこに暗号が隠されているかがよくわからないので、次のように検索した文字列をわかりやすいように表示する必要がある。

```
 A  L (L) A [I] S  T
 U  D  Y {I} S  N  U
 L  L [C] A (N) S  E
 L [S] E  N  D (U) S
[A] B  Y  U  N  I (X)
```

この表示を行うプログラムを作成するうえで、検索して文字にマークを付けた結果が、次のようにleft[35]とright[35]という文字配列に格納されているものと仮定しよう。

```
left[35]  = "  ( [     {     [ (   [   ( [     (";
right[35] = "  ) ]     }     ] )   ]   ) ]     )";
```

これを、先ほどの文字と一緒に並べると、次のような位置関係になる。

```
"ALLAISTUDYISNULLCANSELSENDUSABYUNIX"
"  ( [     {     [ (   [   ( [     ("
"  ) ]     }     ] )   ]   ) ]     )"
"--L-------I-------N-------U-------X"
"----I-----I-----C-----S-----A------"
```

ここまで準備が整えば、あとは次のようなプログラムを書けばよいことになる。

```
for(row = 0; row < MAX_ROW; row++){
    for(col = 0; col < MAX_COL; col++){
        if( left[row*MAX_COL+col] == ' ' ){
            putchar( ' ' );
            putchar( buf[row*MAX_COL+col] );
            putchar( ' ' );
        }else{
            putchar( left[ row*MAX_COL+col] );
            putchar( buf[  row*MAX_COL+col] );
            putchar( right[row*MAX_COL+col] );
        }
    }
    putchar( '\n' );
    putchar( '\n' );
}
```

これを実行することにより、先ほど示した暗号をかっこでマークしたわかりやすい表が得られる。

なお、ここでは説明の関係上、文字と文字の間隔を2文字としているが、xbibleでは横幅が少ないターミナル上でも多数の文字を表示可能なように、文字と文字との間隔を1文字としている。このため、実際のコードでは、上記のコードよりも複雑な場合分けが必要になるが、概念自体は上記で示したとおりである。

マークアップ処理

さて、次は先ほど説明したright[35]というマーク用の配列を作成する方法であるが、実際にマークを行う前に考えなければならない問題がある。それは、1つの表の中にあまりにも多くの単語が発見された場合、表がマークだらけになって見にくくなることである。

マークだらけになった表に意味があるかどうかの検討はコラムに譲るが、プログラムの機能としては、マークする単語に制限をかけて見やすくしたい。この場合、どの単語を優先して表示するかが問題になるが、今回は表の中でスキップ距離が小さい順に表示するものとした。ここでいうスキップ距離とは、単語の1番目の文字と2番目の文字を、表に2次元で配置したときの文字間の距離の2乗である。

もっと簡単に言えば、単語を構成する文字が緊密に集まっている順に表示し、文字がばらばらに離れている単語は、表示しないようにする。なお、何個までの単語を表示するかは、コマンドラインオプションの「-m」で指定ができる。

さて、この機能を実現するためには、検索した単語のすべての距離スキップを計算し、それをソートして小さい順に取り出すのが簡単である。このソートを行うためには、次の構造体を使用している。

```
typedef struct {
    int   dst;
    int   pos;
    int   len;
    int   skip;
} mark_list_t;
```

この構造体メンバのdstが前述のスキップ距離であり、posが表における位置、そしてlenとskipが文字の長さとスキップ数である。

この構造体を要素とする配列を作成し、その配列をソートすることにより、単語の表示を制限し、表がマークだらけになることを避けることができる（ソートについては**「特別講座 ステップアップC言語」**参照）。

ここまでで表示すべき個数分の構造体情報が得られるので、あとはそれを使ってマークを行うことになる。**リスト1**がそのプログラムであるが、最初のqsort関数において

第1引数は前述の構造体配列である。この配列をソートしたあとに、「-m」で指定されたopt_mにより、find_count変数を制限し、forループによってexec_mark関数を呼び出している。

このexec_mark関数は、**リスト2**に示すように最初にマークすべき文字（かっこ）を、mark_charに取得している（文字列が縦に並ぶ場合は、'|'を使用する）。

そして、for（i=0; i<word_len; i++）というループで、文字列の長さ分だけマークを行うが、すでにマークが行われたかどうかを、次のif文によって判断している。

mark配列にはmark_char、つまりかっこ文字を代入することによりマークを行っているが、numb配列にはこの位置が単語の何番目であるかの数字を入れている。これは「-m」オプションを指定したときの表示で使用される。

プリントする表の大きさと内部表現

ところで、ここで扱う等距離文字列には数千スキップ（数千文字おきに改行）以上になるものがある。しかしながら、表示装置の横幅には制限があるので、そのウィンドウに応じた幅で切り取って表示する必要がある。この幅は、多くの場合コマンドラインオプションの「-w」で設定された値である。ところが、たとえば-w80を指定した場合で、スキップ数が10であるときは、横10文字で改行する必要がある。また、「-l」オプションで行分割数の指定をした場合は、通常の場合より改行を減らす必要がある。

そして、検索した単語は表の中央になるように表示させたいので、左上の位置が全文書の中でどの位置になるかを演算する必要もある。

このように表の大きさとその中の単語の配置にはさまざまなケースがあるので、それぞれの場合に応じて次のような値を求める必要がある。

- **X軸の最大文字数（max_x）**
- **Y軸の最大文字数（max_y）**
- **表の左から検索した単語までの横マージン（mar_x）**
- **表の上から検索した単語までの縦マージン（mar_y）**
- **表における1行あたりの折り返し文字数（n_line）**

これらの数値は、検索して得られたスキップ数（skip）と、検索された単語の長さ（len）、そして「-w」、「-l」オプションで指定された値より計算可能であり、次の関数によって演算される。

```
window_value_setup(skip, len,
    &max_x, &max_y, &mar_x, &mar_y, &n_line);
```

▼リスト1　ソートとマークアップ

```
qsort(list, find_count, sizeof(mark_list_t), sort_cmp );

find_count = ( opt_m < find_count ) ? opt_m : find_count;

for(n=0; n<find_count; n++){
    exec_mark( mark, numb, n,
        list[n].pos, list[n].skip, list[n].len, max_x );
}
return find_count;
```

▼リスト2　マークアップ処理の中身

```
void
exec_mark(char *mark,char *numb,int n,int pos,int skip,int word_len,int max_x)
{
    char    mark_char;
    int     i;

    mark_char = (skip==max_x) ? '|' : mark_char_get();

    for(i=0; i<word_len; i++){
        if( mark[pos+i*skip] == ' ' ){
            mark[pos+i*skip] = mark_char;
            numb[pos+i*skip] = ( n < 10 ) ? n + '1' : '*';
        }
    }
}
```

さて、今回のプログラムは、コマンドラインで最初に指定された単語をある大きさの表の中心に配置し、その表の中で2番目以降に指定された単語をマークする。この2番目以降の単語を探すために、いちいち聖書テキストから探していては、演算が面倒である。そこで、最終的に表示する部分だけをバッファに取り出してから検索するようにすると、理解しやすく、高速なプログラムが作成できる。

具体的には、表のサイズ分のバッファを用意し、その先頭アドレスをtextという文字へのポインタ変数に代入し、

```
for(y=0; y<max_y; y++){
    for(x=0; x<max_x ; x++){
        cur = n+(y*n_line+x)-(mar_y*n_line+mar_x);
       *text++ = (0<=cur &&cur<BIBLE_LEN)?bible
[cur]:'_';
    }
```

▼リスト3　countup.pl

```
for(;;){
    if( ($c = getc) eq "" ){
        last;
    }
    $count{$c}++;
}

foreach $c ( sort keys %count ) {
    printf("\t{ '$c', %5d }, /* %2d */\n",
                        $count{$c}, ++$i );
}
```

▼リスト4　聖書におけるヘブライ文字の出現頻度

```
{ 'S',  1833 }, /*  1 */
{ 'T',  1804 }, /*  2 */
{ 'a', 27059 }, /*  3 */
{ 'b', 16345 }, /*  4 */
{ 'c',  3962 }, /*  5 */
{ 'd',  7032 }, /*  6 */
{ 'g',  2109 }, /*  7 */
{ 'h', 28056 }, /*  8 */
{ 'k', 11968 }, /*  9 */
{ 'l', 21570 }, /* 10 */
{ 'm', 25090 }, /* 11 */
{ 'n', 14126 }, /* 12 */
{ 'o', 11250 }, /* 13 */
{ 'p',  4805 }, /* 14 */
{ 'q',  4695 }, /* 15 */
{ 'r', 18125 }, /* 16 */
{ 's', 15595 }, /* 17 */
{ 't', 17950 }, /* 18 */
{ 'v', 30513 }, /* 19 */
{ 'x',  7189 }, /* 20 */
{ 'y', 31531 }, /* 21 */
{ 'z',  2198 }, /* 22 */
```

}

というようなプログラムを用いている。つまりtextが示すメモリへコピーしてから、2番目以降の単語の検索を行っているのである。なお、上記のnは、

検索した位置ー（横幅＋縦幅×n_line）

という値である。また、聖書の領域外には‘ _ ’を代入している。この結果として、textで示されるバッファに表示すべき文字が格納されるので、あとは先月解説した検索方法により与えられた単語を検索し、先ほど述べたようにマークすることにより、表示を行うことができる。

ランダムモード

さて、ここで今回のプログラムを作成する目的であった、暗号の真偽を検証する方法について考えてみる。聖書にある暗号が単なる偶然なのか、そうでないかを判断する方法として、聖書と同じような文字出現分布を持つ完全にランダムなテキストを作成して比較することが考えられる。

もし、聖書の代わりに完全にランダムなテキストを使っても、同じような暗号表が現れるならば、聖書における等距離暗号というものは、まったくのデタラメであると結論づけることができるはずだ。

■文字出現分布の調査

ランダムなテキストを作成するために、まず聖書におけるヘブライ語の出現頻度を調査してみる。このために**リスト3**に示すPerlスクリプトcountup.plを用意し、このスクリプトの標準入力に、聖書のデータbible.txtを次のように

▼リスト5　lrand48関数のインクルード

```
#ifndef __P
#define __P(x)          x
#endif

#define srand48(x)      bsd_srand48(x)
#define lrand48(x)      bsd_lrand48(x)
#define _rand48_seed    bsd__rand48_seed
#define _rand48_mult    bsd__rand48_mult
#define _rand48_add     bsd__rand48_add
#define _dorand48       bsd__dorand48

/* FreeBSD-3.3 <src/lib/libc/gen/*.c> */
#include "lib_FreeBSD/_rand48.c"
#include "lib_FreeBSD/srand48.c"
#include "lib_FreeBSD/lrand48.c"
```

与える。

```
$ perl countup.pl < bible.txt
```

このコマンドを実行すると、**リスト4**のような結果が得られ、全聖書の304805文字中に各文字がそれぞれ何文字出現したかがわかる。

■ランダムな数字の生成

多くの処理系では、乱数を発生させる関数が用意されている。たとえば、lrand48という関数は、呼ばれるたびに0から0x7fffffffまでのランダムな数を返す。なお、このlrand48関数は、srand48という関数の引数に乱数の「種」を与えて初期化することにより、その種に応じた乱数列を生成することができる。今回は何回実行しても同じ乱数列が生成されるようにしたいので、xbibleの初期化時に、

```
srand48(0x12345678);
```

を呼び出している。しかし、このようにしてもすべての処理系でまったく同じ暗号表が得られるわけではない。たとえば、Windows用クロスコンパイルmingw32のライブラリにはlrand48自体が存在しない。

そこで、ライブラリに依存しない独自のlrand48を作成することにした。具体的には、FreeBSDのlibcから、lrand48を定義している4つのファイルを抜き出し、lib_FreeBSDディレクトリに配置した。そして、**リスト5**に示すようなマクロを定義することにより、bsd_srand48()、bsd_lrand48()という関数を使用できるようにして、OSのライブラリに依存しないように構成したのである。

Column

Linux用アプリケーション「bitana」

今回の検索プログラムを書く前に、すでに開発されているものはないかといろいろと探したのだが、3月号で紹介した商用のWindows版バイブルデコーダだけしか見つけることができなかった。ところが、筆者の探し方が足りなかっただけで、すでにすばらしいアプリケーションが開発されていることを最近発見した。

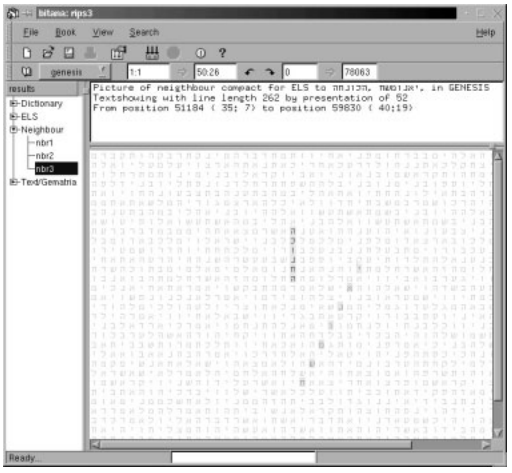
このプログラムはbitanaという名称で、Linux上のKDEアプリケーションである。bitanaはQtと呼ばれるツールキットを使って書かれたGUIアプリケーションであり、かなり気合いが入ったソフトウェアである。また、このソフトウェアのソースコードは、http://bitana.de/においてGPLライセンスに従って配布されている。ちなみに、このページには、**画面1**や**画面2**に示すようなスクリーンショットや、ドキュメント（ドイツ語）などもある。

早速ダウンロードしてきてFreeBSDやRed Hat Linux 7.0Jでコンパイルしたのだが、うまくコンパイルすることができなかった。

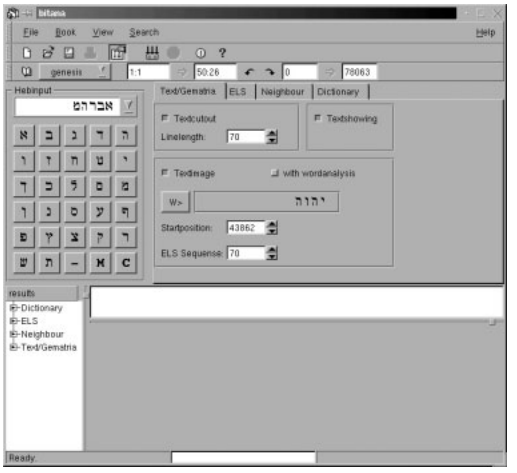
そこで、Red Hat 6.2系の数種類のディストリビューションで試したところ、コンパイルは通るものの、すぐにコアダンプしてしまった。デバッグオプションを付けて再コンパイルし、デバッガでバックトレースをしてみたところ、bitanaの外の深い場所で落ちていたので、原因追求に時間がかかりそうなことがわかった。

仕方がないので、bitanaの開発者が使用しているのと同じディストリビューションであるSuSE 6.2のCD-ROMを入手してOSのインストールをしたところ、その環境ではうまく動作させることができた（ただし、筆者の環境が特殊なためか、SuSEのインストール時に、別パーティションのOSが吹き飛んでしまった）。

実際に操作してみたが、マニュアルがドイツ語しかないこともあり、筆者には操作系がちょっとわかりにくく感じた。まだあまり操作を把握していない状態であるが、プログラムは非常に大規模なものであり、さまざまな機能があるようである。



画面1　KDE対応のバイブルデコーダ「bitana」



画面2　bitanaで表示される検索結果

このようにすると、どんな処理系でもまったく同じ乱数列が得られることになる。

■ランダムなテキストの生成

さて、聖書と同じ文字分布を持つランダムなテキストの生成方法であるが、まず0から304804（聖書の全文字数ー1）までの一様な数字を発生させる。これには、

```
( (304805-1) * bsd_lrand48() )  / 0x7fffffff
```

という演算を行えばよい。bsd_lrand48（）は、0から0x7fffffffまでの数を生成するので、上記の式では0から304804までの数が計算されることになる。

あとは、この数に従って、先ほど**リスト4**に示した文字出現頻度表を参照して、各文字を生成すればよいことになる。たとえば、0から1832の数字が生成された場合は「S」に割り当て、1833から3636（1833＋1803）までは「T」に割り当てることになる。詳細を知りたい場合は、ソースコードのsetup_random関数を参照するとよいだろう。

このようにして、聖書と同じ文字出現頻度を持つランダムテキストが得られる。

検索の高速化

ランダムモードを実装して、さまざまな文字列を比較検索してみると、検索速度が遅くて耐えられなくなってしまった。そこで、先月号の終わりに書いたように、あらかじめ検索しておいた文字列データベースを参照して高速化するという改良を行うことにした。

この方法は、たとえば、“ abcd ”という文字を検索する場合、最初に全文書の中から、単語の先頭にある“ a ”という文字列が全文書の中のどこに存在するかをすべて調べあげ、その位置をindex_fwdという文字列へのポインタ配列に記憶しておくという方法である。この考え方を導入することにより、先月号では、

```
#define   LEN 304805
char  top[LEN] = { 聖書の全文; }
char *buff;
int   s;

for(s=1; s<LEN; s++){
    for(buff=top; buff<top+LEN; buff++){
        if( compare(buff, s, word) ){
            printf("found !\n");
        }
    }
}
```

となっていた部分を、

```
char *index_fwd[xxxx];

for(s=1; s<LEN; s++){
    while( *index_fwd ){
        if( compare(*index_fwd++, s, word) ){
            printf("found !\n");
        }
    }
}
```

として、劇的な高速化が期待できる。

先月号に収録したバージョン1.20では、halcpnalhymamtという文字列を全文書から検索するのに、Pentium Ⅲ 650MHzのPCで11分42秒かかっていたが、この変更をした今月号のバージョン1.24では検索時間が16.6秒となり、42倍の高速化を実現することができた。

おわりに

等距離文字暗号プログラミングを題材として、コマンドラインオプションの扱い、文字列検索、ソート、そして乱数の発生など、いくつかのプログラミング手法についての解説を行った。これらはいずれも、プログラミングにおいては初歩的な概念ではあるものの、これらを理解することにより十分実用的なプログラムを作成できるようになる。

なお、聖書の暗号は、UFO目撃談とは違い、数学的な検証が可能な現象である。ところがその現象を明確に肯定したり、論理的に否定したりすることは、高度な確率の知識が要求され、意外に難しい。そのため、Windows版バイブルデコーダのように、ソースが公開されていないため、自分で改造できないプログラムを不自由なまま使用して、視野が狭い結論を導くこともありがちである。しかしながら、プログラミングが自由に行えるならば、その解析はより深いレベルまで可能になる。

WindowsなどのOSでは、プログラム開発環境を整える

ことは費用の面などから難しいことが多いが、フリーのOSではインストールしてすぐにCコンパイラやPerlなどが使用できる。今回のような暗号解析ツールを作成することはたいへん簡単であるので、ぜひ挑戦してみてほしい。

Column

暗号の真偽

現代のさまざまな事件が聖書に暗号化されているとされ、3月号で紹介した『神の暗号』の151ページでは、ヒ素入りカレー殺人事件に関する暗号表が掲載されている。この暗号表は、xbibleに「-w76x27 -o+3,0 -l1:1 mhayasy rol tsnh qry rcx」というオプションを与えると、画面3のように表示される。ここで与えた5つの文字列は事件に関係する単語であり、その意味は表1に示すとおりである。

『神の暗号』に掲載されている暗号表では5つの単語が、それぞれ1つだけしかないように印刷されているが、画面3を見れば明らかなように、表の中にはたくさんの組み合わせが存在する。毒薬を示す単語であるrolは、この表の中に38の組み合わせで出現し、カレーに相当するqryのほうは40もの組み合わせで表れる。これは珍しいことではなく、たとえば、「xbible -c9999 rol qry」を実行すれば無数の表が表示され、「毒薬」や「カレー」を意味する単語はいたるところに存在することがすぐにわかる。

また、聖書の暗号をとりあげている多くの本では、事件が起こる年の表記法としてユダヤ歴を用いている。**画面3**においても、tsnhがユダヤ歴5758年（西暦1998年）を示すとしている。ギリシャ文字と同じように、ヘブライ文字もそれぞれの文字に数字が割り当てられ、tが400、sが300、nが50、hが8に相当する。このためtsnhは400＋300＋50＋8で758を意味している。西暦の場合1998年を98年と略すが、ユダヤ歴でも同様な省略が行われるため、758はユダヤ歴5758年を示すというわけである。しかし、xbibleを使って実験すればすぐにわかるように、4文字（数字3桁）の年号は無数に存在し、特定の表において適当な年を見つけることは難しいことではない。

上で述べたように、3文字や4文字の単語の存在は決して珍しいことではないが、上記の5つの単語が1つの表に「同時に」存在することは珍しいことなのであろうか？

これを検証するため、次のように「-g」オプションを指定して全文書の中に、上記5つの単語が現れる表がいくつあるかをwcコマンドで数えてみよう。

```
xbible -g -w76x27 -o+3,0 -l1:1 \
        mhayasy rol tsnh qry rcx | wc -l
```

この結果、76×27というウィンドウサイズで、これらの5単語が同時に存在する表は、206個あることがわかり、**画面3**の表自体が珍しくないことがわかる。

また、上記にランダムモードを指定する「-r」オプションを追加して実行すると、その結果は190個となる。つまり、聖書の代わりに完全にランダムな文書を用いても、「毒薬、1998年、カレー、殺人」という単語が同時に存在する表は簡単にたくさん見つかるということである。

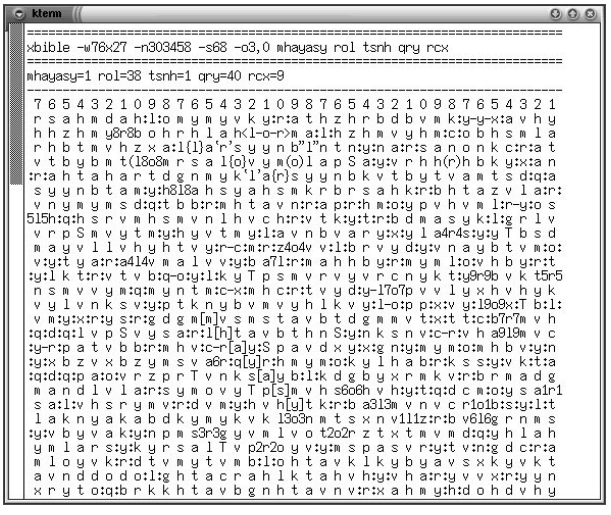
なお、『聖書の暗号』という書籍の中でも、3文字程度の単語を使用した表が掲載されているが、xbibleを使ってちょっと実験すれば、3文字程度の短い単語の存在は何も意味を持たないことはすぐにわかる。『聖書の暗号』については、次のような批判もある。

「彼の本は聖書の暗号をもっとも不幸な見地から紹介したもので、日常のことのように暗号を使って未来を予言できるかのように述べている。まじめな研究の信用を、議論の余地もなく失墜させる可能性のあるものだ」

※『聖書のミステリー』359ページより引用

また、今回xbibleを作成していろいろと実験した筆者自身の経験によれば、「8文字未満の短い単語だけを用い、単純にそれらしい表を示すことだけによって暗号が存在すると主張すること」は誤りであると感じている。

しかしながら、『聖書のミステリー』で示されている、halcpnalhymamt（神は暗号化した、神は真実である）のような14文字もの長い文字列の存在が、単なる偶然であることを証明するには、もっと別の面からのアプローチが必要なようである。また、『統計科学』という学会誌に発表された論文については、さすがに高度な証明が展開されていて、一読しただけで否定することは難しい。そのうち暇があれば、この論文を検証するプログラムを自分自身で作成してみたいと思っている。



画面3　「xbible -w76x27 -o+3,0 -l1:1 mhayasy rol tsnh qry rcx」の実行結果

単語	意味	存在数
[mhayasy]	M.林	1
{rol}	毒薬	38
(tsnh)	ユダヤ歴5758年	1
‘ qry ’	カレー	40
“ rcx ”	殺人	9

表1　事件に関係する文字列の意味

特別講座

ステップアップC言語

ソート

■ソートのアルゴリズム

ソートは、さまざまなプログラムで用いられ、計算機関連の授業では必ずとりあげられるたいへん重要な分野である。

ソートを行うアルゴリズムは、速度やメモリ使用量などのさまざまな観点から、いくつかの方法が考案されている。そのなかでも、ソートする対象があまり大きくない場合には、クイックソートと呼ばれるアルゴリズムがよく使われる。

クイックソートは、名前のとおり高速であることが特徴であり、そのアルゴリズムはたいへん興味深い。一度詳しく勉強してみて、実際にソートをするプログラムを書いてみるとたいへん参考になる。

■qsort関数

ソートのアルゴリズムの勉強はどうでもよいから、とにかくソートをしたいという場合には、C言語のライブラリに標準装備されている、qsort関数を用いるのが手っ取り早い。この関数は次のような引数と戻り値をとる。

```
void
qsort(
  void    *base,
  size_t   nmemb,
  size_t   size,
  int     (*compare)(const void *,
                     const void * )
)
```

ここで、baseはソートしようとする配列の先頭を示すポインタであり、nmembはその配列の個数である。

また、sizeは配列の要素が何バイトであるかを示し、compareは比較を行うために使用する関数である。このcompare関数には、上記の配列の要素へのポインタが渡されることになる。

この関数は、渡された2つの引数が等しいときに0を返すようにし、第2引数よりも第1引数が大きい場合は正の数を、逆の場合は負の数を返すようにする。

■ソートの例

ソートの例として、人の名前と、その人の身長と体重を示す次のような構造体を考えてみる。

```
typedef struct {
        char   *name;
        int     height;
        int     weight;
} info;
```

この構造体を要素とする5つの配列は、たとえば次のように宣言することができる。

```
#define   ARRAY_MAX 4
int array[ARRAY_MAX] = {
        { "sato",   170, 60 },
        { "suzuki", 180, 70 },
        { "tanaka", 160, 50 },
        { "yamada", 175, 65 },
};
```

この配列を身長でソートするには、比較するために使用する関数を次のように定義すればよい。

```
int
cmp_h(const void *pa, const void *pb)
{
  int a = ((info *)ptr_a)->height;
  int b = ((info *)ptr_b)->height;

  if( a == b ){
        return 0;
  }
  if( a > b ){
        return 1;
  }else{
        return 0;
  }
}
```

このように定義したcmp_hを用いると、次のようにqsort関数を呼び出すことができる。

```
qsort( array,
       MAX_ARRAY,
       sizeof(info),
       cmp_h
);
```

この結果、arrayの中身は身長の小さい順にソートされることになる。

■文字列のソート

一方、上記構造体を名前の順に並び替えるには、次のような比較関数を用意すればよいことになる。

```
int cmp_n(
  const void *ptr_a,
  const void *ptr_b )
{
  char *a = ((info *)ptr_a)->name;
  char *b = ((info *)ptr_b)->name;

  return strcmp(a,b);
}
```

ちなみに、strcmpは標準ライブラリにある関数である。この関数は、第1引数と第2引数が一致する場合には0を返し、第1引数より第2引数が辞書の配列より前にあれば正を、その逆であれば負を返す。

strcmp関数の仕様は、qsortの第4引数の関数と同じなので、もし比較する対象が文字列へのポインタ配列であればqsortの第4引数としてstrcmpを渡せばよいことになる。

Perlスクリプト入門
すぐにできる実践Perl

今回はPerlの最大の特徴ともいえる、正規表現について解説します。正規表現を使うことで、あいまいな検索や、複雑な検索ができるのです。これはPerl以外のテキスト処理系のスクリプトや、エディタソフトなどでも使われる重要な表現方法です。

第3回　正規表現

文：おもてじゅんいち／かざぐるま
Text : Junich Omote/Kazaguruma

Perlがテキストデータの処理に強いという最大の理由は、この正規表現が使えることにあります。正規表現というと、なんだか難しそうに感じるかもしれませんが、要は、文字の検索の仕方だと思ってもらえばいいでしょう。ただ、単なる決まった文字や語句の検索だけでなく、結構あいまいに、しかもその行のどのあたりにあるか（行頭、行末）などを指定することができます。たとえば、

- 「apple」か「リンゴ」か「林檎」のどれかを探す
- 0から9までの数字のどれかを探す
- <title>と</title>で挟まれた部分を探す
  （挟まれている間の文字はなんでもよい）
- 行頭に%があって、￥の含まれない行を探す
- 数字が4桁並んでいる部分を探す

など、挙げればきりがないのですが、しいて言えばどんな状態の文字でも探すことができる、万能の検索機能だと思えばよいでしょう。

ただ、この検索機能を使いこなすためには特定の記号や約束事を覚える必要があり、最初はわけがわからなくなるかもしれません。しかし、正規表現はいわば共通の取り決めなので、Perlに限らずほかのプログラム言語や、正規表現の使えるエディタソフトなら、ほとんど同じように使えます。一度覚えれば、今後いろいろなところで重宝することでしょう。

また正規表現には、2通りの使い方があります。文字の検索と置換です。検索はその行に、あるパターンの文字（列）があるかどうかを判断するもので、Perlではこれをパターンマッチと呼びます。置換については後述します。

なお、今回の説明では理解しやすいよう、例文に日本語を使用していますが、実際に正規表現で日本語を使用するには、jperlを使う必要があります。

パターンマッチ

Perlでパターンマッチを行うには、以下のような書式を使います。今、$aに調べたい行や文字列が格納されているとして、

```
$a =~ /……/
```

のように、/と/の間に正規表現でパターンを指定します。もし$aの中に、そのパターンと一致（マッチ）する部分があればこの式は真を返します。そうでなければ偽になります。ですから、パターンマッチの式は、if文などの条件式として使います。

```
if ( $a =~ /……/ ) { …… }
```

では、パターンマッチの例を見てみましょう。

$aに「今日は12月5日です。」という文字列が格納されている場合、

```
$a =~ /今日/          真
$a =~ /明日/          偽
$a =~ /12/            真
$a =~ /123/           偽
```

となります。つまり、/ /の中の内容が、$aの中にあれば真になるというわけです。

次はもう少しあいまいなパターンの場合の例を見てみましょう。

①「～月～日」という表現があるかどうか

```
$a =~ /月.＊日/                    結果は真
```

②「年～月～日」という表現があるかどうか

```
$a =~ /年.＊月.＊日/                結果は偽
```

③最後が「。」で終わっているかどうか

```
$a =~ /。$/                        結果は真
```

④数字が含まれているかどうか

```
$a =~ /[0－9]/                     結果は真
```

⑤「今日」か「明日」が含まれているか

```
$a =~ /(今日|明日)/                結果は真
```

⑥漢字とひらがな以外が含まれているかどうか

```
$a =~ /[^ぁ－ん亜－熙]/             結果は真
```

どうです、雰囲気はつかめましたか。パターンに現れる各種記号については、これから学習します。

次に、1つの検索パターンに対して、次々と調べられる文字列のほうが変わっていくとどうなるかを考えてみましょう。

前回学んだように、1つのテキストファイルを処理するときは、1行ごとに読みこんで、whileループで各行の処理を行うというのが通常のやり方でした。ですから、そのwhileループの中で、1行ごとにパターンマッチを行ってやれば、1ファイルの全行に対して、マッチするかしないかを調べることができるのです。たとえば、test1.txtというファイルの各行に対して、数字が含まれているかどうかを調べて、数字を含んだ行だけをprintするには、

```
open(IN,"test1.txt");
while( $a = <IN> ) {
   if ( $a =~ /[0－9]/ ) { print $a; }
}
close(IN);
```

とすればよいのです。if文の($a =~ ／[0－9]/)が、それぞれの行に数字が含まれているかどうかを判定し、その結果が真であれば、{ }内のprint $a;を実行するというわけです。

百聞は一見にしかずです。早速、このスクリプトを書いて実行してみましょう。test1.txtというファイル名を好きなものに変えて、実際に何が出力されるかを見てください。また、できればパターンの部分を、前例の①～⑥に変えてみて、どうなるかを調べてみましょう。

/ /の中の記号はそのままで、文字を変えてみることにも挑戦してください。

たとえば、『①「～月～日」という表現があるかどうか』を「～時～分」にするときは、

```
$a =~ /時.＊分/
```

としてください。

■正規表現

パターンマッチを行うときに、/ /内のパターンを表現するために用いるのが正規表現です。基本的には、/ /内に書かれた文字が存在するかどうかを調べますから、

```
/今日/
```

は、一番基本的なパターンといえます。

■選択

次に、いくつかの語句のうちいずれかを含むかどうかを調べたいときは、それらの候補を｜で区切り、全体を( )で囲みます。

```
/(昨日｜今日｜明日)/
```

候補はいくつあっても構いません。

■繰り返し

調べたい文字がはっきり決まっているときは、この2つのパターンでよいのですが、途中は何でもよいという場合があります。前例の①のように、「〜月〜日」を調べたい場合、「月」と「日」以外はなんでもよいわけですから、なんでもよいという部分を、.＊（ピリオド＋アスタリスク）と表現します。すると、パターンは、

```
/月.＊日/
```

となります。

```
/.＊月.＊日/
```

になるのでは？　と考えそうですが、結果はどちらも同じです。もともと/ /内に書かれたパターンが存在するかどうかを結果として返すものなので、その前後はなんでもよいということになりますから、「月」の前の.＊は、書いても書かなくても結果は同じになります。

.＊の「.」は、「任意の文字にマッチ（何か1文字）」、「＊」は「直前の文字の0回以上の繰り返し」を意味します。ですから合わせると、「任意の文字の0回以上の繰り返し」になります。「任意の文字」とは「どんな文字でもよい」ということですから、「0文字以上のすべての文字」ということになり、つまりどんな文字が何文字あってもよいということになるのです。

たとえばここで、「月」と「日」の間の文字が1文字だけ、というパターンにしたければ、

```
/月.日/
```

になります。この場合、「月」と「日」の間に2文字以上の文字があると、結果は偽になります。

同様に、「月」と「日」の間の文字が2文字だけということであれば、

```
/月..日/
```

となります。

また、この＊は直前が . でなくても使えます。「あ」の繰り返しというパターンであれば、

```
/あ*/
```

とすれば、「あ」「ああ」「あああ」「ああああ」……など、どれでもマッチしますし、数字だけの繰り返しというパターンの場合は、

```
/[0－9]*/
```

となります。

ところがちょっと問題なのは、＊が「0回以上の繰り返し」であることなのです。1回以上はわかりますが、0回とはどういうことなのでしょう。

実は、先ほどの「/あ＊/」は、確かに「あ」という文字の繰り返しにマッチするのですが、「あ」の0回にもマッチしてしまいます。「あ」の0回というのは、「あ」が含まれていなくてもよいわけですから、結局、何にでもマッチしてしまうことになりますから、これだけでは意味がありません。

実際に「あ」の繰り返しや、数字だけの繰り返しにマッチさせるには、

```
/あ+/
/[0－9]+/
```

というように、＊の代わりに＋を使います。＋は「直前の文字の1回以上の繰り返し」を意味します。「〜月〜日」の場合は、「12月5日」などには＊、＋どちらもマッチしますが、

```
/月.＊日/
```

の場合は、「月日」という文字列にもマッチし、

```
/月.＋日/
```

では「月日」にマッチしません。

「〜月〜日」だけでなく、「月日」にもマッチさせたいときは、＊を使わなければなりませんが、通常「繰り返し」というのは、1回以上の場合のほうが多いと思われますので、まず＋を、そして特殊なときに＊を使うことを考えましょう。

■文字クラス

さて、この「数字だけ」を表している[0－9]ですが、この[ ]は、文字クラスと呼ばれるもので、[ ]内に書かれた

どれかの文字にマッチするかどうかを調べるものです。
これは、

```
/[0123456789]/
```

と書けば、[ ]内に書かれたどれかの文字、すなわち「0123456789」のどれかの文字があるかどうかを調べられます、

```
/[あいうアイウ]/
```

と書けば、「あ」「い」「う」「ア」「イ」「ウ」のどれかの文字があればマッチします。
しかし、「数字」や「ひらがな」、あるいは「アルファベット」を調べるのに、いちいち全部の文字を書くのも面倒なので、連続した文字の場合は、「－」で最初と最後をつないでやれば、その範囲の文字のどれかという意味になります。

数字	/[0－9]/
ひらがな	/[ぁ－ん]/
アルファベット	/[a－zA－Z]/

なお、[ ]－.＊など、正規表現の記号は必ず半角文字で書く必要があります。
ひらがなだけでなく、ひらがなと「、。」の場合であれば、単純にそのあとに書き足して、

```
/[ぁ－ん、。]/
```

と書けばよいだけです。
ただし、正規表現で意味を持つ、^ [ ] ( ) $. ＊＋－？\ などはメタ文字と呼ばれ、そのまま書くと、それらの持つ働きが効いてしまいますので、これらの文字そのものを調べたいときは、\ に続けて書かなければなりません。
たとえば、数字と＋－.の場合は、

```
/[0－9¥+¥－¥.]/
```

と書くことになります。\ を調べたい場合は \ \ です。

■～以外の文字

[……] を [^……] と書けば、「……以外の文字」という意味になります。
たとえば、「ひらがな以外の文字」であれば、

```
/[^ぁ－ん]/
```

ですし、「漢字とひらがな以外の文字」であれば、

```
/[^ぁ－ん亜－熙]/
```

になります。
ここで、「漢字」を表すのに、何やら見慣れない文字が並んでいますが、「亜」と「熙」は、漢字をJISコードで並べたときの最初と最後の文字なのです。このように、文字の範囲を表す「－」は、文字コード順に並べたときの最初と最後になりますから、文字コード表などを参考にして、正しく最初と最後の文字を書くようにしてください。

■行頭・行末

行頭と行末を表すには、^と$を使います。
たとえば、「今日」という文字が行頭にある行にマッチさせたければ、

```
/^今日/
```

と書き、行末に「する。」がある行なら、

```
/する。$/
```

と書きます。また、その両方を使って、

```
/^わたし$/
```

と書けば、「わたし」だけの行（前後になにもない）にマッチします。前述の文字クラス、繰り返しと合わせれば、「数字だけの行」は、

```
/^[0－9]＋$/
```

になります。もうわかりますね。

■正規表現のまとめ

ほかにも、正規表現にはさまざまな記号があるのですが、とりあえず、ここまでのものをマスターすれば実用にはな

ります。これまでに出てこなかったものも含めて、表にしておきましたので参照してください。

■パターンマッチでの$_

前回解説した$_ は、ここでもやはりデフォルト変数として登場します。通常のパターンマッチは、

```
$a =~ /[0－9]/
```

でしたが、調べたい文字列が$_に入っているときは、

```
/[0－9]/
```

だけでパターンマッチを行うことができます。つまり、$a =~などを書かなければ、Perlは自動的に、そのパターンを$_のデータに対して調べてくれるのです。

ですから、前述の数字を含んだ行だけをprintするというスクリプトは、

```
open(IN,"test1.txt");
while( <IN> ) {     # $_に1行読み込む
   if ( /[0－9]/ ) { print; }
   # $_をパターンマッチ調査して真ならprint
}
close(IN);
```

と、またもやまったく変数を使うことなく書くことができるのです。

パターンマッチの場合は、$_を使えば、スクリプトはかなりシンプルになりますから、できれば、$_を使うようにしたほうがよいでしょう（無理することはありませんが）。

正規表現による置換

先ほど学んだ正規表現は、単にパターンのあり／なしを判定するだけのものでしたが、ここでは正規表現を使った、パターンの置換を学習します。この置換は、「テキスト加工」ともいえる処理が行えるという、Perlの中でも一番強力な部分です。ただし、先に出てきたパターンマッチがすべての基本となりますので、十分復習しておいてください。

■置換の書式

置換のための書式は、

```
$a =~ s/……/……/;
```

になります。1つめの/ /の中に探したいパターンを記述し、2つめの/ /の中に、置き換え後の文字を記述します。2つめに何も書かなければ、パターンにマッチした部分が削除されます。たとえば、

```
s/株式会社/（株）/;   # 株式会社を（株）に置換
s/株式会社//;         # 株式会社を削除
s/H1//;               # H1 を削除
s/H[1－4]//;          # H1、H2、H3、H4を削除
s/[0－9]+//;          # 数字の部分を削除
s/^[0－9].*¥n//;      # 数字で始まる行を削除
```

というように、1つめには正規表現のパターンが使えます。

また、2つめに変数を書けば、変数展開されます。

```
$b = "株"
s/株式会社/$b/;     # "株"に置換される
s/株式会社/（$b）/; # （株）に置換される
```

置換に使用するパターンの両方とも、パターンマッチのときと同様に、＾［］（）$．＊＋－？ \ などのメタ文字をただの文字として扱いたいときは、正規表現の意味を持つ記号と混同されないように、必ず \ に続けて書かなければなりません。たとえば、（株）の （）を半角にしたいなら、

```
s/株式会社/¥(株¥) /;    # 半角( )の場合
```

になります。一見記述はややこしくなりますが、気をつけ

.	任意の１文字にマッチ
＊	直前の文字の0回以上の繰り返し
＋	直前の文字の1回以上の繰り返し
?	直前の文字の0回か1回にマッチ
[]	[]内のどれかの文字にマッチ
[^]	[^]内の文字以外にマッチ
(|)	いずれかの語句にマッチ
^	行頭
$	行末
\	直後の文字（^[](){}$.＊＋－？ \）を普通の文字として扱う
\n	改行
\t	タブ
\x00	16進数で表した文字コード
\d	数字（[0－9]と同じ）
\w	英数字（[_A－Za－z0－9]と同じ）
\s	空白類（スペース、タブ、改行など）

表　正規表現に使われる記号

てください。

また、ここでもデフォルト変数$_は有効で、$aを置換するときは、

```
$a =~ s/……/……/;
```

ですが、単に、

```
s/……/……/;
```

とだけ書いた場合は、デフォルト変数$_の内容が置換されます。

■後方参照

これまでは、決まった文字への置換しかできませんでしたが、テキスト加工では検索されるパターンによって、置き換え後の結果を変えたい場合が多いものです。

たとえば、金額の「5000円」を、「\5000」に変えるように、数字の部分はそのままにして、最後の「円」を削除して、先頭に「\」を付けたい場合などです。また、金額の数字部分は5000に限らず、どんな数字でも同じように、たとえば「4800円」でも「\4800」と変換したいとしたらどうしたらよいでしょうか。

こういった場合、1つめのパターンは、

```
/[0－9]＋円/
```

となります。しかし、これを\に変えるように、

```
s/[0－9]+円/¥¥/
```

とすると、結果は「5000円」全体が、「\」に変わってしまって、5000の部分がなくなってしまいます。

こういう場合、1つめパターンで、

```
/([0－9]＋)円/
```

のように、数字にあたる部分を( )で囲むと、それにマッチした実際のデータは、$1という変数に自動的に格納されるのです。そして、その$1は、2つめの置き換え部分で使えるので、

```
s/([0－9]＋)円/¥¥$1/
```

とすれば、置き換え結果はめでたく、「\5000」や「\4800」になるのです。

このような、パターン内の( )と、$1の対応を後方参照といい、パターン内のどんな部分でも、( )で囲めば、その中身は順番に、$1、$2、$3……という変数に格納されているのです。

後方参照により、置換の幅が大きく広がることになります。もはや置換というより、加工といったほうがぴったりくるでしょう。それでは、いろいろ例を見てみましょう。ややこしい記述ですが、じっくり見て理解してください。

①03－1234－5678を、03(1234)5678に変える

```
s/([0－9]＋)¥－([0－9]＋)¥－([0－9]＋)/$1¥($2¥)$3/;
```

②1230001という郵便番号を123－0001にする

```
s/([0－9][0－9][0－9])([0－9][0－9][0－9][0－9])/$1¥－$2/;
```

この場合、同じものの回数指定の繰り返しは、{n}を使って書くこともできます。

```
s/([0－9]{3}) ([0－9]{4})/$1¥－$2/;
```

また、[0－9] は、\dというメタ文字で書けるため、

```
s/(¥d{3})(¥d{4})/$1¥－$2/;
```

と書いても同じ意味になります。

以上の3種類の書き方がありますが、どれを使うかは自分の好みで構いません。しかし、できれば自分のスタイルを統一しておいたほうが、あとでプログラムを見直すときに困りません。

③CSV（カンマ区切りテキスト）ファイルのデータの並び順を変える。

```
123－4567,東京都,山田太郎,男性
```

というデータを、

```
山田太郎,男性,123－4567,東京都
```

にする。

```
s/([^,]*),([^,]*),([^,]*),([^,]*)/$3,$4,$1,$2/;
```

このように、CSVファイルの加工まででできてしまいますが、CSVファイルの加工については、次回以降でもっと良い方法を説明しますので、ここでは正規表現でここまでできるということを理解しておいてください。

■修飾子

s/……/……/では、置換は1回しか行われません。ある行（データ）に対して、該当する箇所があるだけ、繰り返し置換を行いたい場合は、

```
s/..../..../g
```

と、最後に「g」を付けます。

```
$a = "あの日、あの場所、あの歌";
```

を置換する場合、

```
$a =~ s/あの/アノ/;
```

では、

```
$a = "アノ日、あの場所、あの歌";
```

になりますが、

```
$a =~ s/あの/アノ/g;
```

であれば、

```
$a = "アノ日、アノ場所、アノ歌";
```

となります。たいていの場合は、「g」を付けておくほうがよいでしょう。

そのほかの修飾子として、「i」を最後に付ければ、パターンマッチのときに大文字と小文字を区別しません。

```
s/……/……/i;
```

または、

```
s/……/……/ig;
```

となります。

■変換演算子

置換は、パターンにマッチした部分全体を置き換えますが、時には、大文字を小文字に、半角を全角になど、それぞれの文字を違う文字種に置き換えたいときがあります。その場合は、

```
tr/……/……/;
```

を使います。大文字を小文字に変換するなら、

```
tr/ABCDEFGHIJKLMNOPQRSTUVWXYZ/abcdefghijklmnopqrstuvwxyz/;
```

です。それぞれ対応する文字が2つめの文字に置き換わります。連続している文字なら、

```
tr/A－Z/a－z/;
```

と書けます。

全角数字を半角数字にするなら、

```
tr/[０－９]/[0－9]/;
```

です。1つめの０と９は全角文字ですが、－は半角で記述しなければなりませんので注意してください。

/ /の中身は、何も文字の順になっている必要はありません。

```
tr/ACEDB/acedb/;
```

でも構いませんし、

```
tr/ABCDE/edcba/;
```

と書いても構いません。ただし後者の場合は、Aがeに、Bがdに変わります。変換は並んだ文字の順に対応して行われます。

もし前半と後半の文字の長さが違うとどうなるでしょうか。

```
tr/ABCDE/abc/;
```

この場合は、

```
tr/ABCDE/abccc/;
```

というように、最後の文字が延長されているものとして変換されます。

ですから、

```
tr/A－Z/x/;
```

とすれば、A～Zすべての文字がxに変換されます。パスワードなどの伏字を作るにはよさそうです。

ちなみに、数字を漢数字にするときは、漢数字は一、二、三……がコード順には並んでいないので、

```
tr/0－9/〇一二三四五六七八九/;
```

と書かなければならないことに注意してください。

わからない人は、それぞれの文字の文字コードを調べてみましょう。

とにかく、この正規表現と置換で、1行内の処理であればほとんどの場合のテキスト加工が可能になるはずです。パターンを作るのには苦労するかもしれませんが、いろいろな目的のテキスト加工をぜひ実践してみてください。

■クォート文字について

パターンマッチや置換の場合は、パターンを囲む記号として「/」を使っていました。そのため、パターン内に「/」という文字が含まれる場合、

```
/http:¥/¥/www/;  # http://wwwとマッチ
```

というように、\/で表さなければなりません。

「/」が頻繁に現れる場合などは、いちいち「/」の前に\を書かなければならないのは面倒なので、パターンを囲む区切り文字を変更することができます。その場合は直前にmを付けて、

```
m#……#
m@……@
m{……}
```

などと書くこともできます。ただし、使えるのは英数字以外で、カッコ類（< >、( )、[ ]、{ }）を使うときはペアにして、それ以外の場合は同じ文字のペアでなければなりません。

こうすれば、「/」はもはや特殊な文字ではなくなりますから、

```
m{http://www};
```

というパターンが書けるようになります。実は、「/」のときも、「m//」と書くのが原則なのですが、「/」に限って、「m」を省略してもよいことになっているのです。

置換の場合も同様ですが、カッコ類のときがちょっと違っていて、

```
s#……#……#
s@……@……@
s{……}{……}
```

のようになります。

同じような規則が、文字列を囲む"や'にもあり、それぞれにおいて、"や'を使いたいときなどは、

```
'あいうえお'  →  q{あいうえお}
"あいうえお"  →  qq{あいうえお}
```

と書くことができます。

さて、これでもうあなたは、Perlの大きな山をひとつ越えました。正規表現を手にしたあなたは、Perlはもちろん、ほかのいろいろなところで役立つ、プログラマーとしての技量を身に付けたことになるのです。

ここまでの知識だけでも、今まで手作業だけでは大変だった検索作業を、正規表現を使ったファイル内検索ツールを作ることによって簡略化できるはずです。今回の正規表現の解説を読んだだけでも、正規表現の可能性が理解できたと思います。住所録のフォーマットを統一したり、HTMLファイルを整形したり、文章のゆれを統一したりと、さまざまな場面で正規表現が活躍するでしょう。ぜひ、自分なりにいろいろなスクリプトを書いて、実際に使ってみてください。とにかく書いてみること。さあ、第一のゴールは目前です。

いま一度、ここまでの学習が本当に身についたかを再確認してください。

Rubyで行こう

コンピュータ関係って、なんだか特殊な用語が多いですよね。特に、英文字・数字の略語が目立ちます。筆者が学生時代バイトしていた某外資系企業では、社内で流通する英字略語のための辞書まで用意されていました。

第16回　コレクター

文：赤松智也
Text : Tomoya Akamatsu

今回は特にふだんあまり接することの少ない海外からの話題を中心に、Rubyにまつわる用語を集めてみました。この連載は、話題の連載「Linuxガベコレ」ではありませんから、内容は信じても大丈夫（のはず）です（笑）。

今月の用語
【Ruby】
【matz】
【RAA】
【RCR】
【Ruby Conference】
【RD】
【pickaxes本】
【ri】
【256倍本】
【Wiki】
【irb】
【ヤギ本】
【POLS】

Ruby【るびぃ】

いわずとしれた本連載のテーマ。オブジェクト指向スクリプト言語です。世の中にはオブジェクト指向言語は山のようにあるのですが、そのなかでもRubyは不思議な魅力を持っています。

これはなんなのでしょう。Rubyは最も単純な言語でも、最も複雑な言語でもありません。また、飛び抜けて性能が高いわけでも、ほかの言語にまったくできない何かができるわけでもありません。などと言ってしまうと実もふたもありませんが、はっきり言えば事実です。

しかし、Rubyは私を含めた多くの人々を引き付ける何かがあるというのも、また事実です。このことについて、『Programming Ruby』の著者でもあるDave Thomasが、Rubyメーリングリストruby-talkの［ruby-talk:12606］でおもしろい考察をしています。

彼によれば、RubyはChaordicなのだそうです。Chaordicというのは、カオス（Chaos）と、安定秩序（Order）の中間を現す造語（CHAos＋ORDer）で、挙動をまったく予想できないカオスでもなく、逆に変化が期待できない安定秩序でもないのがChaordicなシステムです。

Chaordicなシステムは、少々不安定であるが非常に機動的で、少ない入力で多くのことを達成できるため、生産性が高い傾向があるということです。また、カオスな世界と安定の世界をつないで、新しいものを産み出す力があるということでした。

このことが、本当にあてはまるのかどうかは、にわかには判断できませんが、Rubyの生産性や気持ちよさというのは、機能や仕様だけでは説明できないものがあるように

思います。これがChaordicということなんでしょうか？

matz【まつ】

Rubyの作者。日本では「まつもと ゆきひろ」として、ひらがなの名前で知られていますが、海外では「matz」で通っているのだそうです。たしかにガイジンには発音しにくそうな名前ですよね。

「ゆきひろ」は、ときどき「ひろゆき」に間違えられています。ま、「ひろゆき」のほうが圧倒的に多い名前ですから。筆者も、何度か間違いを目撃したことがあります。まつもとさんがおっしゃるには、「間違えられると激怒する」のだそうですから、くれぐれも間違えないように……。もっとも、まつもとさんが激怒したところってあんまり想像できないんですが。

まつもとさんが名前をひらがなで書く理由は、「ありふれた名前を特徴づけるため」なんだそうです。「それに小学校のときからひらがなで名前書いてたし」ってことでしたが、それってみんなそうなんではないでしょうか（笑）。Ruby関連ではひらがなで名前を書く人が多いんですが、それも同じような理由なんでしょうか？

RAA【あーる・えー・えー】

Ruby Application Archive。訳せば「Rubyを使ったプログラムの貯蔵庫」とでもいうのでしょうか？　さまざまな分野のRubyプログラムが登録されているWebページです（**画面1**）。RAAのURLは、

```
http://www.ruby-lang.org/en/raa.html
```

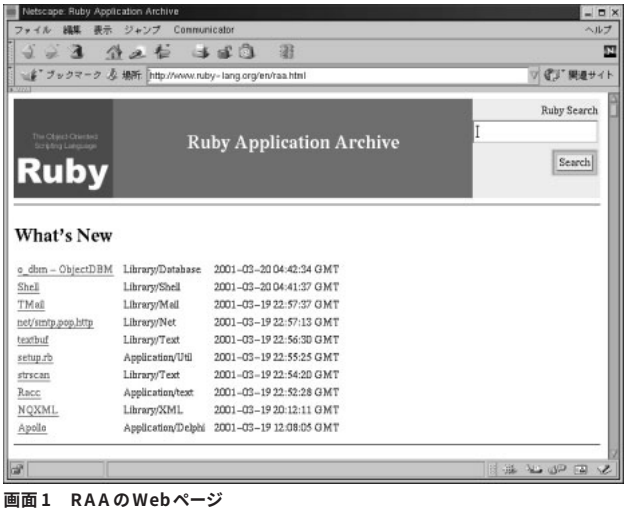
です。

RAAは大きく「プログラム」、「ライブラリ」、「ポーティング」、「ドキュメント」の4分野に別れており、原稿執筆時点では全部で264本（プログラム95本、ライブラリ154本、ポーティング5本、ドキュメント10本）が登録されていました。

登録・管理とも、eRubyを使ったCGIプログラムで完全自動化されています。PerlのCPAN（Comprehensive Perl Archiver's Network）のような自動インストール機能や検索機能はまだありませんが、将来が楽しみなアーカイブではあります。

皆さんも自作のプログラムをRAAに登録してみませんか？　英語のドキュメントが必要ですが、ソースコードは万国共通なので、中学生程度の英文でも十分でしょう。登録手順は簡単です。

1. ［add new entry］をクリックする。
   What's Newの下、“ The Ruby Application Archive ”のタイトルのすぐ下右のほうに、[add new entry]というリンクがあります。ここをクリックすることで、新しいエントリーを登録する登録フォームのページにジャンプします。
2. 登録フォーム（画面2）を埋める
   Homepageなどは、なければ空欄のままでかまいません。
3. ［register］ボタンをクリックする
   簡単でしょう？　パスフレーズは以後の更新に必要ですから、忘れないようにしてください。更新にはRAAの各エントリーにある［update］というリンクをクリックします。



画面1　RAAのWebページ

画面2　RAA登録フォーム

RCR【あーる・しー・あーる】

Ruby Change Request、訳せば「Ruby変更依頼」ですね。PerlのRFC、PythonのPEPに相当するものですが、それらに比べるとかなりラフな感じです。要するに、まつもとさんがせっかくの提案を忘れてしまわないためのメモとしての性格を持つからだと思います。

原稿執筆時点では、17個のRCRが提案されています(**表1**)。

RCRの提案はruby-talkで提案を出すと、RCRの管理者のMike Wilson氏がRCRのページ、

```
http://www.rubygarden.com/ruby?RubyChangeRequest
```

に追加してくれます。日本語で提案したい場合には……、うーん、だれかに英訳してくれるように頼むのでしょうか？　まあ、ruby-listでまつもとさんに直接頼めばそれで済むことではありますが、先ほど説明したRAAも完全に英語ですし、Ruby界の公用語は急速に英語になりつつあるのでしょうか？　国際化って不便なこともありますね。やっぱり英語を勉強しよう。

Ruby Conference【るびぃ・かんふぁれんす】

以前は「～という名前の単なる宴会」のことで、2、3回開かれているはずです。実際のカンファレンスとしては、2000年11月には京都で「Perl/Ruby Conference」(2001年2月号の本連載でレポート）が開かれています。先頭の「Perl/」を取り除いてしまえば（おいおい)、りっぱな「Ruby Conference」ですね。

Ruby単体のカンファレンスとしては、今年アメリカで初の「Ruby Conference in US」が開かれるのだそうです。時期は2001年10月12日、13日。場所はフロリダ州TampaにあるHoliday Inn Tampa City Centreホテルです。OOPSLAというオブジェクト指向関連のカンファレンスにあわせて開かれるのだそうです。まつもとさんは招待されているそうですが、私も行きたいなあ。旅費出ないかなあ（Linux magazineで特派員を出しませんか？)。

Ruby Conference in USの詳しい情報は、

```
http://www.rubyconf.org/
```

RCR	内容
#U001 cut operator for short-circuiting method chains	bangメソッドの連鎖ができるようなcutオペレータのメソッドの提案。
#U002 new proper name for Hash#indexes, Array#indexes	キーを返すindexと値を返すindexesでは変なのでもっと適切な名前を、という提案。なかなか良い名前は決まらないようです。
#U003 infix 'function composition' operator	ProcやMethodのcurry化（？）を行うメソッドの提案。
#U004 More list operators	HaskellやLispのあるようなリスト操作メソッドを追加する提案。
#U005 Discuss possibility of icon-like suspend or "inside-out yield"	Iconのようなgeneratorがほしいという提案。提案者もちょっと無理だと思ってるみたいです。
#U006 Alternate variable declaration	Perlのmyのようなものがほしいという提案。でも、ちょっとRubyには難しいような。
#U007 Pragma to disallow variable creation with just an assignment	Perlのuse strict varsに相当するものがほしいという提案（なんだと思います)。
#U008 Keep track of implicitly called methods	メソッド名の一覧を管理することでタイプミスを検出しやすくしようという提案のようです。
#U009 Allow "#{foo} bar" to be used as "#foo bar"	文字列中の#fooで#{foo}相当にする提案。便利かも。
#U010 new method Enumerable#hashify(value)	配列からハッシュを作るメソッドの提案。
#U011 Add a method: Time#set(seconds <,microseconds)	Timeオブジェクトの状態をあとから変更できるようにしようというものです。オブジェクトをmutableにすることは抵抗も大きいようです。
#U012 Discuss making Ruby XML-ready "out of the box"	XMLライブラリを標準添付しようという提案です。xmlparserライブラリはちょっと大きすぎるので、ChibiXMLとかNQXMLを添付するのでしょうか。
#U013 shortcut for instance variable initialization	引数リストにインスタンス変数を直接書くことで、初期化を簡単にする提案です。ruby-listでも以前見た提案ですね。
#U014 new method, instance_eval＋value passing	instance_evalと同様にselfをすり替えるメソッドで、ブロックにパラメータを渡せるメソッドの提案です。名前がなかなか決まらないようです。
#U015 replace 'caller' with 'call_stack'	callerの代わりにスタック情報を多く含むcall_stackメソッドの提案です。
#U016 Block form of Dir.chdir	Dir.chdir(dir) { ... } でブロックの範囲内だけディレクトリを変更するという提案です。これも以前にも見たような。
#U017 "partition" method	ブロックで指定した条件を満たす要素と満たさない要素に分割するメソッドです。名前はpartitionで決まりのようです。問題は採用するかどうか。

表1　RCRリスト

にあります。日程は**表2**を参照してください。

RD【あーる・でぃー】

Ruby Documentの略で、Rubyでよく用いられるドキュメント用フォーマット。「PODとPlain2をたして2で割った」でわかる人にはわかるフォーマットです。でも、それではわからない人にはまったくわからないので、実際の例によって説明しましょう。

RDの文章は**リスト1**のようになります。

例にもあるように「なんとなくプレーンテキスト」っぽくドキュメントが書けるという点がなかなかの優れものです。なお、このRDも、まつもとさんがオリジナルを設計したものだそうです。まさにプログラマ向けドキュメントフォーマットと言ったところでしょう。

Ruby自身のリファレンスマニュアルはながらくHTMLで記述されていましたが、バージョン1.6対応版からは、このRDで記述されるようになりました。バージョン1.6対応のリファレンスマニュアルは、

```
http://www.ruby-lang.org/~rubikitch/refm/ruby162-refm.rd.gz
```

から入手できます。また、Web上では、

```
http://www.ruby-lang.org/ja/man-1.6/
```

で参照できます。

RDはRDtoolというツールで処理します。RDtoolの本体はrd2というコマンドです。上の例文をrd2コマンドに加えると**リスト2**のような変換結果が得られます。

2001年10月12日（金）
8:00 受け付け
9:00 Codefest
12:00 昼食
1:30 キーノートスピーチ
2:45 休憩
3:15 発表
5:00 自由時間
5:30 夕食
7:00 matzを囲んで

2001年10月13日（土）
8:00 発表
9:15 休憩
9:30 ワークショップ
12:00 昼食
1:30 ワークショップ
4:00 休憩
4:15 表彰・閉会

その他
Tシャツ・書籍即売会
表彰

表2 Ruby Conference日程（予定）

▼リスト1 RD文書（例）

```
=begin

= RD文章の例

RD文章は =begin で始まり、=endで終わります。そして、
イコールで始まる行は、章・節のタイトルになります。

RDの特徴は

  * なんとなくプレーンテキストに見える
  * そのわりに構造のある文章が書ける
  * やろうと思えば、((*修飾*)) もできる

というところにあります。

=end
```

▼リスト2 RDからHTMLへの変換結果

```
<?xml version="1.0" ?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>example</title>
</head>
<body>
<h1><a name="label:0" id="label:0">RD文章の例
</a></h1><!-- RDLabel: "RD文章の例" -->
<p>
RD文章は =begin で始まり、=endで終わります。そして、
イコールで始まる行は、章・節のタイトルになります。
</p>
<p>
RDの特徴は
</p>
<ul>
<li>なんとなくプレーンテキストに見える
</li><li>そのわりに構造のある文章が書ける
</li><li>やろうと思えば、<em>修飾</em>もできる
</li></ul>
<p>
というところにあります。
</p>

</body>
</html>
```

```
$ rd2 example.rd > example.html
```

HTMLへの変換結果をブラウザで表示させると、**画面3**のようになります。

RDをテーマにしたRubyの256倍本が出版されるそうです（『Rubyを256倍使う本 魔道編』ISBN4-7561-3747-4）。本誌が書店に並ぶより前に皆さんの元に届いているのではないでしょうか。RDだけで1冊書いてしまうというのも、スゴイ話ではあります。

pickaxes本【ぴっける・ほん】

Dave ThomasとAndy Huntによる『Programming Ruby』の愛称。表紙のピッケルとRubyの原石の絵（**写真1**）からこう呼ぶ人が多いようです。ピッケルってこういうスペルだったんですねえ。現時点で地上に存在する最も網羅的なRuby関連の本でもあります。なぜ日本語ではないのか疑問なところですが、鋭意翻訳中だということです。

ところで、最近非常に話題になったのは、この本の内容全部が、オープンコンテンツとしてフリーに公開された点です。ええ、本当です。ごく一部ではなく、全部です。以前にも、たとえばオライリーの『オープンソース』という本の内容が公開されたことがあります。が、このようなきちんとした本として出版され、絶版にもなっていないものが、全部公開された例ってほかにあるんでしょうか？

アディソン・ウェスレーも大胆なことをするものです。個人的にはオープンソース関係の書籍ならオライリーという一種の偏見があったのですが、今回の件ですっかり見直しました。

画面3　RD変換結果の表示

さて、その公開されたコンテンツは、

```
http://www.rubycentral.com/book/index.html
```

からアクセスできます。残念なことに、まつもとさんの書いた序文を含めてすべて英語です。また、以下のURLからソース（XMLおよびHTML形式）をダウンロードできます。

```
http://www.pragmaticprogrammer.com/ruby/downloads/book.html
```

すごいですねえ。この本の日本語訳が出版されたらそれもオープンコンテンツにならないものでしょうか？　出版社に対して署名運動をするとか。

ri【あーる・あい】

上記のピッケル本の後半、リファレンス部分を検索可能にしたプログラムがriです。riは「Ruby InteractiveReference」の略なのだそうです。

英語なのが残念なのですが、これは非常に便利なプログラムです。たとえば「each_with_indexってどんな動作をするんだっけ」と思ったら、

```
$ ri each_with_index
```

とタイプするだけで、**画面4**のような例題付きの出力が得られます。

また、openというメソッドがどのクラスに定義されているか知りたければ、

```
$ ri open
```

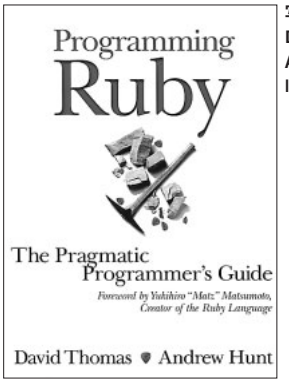


写真1　Programming Ruby
David Thomas & Andrew Hunt著
Addison-Wesley
ISBN0-201-71089-7

で、

```
The method named `open' is not unique among Ruby's
classes and modules:
        Dir::open, File::open, Kernel::open
```

とDirクラス、Fileクラス、Kernelモジュールにopenが定義されていることを教えてくれます。

それから、Dir::openについて知りたければ、**画面5**のように表示されます。至れり尽くせりです。

riはすばらしいプログラムですが、不満を述べるとすれば、以下のような点があります。

・表示が全部英語である
日本生まれのRubyの情報が英語のみというのはちょっと悲しいです。riもオープンソースですから、翻訳プロジェクトをスタートさせるべきかもしれません
・組み込みクラスしかカバーしない
よく使うSocketやNet::FTPのことはわからないのなどはちょっと悲しいです。これも追加データの編纂をプロジェクト化することができるかもしれません

いつか日本語riがほしいものです。どうです、あなたが始めてみませんか？

256倍本【にひゃくごじゅうろくばい・ほん】

アスキーが出版している『～を256倍使う本』というタイトルのシリーズの総称。タイトルは昔の『プロ野球を10倍楽しむ本』のパロディだと思われます。一時はたくさん出ていましたが、『インターネットを256倍使う本 vol.2』なんかしばらく出版されなかったような気がします。

沈黙を破って2000年10月に出版されたのが『Rubyを256倍使う本 邪道編』。「邪道」とう言葉から、「やがて、きっと王道編が出るに違いない」とだれもが思ったものです。しかし、その予想は良い意味で裏切られました。その後『極道（きわめみち）編』、『無道編』、そして『魔道編』までが次々と出版されるとはとても予想できませんでした。しかも、それぞれが「Rubyの周辺」という心憎いポジションを狙っているのも予想以上です。

この連発には、金光さんという編集者が一役買ってるそうです。『極道編』にはその辺の生々しいやり取りが記録されています（『Rubyを256倍使う本 極道編』3ページ）。

「Ruby関連の本を書きませんか？ 何でも好きなことを、好きな分量で、をめざしています」

```
------------------------------------------------------------------------
     enumObj.each_with_index {| obj, i | block }  -> nil
------------------------------------------------------------------------

     Calls block with two arguments, the item and its index, for each
     item in enumObj.

        hash = Hash.new
        %w(cat dog wombat).each_with_index {|item, index|
          hash[item] = index
        }
        hash   #=> {"dog"=>1, "wombat"=>2, "cat"=>0}
```

画面4 「$ri each_with_index」の出力

```
------------------------------------------------------------------------
     Dir.open( aString ) -> aDir
     Dir.open( aString ) {| aDir | block } -> nil
------------------------------------------------------------------------

     With no block, open is a synonym for Dir::new. If a block is
     present, it is passed aDir as a parameter. The directory is closed
     at the end of the block, and Dir::open returns nil.
```

画面5 「$ ri Dir::open」の出力

という金光さんのメールが届いたのが始まりだった。私は戸惑い、

新手の詐欺か新商法

かと疑ったのはいうまでもない。

金光さん、いったい何人に連絡したんでしょう。実は私にも編集部経由で連絡がありました。これから、いったい何冊256倍本が出るんでしょう。1冊の値段が安い（1200円＋税）のが救いです。でも、それぞれ値段分以上の価値はありますよ。

Wiki【うぃき】

「だれにでも書き込めるWeb」という画期的な存在。Rubyの世界でもWikiは活躍しています。RubyによるWikiの実装RWikiは、WebページにRDが使える便利なものです。たとえば、

```
http://www.ruby-lang.org/ja/man-1.6/
```

はRWikiで構成されています（ので、だれでも間違いを修正できます）。そのほかにも「Rubyインストールガイド」の、

```
http://www.ruby-lang.org/~gotoken/install-ja.mrb?cmd=view&name=top
```

やMLトピックス、

```
http://www.jin.gr.jp/~nahi/RWiki/?cmd=view;name=ML+Topics
```

がRWikiを使っています。

海外では、

```
http://www.rubygarden.org/ruby/
```

がWikiによるRubyコミュニケーションを実現していますが、これは諸般の事情からRWikiではなく、（Rubyで実装されていない）別のWikiを使っているようです。

こちらの特徴的なコンテンツには、

・RubyChangeRequest(RCR)

```
http://www.rubygarden.org/ruby?RubyChangeRequest
```

上述のRCRも、メンテナは決まっていますが、Wikiで管理されています

・RubyUsersGroup

```
http://www.rubygarden.org/ruby?RubyUserGroups
```

世界中のRubyユーザーが「ご近所を探せ」状態で登録しています。アメリカ、イギリス、フランス、ドイツ、ロシア、オーストラリア……。本当に世界中ですね。ほとんどのユーザーグループはメンバーが1人とか2人とかですが、なかにはグループミーティングを始めたところもあるようです

irb【あい・あーる・びー】

irbはInteractive Rubyの略で、Rubyを対話的に使うためのツールです。rubyインタプリタがインストールされていれば、irbはほぼ自動的にインストールされるはずです。irbの起動にはirbとタイプするだけです。

```
$ irb
irb(main):001:0> VERSION
"1.6.3"
```

入力した式の結果がすぐに表示されるので、ちょっと試してみたいときなどに便利です。また、readlineライブラリがインストールされていれば、入力時の行編集やヒストリ機能も有効になりますので、ますます便利です。

Rubyのドキュメントがなにを意味するのかわからなくなったら、irbを使って簡単なプログラムを実際に動かして確認するのがよいでしょう。

irbはRubyで記述されていて、入力されたプログラムの構文解析のようなことまで、Rubyでやっているそうです。irbの作者は、Ruby本の共著者の石塚圭樹さんです。

ヤギ本【やぎ・ほん】

八木さんが書いた本ではなくて、まつもとさんによる『Rubyデスクトップリファレンス』の愛称です。まつもとさんは、羊をマスコットにしたいと常々思っていたのだそうです。Ruby本にも、各ページの上のほうに小さく羊とハチドリのマークが付いてますよね。オライリーの本では、表紙の動物がしばしば話題になるので、ぜひ羊をと伝えた

はずなのに、どこをどう間違ったのか、実際に表紙になったのはヤギだったのだそうです（**写真2**）。なお、このヤギはベゾアールヤギという品種だそうです。

一部では、絵本に出てくるヤギにそっくりだということで「がらがらどん」と呼ばれているんだそうですが、知らない人にはなんのことだかわかりませんね。

ヤギ本は「デスクトップリファレンス」シリーズにはあるまじき厚さを誇る分量です。オライリーのデスクトップリファレンスなかでも、最厚ではないでしょうか。それだけ内容が充実しているということでもあるのですが、Rubyの仕様をちゃんと書くとそれだけで結構な分量になるということも意味するような気がします。

『Rubyデスクトップリファレンス』は、海外で英語版の出版が決まったそうです。［ruby-talk:12619］によると、あまりに厚いので（？）、書き足してフルサイズの本にすることになったのだそうです。

POLS【ぽるす？】

Principle of Least Surprise（驚き最少の原則）の略。Rubyの設計原理の1つ。私は「ぽるす」と発音してますが、ほんとはどうなんでしょう。

Rubyで何かをしようとした場合に、人間が一番自然に感じる仕様を選ぶという原則。もっとも、何に驚くかはたいていの場合は人によって異なるので、厳密に言うと「驚き最少」とは、設計者（この場合はまつもとさん）の「驚き」ということになるのでしょう。でも、Rubyを気に入っている人が多いということは、実は「まつもとさんの驚き最少」というのは、結構普遍的なのかもしれません。少なくとも、私を含めて多くの人はRubyの「驚き最少」を気持ちいいと感じてますし、また、日本人以外でも同じことを感じる人はたくさんいるようです。

まとめ

今回は「Rubyにまつわる用語集」ということでまとめてみました。海外情報を中心に、あまり知られていない内容を厳選したつもりですが、皆さんにとって新しい情報がありましたでしょうか。Rubyの今後がますます楽しみではあります。

今回もまた、プログラムを作らなかったので、来月あたりはもうちょっと実用的な話に帰ろうかなあと思います。お楽しみに。

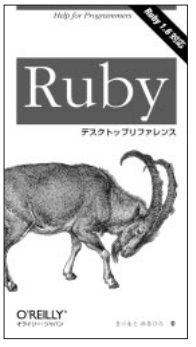


写真2　RUBYデスクトップリファレンス
まつもとゆきひろ著
オライリー・ジャパン
本体価格1000円
ISBN4-87311-023-8

Column

Rubyを256倍使う本 無道編

うーむ、本当に次々と出版されるものです。しかし、「無道」っていったい？　辞書を引くと、

【無道】
道理にそむくこと。人の道にそむくこと。

だそうです。なんて名前だ。どういう経緯でこういう名前が付いたのだか、ぜひとも聞きたいものです。とにかく、この『無道編』のテーマはRaccです。Raccというのは、Rubyで書かれたパーサ・ジェネレータです。パーサ・ジェネレータというのは…（以下6段省略）。ま、はっきり言ってしまうと、普段の生活ではあまり使うことのないツールに思われがちなものです。パーサ・ジェネレータの代表例であるyacc（GNU版はbison）について解説した書籍でさえ、片手の指の数で足りるくらいしか出てないのに（しかも、その大半は絶版）、256倍シリーズで出しちゃうとは。やはり、『無道編』の名は伊達じゃない？

だからといって、難しいとか、わかりにくいと思う必要はありません。むしろ、256倍らしいとてもわかりやすい語り口で、Raccとその周辺を解説してくれています。過去3冊のRuby256倍本の中では、最も「256倍テイスト」が強いのではないでしょうか？　また、周辺の話題として、ファイル構成や、インストーラ、それから開発の裏話がとても楽しい本です。楽しく読めて、学べてしまう、一粒で二度おいしい一冊でしょう。

写真3　Rubyを256倍使う本 無道編

[超]入門シェルスクリプト

今回は、if制御構造よりも簡潔に多分岐処理を記述できるcase制御構造や、そのパターン中で使えるワイルドカード、for制御構造とcase制御構造の組み合わせ方などについて説明したのち、さまざまな形式のアーカイブファイルを展開するシェルスクリプトを作成する。

第7回　case制御構造による多分岐処理

文：大池浩一
Text:Koichi Oike

前回までで学んできたif制御構造とtest（あるいは[とも書く。ここではこちらを使う）の組み合わせを使えば、条件式を用いた分岐処理のあらゆるパターンをシェルスクリプトで実現できる。

たとえば、最初の条件式を評価して、その結果が偽（不成立）なら第1の条件式を評価し、それも偽なら第3の条件式を評価し…、というように、複数の条件式を次々に評価する「多分岐処理」は、

```
if [ 最初の条件式 ]; then
  最初の条件式が真（成立）の場合のコマンド
elif [ 第2の条件式 ]; then
  第2の条件式が真（成立）の場合のコマンド
elif [ 第3の条件式 ]; then
  第3の条件式が真（成立）の場合のコマンド
            :（中略）
else
  どの条件式も偽（不成立）の場合のコマンド
fi
```

のように書けばいい。

この方法では、ifやelifの条件部で実行されるtest（あるいは[）に、それぞれ異なる条件式を指定できる。そのため、elifの数が多くなると、どのような処理をしているのか一目ではわからなくなってしまう。

case制御構造による多分岐処理を行う

シェルスクリプトで必要になる多分岐処理には、「あるシェル変数の内容を複数の値と比較して、等しい場合はそれぞれの処理を行う」というものが多い。その典型例がスクリプト起動時のオプション処理で、コマンドライン引数を「-a」などの文字列と比較し、それぞれのオプションごとに異なるコマンドが実行される。

こうした単純な多分岐をif制御構造で記述すると、条件分岐とコマンドを条件の数だけ並べたものになるため、見た目に複雑なスクリプトになってしまう。そこで、bashを含むBourne系シェルには、この手の多分岐処理をすっきりと記述する「case制御構造」が用意されている。1つの式（たいていはシェル変数の内容）を複数のパターンと順番に比較し、マッチした場合には指定されたコマンド群を実行するというものだ。

以下では、

- **case制御構造の書式と利用方法**
- **複数パターンやワイルドカードの利用**
- **for制御構造とcase制御構造の組み合わせ方**

について説明した後、case制御構造を利用したスクリプトの作成を行う。

●case制御構造の書式と利用方法

case制御構造の書式は以下の通りだ。

```
case 式 in
パターン1 ) コマンド群1 ;;
パターン2 ) コマンド群2 ;;
        :
      (中略)
        :
パターンN ) コマンド群N
esac
```

全体の流れとしては、caseの直後に指定された「式」の値と「パターン1」～「パターンN」が、上から順番に1つずつ比較される。式の値がいずれかのパターンにマッチすると、対応する「コマンド群」が実行され、case制御構造の残りの部分はすべてスキップされる。

「式」には、通常の文字列やコマンド置換、数値演算結果で置換される$((…))構造も記述できるが、たいていはシェル変数の内容に展開される「$変数名」を記述する。たとえば、シェル変数fooの内容によって多分岐処理を行う場合、caseの最初の行は、

```
case "$foo" in
```

と書く。

なお、周囲を「"」で囲む（クォーティングする）のは、変数の内容に空白文字（スペースやタブ、改行）が含まれていても正常に処理できるようにするためだ。

2行目以降には、「パターン）コマンド群 ;;」という書式で、式の値と比較される「バターン」と、式の内容がパターンにマッチした場合に実行される「コマンド群」の組を指定する。

パターンには、通常の文字列を指定するほか、ワイルドカードを含めたり、複数のパターンを並べることもできる（これらについては後ほど説明する）。一方、コマンド群には、外部コマンドやシェルの組み込みコマンドを記述する。複数のコマンドを改行や「;」で区切って並べることも可能だ。

たとえば、式の内容が「hoge」の場合に、「cd ~/hoge」と「ls -l」を実行するには、

```
hoge ) cd ~/hoge ; ls -l ;;
```

と1行にまとめて書くか、あるいは、

```
hoge ) cd ~/hoge
       ls -l ;;
```

▼リスト1　前回のスクリプト

```
#/bin/sh
month=`date +%m`
if [ $month -eq 1 ]; then
  echo "睦月（むつき）"
elif [ $month -eq 2 ]; then
  echo "如月（きさらぎ）"
elif [ $month -eq 3 ]; then
  echo "弥生（やよい）"
elif [ $month -eq 4 ]; then
  echo "卯月（うづき）"
elif [ $month -eq 5 ]; then
  echo "皐月（さつき）"
elif [ $month -eq 6 ]; then
  echo "水無月（みなづき）"
elif [ $month -eq 7 ]; then
  echo "文月（ふみづき）"
elif [ $month -eq 8 ]; then
  echo "葉月（はづき）"
elif [ $month -eq 9 ]; then
  echo "長月（ながつき）"
elif [ $month -eq 10 ]; then
  echo "神無月（かんなづき）"
elif [ $month -eq 11 ]; then
  echo "霜月（しもつき）"
elif [ $month -eq 12 ]; then
  echo "師走（しわす）"
fi
```

Column

esacとは一体何か

本文中でも触れているように、case制御構造のブロックの終わりを示すために使われる「esac」は、「case」の綴りを逆さにしたものだ。同様に、通常の分岐を行うif制御構造でもブロックの終わりに逆綴り単語の「fi」が使われる。こうした見なれない逆綴り単語を使うのは、Bourne系シェルに独特の「クセ」だ。tcshなどcsh系のシェルでは、制御構造のブロックの終わりに「endif」や「endsw」などのわかりやすい単語が使われる。

幸いなことに、Bourne系シェルでも、繰り返し制御構造（for、while、untilの3種類）については、ブロックの始まりは「do」、終わりは「done」を使うように統一されている。「rof」とか「elihw」、「litnu」なんて単語は登場しないので安心してほしい。

のようにコマンドごとに別の行に書く。いずれの場合も、次のパターンとの間は「;;」で区切らなくてはならないことに注意しよう。なお、パターン中に「)」が含まれる場合、そのままでは最初の「)」の直前までがパターンだと判断されてしまうので、パターン全体を「'」でクォーティングする必要がある。たとえば、「(hoge)」という文字列をパターンに指定するには、

```
'(hoge)' ) コマンド群 ;;
```

とする。また、パターンの先頭や末尾に空白文字が含まれる場合も同様だ。

こうして、処理に必要な数だけパターンとコマンド群を記述したら、最後にcase制御構造の終わりを示すキーワード「esac」を書く。これは「case」を逆向きに綴ったものだ（コラム参照）。

なお、一番最後のコマンド群の末尾には「;;」を記述しなくてもいい。しかし、スクリプトを書き換えている最中には、新たなパターンを後から追加することがよくあるので、どのコマンド群の後ろにも常に「;;」を書くようにしておいたほうがいいだろう。

case制御構造を使った多分岐処理の例として、現在が何月か調べて、月ごとの和名を表示するスクリプトを考えよう（if制御構造を用いた例が先月号の本連載記事に掲載されている）。現在の月の数字を得るには、外部コマンドdateの引数として「+%m」を指定すれば得られる。

```
$ date +%m
04
```

スクリプトでは、case制御構造の式として「date +%m」のコマンド置換を記述し、各月の数字のパターンと比較すればいい。なお、「date +%m」の出力は2桁に固定されているので、1月から9月までは「0」付きになることに注意されたい。具体的なスクリプトは次のようになる。

```
#/bin/sh
case `date +%m` in
01) echo "睦月（むつき）" ;;
02) echo "如月（きさらぎ）" ;;
03) echo "弥生（やよい）" ;;
04) echo "卯月（うづき）" ;;
05) echo "皐月（さつき）" ;;
06) echo "水無月（みなづき）" ;;
07) echo "文月（ふみづき）" ;;
08) echo "葉月（はづき）" ;;
09) echo "長月（長月）" ;;
10) echo "神無月（かんなづき）" ;;
11) echo "霜月（しもつき）" ;;
12) echo "師走（しわす）" ;;
esac
```

スクリプトの細かな動作については図を参照のこと。前

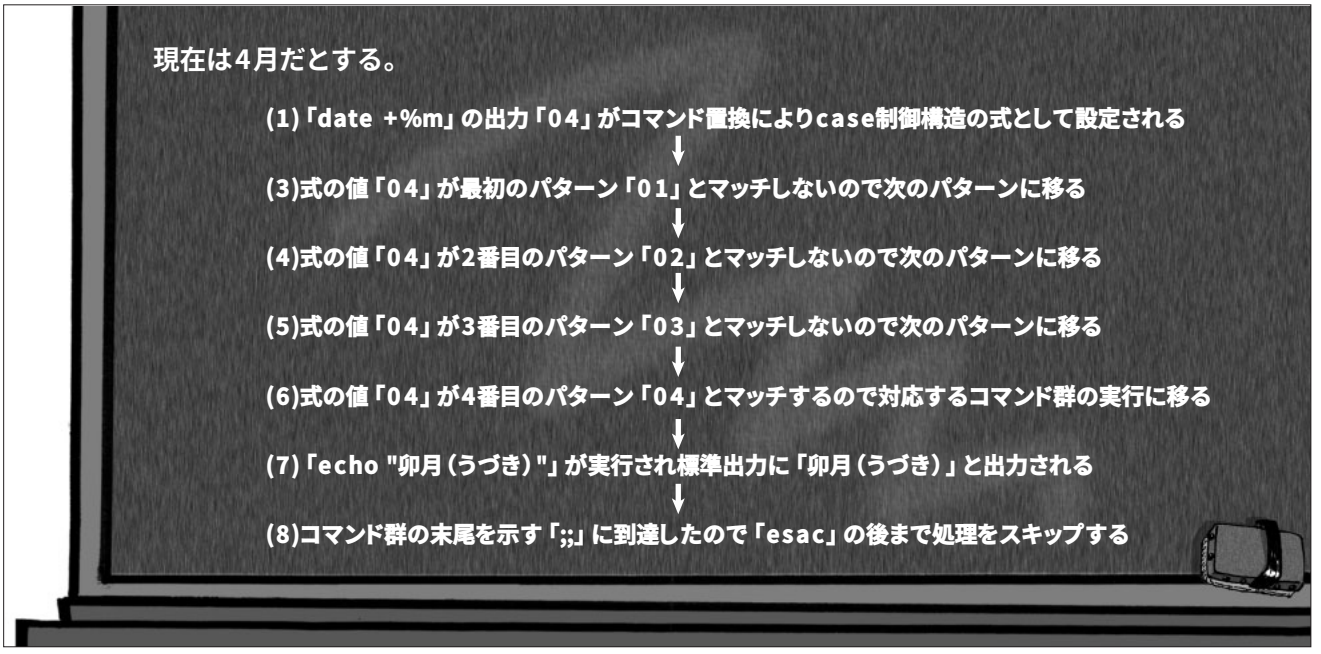


図　case制御構造を使った多分岐処理の例

回のif制御構造を使ったスクリプト（**リスト1**）と比べてみると、全体の行数が少なくなり、内容も理解しやすくなっていることがわかるだろう。

要点をまとめると、

- **case制御構造では、式の値が最初のパターンから順に比　較され、初めてマッチしたパターンに対するコマンド群　だけが実行される。**
- **式の部分には、シェル変数の内容に展開される「$変数名」や、コマンドの実行結果で置換される「`コマンド`」などを記述する。**
- **パターンに「)」や空白文字が含まれる場合には「'」でクォーティングする。**
- **それぞれのコマンド群の末尾には「;;」が必要。**

ということになる。

●複数パターンやワイルドカードの利用

case制御構造では、パターンにワイルドカードを含めたり、複数のパターンを「｜」で区切って並べることで、ひと組の「パターンとコマンド群」で、さまざまな式の値をまとめて処理できる。

(1)パターンにワイルドカードを含める

パターンには、コマンドラインでファイル名の展開に使われるのと同じワイルドカード（**表**）を含めることができる。任意の文字列を表わす「＊」や、任意の1文字を表わす「？」などの特殊記号を利用することで、複数の文字列にマッチするパターンを表現できるわけだ。

たとえば、シェル変数fにファイル名が格納されているとしよう。そのファイル名の拡張子が「.txt」の場合に、lessでファイル内容を表示するには、

```
case "$f" in
*.txt ) less "$f" ;;
esac
```

と書けばいい。「＊.txt」というパターンは、末尾が「.txt」の文字列（たとえば「hoge.txt」や「readme.txt」など）すべてにマッチするからだ。

ワイルドカードのセット構造（コラムを参照）を使えば、もっと複雑なパターンも指定できる。たとえば、上の例で、拡張子が「.Txt」の場合にも対応するには、パターンを「＊.[tT]xt」に変えればいい。このパターンは、文字列の末尾が「.txt」でも「.Txt」でもマッチする。

なお、コマンドライン中のワイルドカードは、実際のファイル名に展開されてからコマンドに渡されるのに対し、case制御構造のパターン中のワイルドカードは、ファイル名に展開されない点に注意してほしい。つまり、case制御構造の式の値が、ファイル名とは何の関係もない単なる文字列であっても、ワイルドカードを含んだパターンによるマッチを行えるということだ。

(2)複数のパターンを「｜」で区切って並べる

ワイルドカードを使ってもひとつのパターンにまとめられなかったり、まとめると複雑になってしまう場合は、複数のパターンを「｜」（パイプ）で区切って並べればいい。この場合、式の値がどれかひとつのパターンにマッチすれば、対応するコマンド群が実行される。

たとえば、シェル変数fに格納されたファイル名の拡張子が「.jpg」か「.jpeg」の場合に、画像ビューアのdisplayを使ってその内容を表示するには、

```
case "$f" in
*.jpg｜*.jpeg ) display "$f" & ;;
esac
```

と書けばいい。「＊.jpg｜＊.jpeg」というパターンは、末尾が「.jpg」か「.jpeg」の文字列すべてにマッチする。また、displayは端末画面を使わないXアプリなので、コマンドラインの末尾（「;;」より前）に「&」を付けてバックグラウンドで実行している点にも注意しよう。

要点をまとめると、

- **パターン中でワイルドカードを利用できる**
- **ワイルドカードのセット構造を活用しよう**
- **複数のパターンを「｜」で区切って並べられる**

ということになる。

●for制御構造とcase制御構造を組み合わせる

case制御構造では、一度に1つの式の値しか扱えないの

?	任意の1文字にマッチ
＊	1文字以上の任意の文字列にマッチ
[…]	大カッコ内のいずれか1文字にマッチ
[!…]	大カッコ内の文字以外の1文字にマッチ

表　Bourne系シェルのワイルドカード

で、複数の対象を扱うには、for制御構造を使った繰り返し処理と組み合わせて順番に処理する必要がある。典型的な例として、コマンドライン引数に指定されたファイル名に対し、それぞれの拡張子に応じたコマンドを実行するスクリプトを作ってみよう。

コマンドライン引数のリストは「"$@"」で参照できる。これは、スクリプト実行時のすべてのコマンドライン引数を1つずつ「"」でクォーティングし、スペースで区切って並べたものと同じだ。

for制御構造のリストに「"$@"」を指定して、

```
#!/bin/sh
for f in "$@"; do
  繰り返す内容
done
```

とすると、先頭の引数から順番に内容がループ変数fに格納され、そのたびに「繰り返す内容」に記述されたコマンドが実行される。

続いて、case制御構造を「繰り返す内容」に記述する。式にはループ変数fの内容が展開される「$f」を書けばいい。空白文字が含まれる可能性を考慮して、「"$f"」とクォーティングしたほうがよいだろう。前節の2つの例をパターンとコマンド群として組み込むと、スクリプトは以下のようになる。

```
#!/bin/sh
for f in "$@"; do
  case "$f" in
  *.txt ) less "$f" ;;
  *.jpg | *.jpeg ) display "$f" & ;;
  esac
done
```

このスクリプトを実行すると、引数で指定したファイル名が「.txt」で終わる場合はlessでファイルの内容（テキスト）が表示され、「.jpg」か「.jpeg」で終わる場合はdisplayでファイルの画像が表示される。

なお、このスクリプトでは、ファイルごとにlessやdisplayが実行される。lsのようにすぐに終了するコマンドなら問題ないが、lessの場合はqキーが押されるまで終了しないため、繰り返し処理が一時停止する。一方、displayのほうはバックグラウンドで実行されるので一時停止の問題はないものの、すべての画像が別のプロセスで表示されるためメモリを消費してしまう。

これらの問題を解決するには、繰り返し処理を終えた時点でそれぞれのコマンドを起動するようにスクリプトを書き換えればいい。

```
#!/bin/sh
for f in "$@"; do
  case "$f" in
  *.txt ) txtfiles="$txtfiles $f" ;;
  *.jpg | *.jpeg ) jpgfiles="$jpgfiles $f" ;;
  esac
done
if [ -n "$jpgfiles" ]; then
  display $jpgfiles &
fi
if [ -n "$txtfiles" ]; then
  less $txtfiles
fi
```

各パターンのコマンド群では、シェル変数txtfilesとjpgfilesに、パターンにマッチしたファイル名（$fで参照）を追加していく。繰り返し処理が終わったら、if制御構造を使ってtxtfilesやjpgfilesの内容が空でないか調べ、空でない場合にはそれを引数に指定してlessやdisplayを起動する。

なお、ファイル名に空白文字を含む場合に対応すると、

Column

ワイルドカードのセット構造

Borne系シェルのワイルドカードには、任意の文字列を表わす「＊」と、任意の1文字を表わす「？」のほか、文字のリスト（または範囲）を指定するセット構造[…]が用意されている。リストを指定する場合は、大カッコの中に直接文字を列挙し、範囲を指定する場合は最初と最後の文字の間に「ー」を書けばいい。

たとえば[ab]と書くと、「a」か「b」のいずれか1文字、[0-9]と書くと、「0」から「9」までの数字1文字にマッチする。

[0-9abcdef]のように、リストと範囲を混ぜて書くことも可能だ。こうしたセット構造は、コマンドラインではキー入力が多くなってしまうのでそれほど使われないが、厳密な処理が必要なスクリプトでは積極的に利用するとよいだろう。

もう少しスクリプトが複雑になってしまう。腕に自信のある人は挑戦してみよう（ヒントは環境変数IFS）。

この節での要点をまとめると、

- **複数の対象を扱うには、for制御構造の繰り返し処理の中でcase制御構造を使う。**
- **forのリストには「"$@"」、caseの式にはfor制御構造の　ループ変数を「"$変数名"」として記述する。**
- **すぐに終了しないコマンドは、繰り返し処理が終わったあとでまとめて実行したほうがいい。**

ということになる。

今月のスクリプト

後半は、既存のスクリプトの内容を解説する「スクリプトを読む」と、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「スクリプトを書く」で構成する。今月はページ数の都合上「スクリプトを読む」はお休みし、「スクリプトを書く」では、

- **指定したアーカイブファイル（複数可）を、それぞれの拡張子に対応したソフトで展開する「extract」**

を作成する。

UNIX系OSで使われる典型的なtarボール（tarとgzip／bzip2の組み合わせ）や、主にWindowsやDOSで使われるZIP／LHAなど、さまざまなアーカイブファイルを統一された操作で展開できるようにする。また、起動時オプションにより、展開先ディレクトリを指定したり、展開の代わりにファイル一覧の表示に切り替えられるようにしよう。

スクリプトの構成は、大まかにいって、

- **起動時オプションの処理**
- **ファイル名とパターンとの比較**
- **アーカイブファイルに対するコマンド群の記述**

という3つの部分に分かれている。以下では、それぞれの処理について詳しく説明しよう。

●起動時オプションの処理

スクリプトextractでは、以下の2つのコマンドラインオプションを処理することにする。

-l　展開ではなくファイル一覧を表示する。
-O　直後の引数で展開先ディレクトリを指定する。

上記以外のオプション（たとえば「-h」など）が指定された場合は、使い方（Usage）を表示して、終了ステータス1でスクリプトの実行を終了させる。

この部分の処理は、for制御構造とcase制御構造を組み合わせて、以下のように書ける。

```
for f in "$@"; do
  case "$f" in
  -l ) list=yes
       shift ;;
  -O ) extdir="$2"
       shift 2 ;;
  -* ) echo 'Usage: extract [-l] [-O dir] files...'
> /dev/stderr
       exit 1 ;;
  esac
done
```

まず、for制御構造によって、コマンドライン引数の値が1つずつループ変数fに格納され、case制御構造で多分岐処理が行われる。

引数に「-l」を指定した場合は、シェル変数listに「yes」が設定される。なお、このまま「-l」を引数に残しておくと、あとの処理でファイル名として扱われてしまい、エラーの原因になる。そこで、シェルの組み込みコマンドshiftを使って、コマンドライン引数の内容を1つずつ前に詰め、「-l」を引数から削除している。

引数に「-O」が指定された場合の処理は、もう少し複雑になる。というのも、「-O ~/src」のように、-Oの直後の引数で指定されるディレクトリも一緒に処理する必要があるからだ。オプション文字列自体はループ変数fに格納されているので「$f」で参照できる。その直後の引数はどのようにして指定すればよいだろうか。

ポイントは、shiftを使った引数詰めにより、処理が終わったオプションが即座に引数から削除されることだ。つまり、現在処理しているオプションは必ず最初の引数（$1で参照）になっている。その直後の引数は2番目の引数なのだから、「$2」で参照できるわけだ。

スクリプトでは、シェル変数extdirに（その時点での）2番目の引数の内容を格納している。また、shiftによる引数詰め処理では、最初の引数と2番目の引数を同時に削除するため、「shift 2」として引数を2つずつ前に詰めるよう指示している。

それ以外のオプションは、最後のパターン「-*」で初めてマッチするので、使い方を標準エラー出力に表示し、終了ステータス1でスクリプトを終了する。

なお、このスクリプトでは、「-」で始まるファイル名を扱えないし（理由は各自で考えよう）、「-lO」や「-O~/src」といったオプション指定も正しく処理できない。このようなオプション指定を柔軟に処理するには、外部コマンドのgetoptや、bashの組み込みコマンドgetoptsを利用する必要がある。getoptとgetoptsの使い方については次号で取り上げることにしよう。

●ファイル名とパターンとの比較

続いては、コマンドライン引数で指定されたファイル名を、case制御構造のパターンと比較する多分岐処理について考えよう。今回のスクリプトで扱うアーカイバ（および圧縮ソフト）と、それに対応するアーカイブファイル名のパターンを以下に列挙する。

- **tar+gzip**　「*.tar.gz」「*.tgz」
- **tar+bzip2**　「*.tar.bz2」
- **tar**　「*.tar」
- **LHA**　「*.LZH」「*.lzh」
- **ZIP**　「*.ZIP」「*.zip」

前半で説明したfor制御構造とcase制御構造の組み合わせを使うと、この部分の処理のアウトラインは以下のように書くことができる。

```
for f in "$@"; do
  case "$f" in
  *.tar.gz | *.tgz ) tar+gzip用コマンド群 ;;
  *.tar.bz2        ) tar+bzip2用コマンド群 ;;
  *.tar            ) tar用コマンド群 ;;
  *.LZH | *.lzh    ) LHA用コマンド群 ;;
  *.ZIP | *.zip    ) ZIP用コマンド群 ;;
  *                ) 対象外ファイル用コマンド群 ;;
  esac
done
```

最後のパターン「*」はすべての文字列とマッチする。最後のパターンに到達するのは、それより上のパターンのいずれにもマッチしなかった場合だけだ。つまり、このスクリプトでは扱わないファイルということになる。

●アーカイブファイルに対するコマンド群の記述

最後に、それぞれのパターンに対応するコマンド群で、アーカイバを起動する処理を行う。なお、アーカイバは展開（あるいは一覧表示）処理後すぐに終了するため、for制御構造の中でそのまま実行して構わない。

たとえば、tar＋gzipされたアーカイブファイルをカレントディレクトリに展開するには、

```
$ tar xzf アーカイブファイル名
```

とする。オプションのxは展開、zはgzipの利用、fは直後の引数でのファイル名の指定をそれぞれ意味している。展開先のディレクトリを指定するには、-C（大文字）オプションを追加して、

```
$ tar xzf アーカイブファイル名 -C ディレクトリ
```

とすればいい。

一方、アーカイブ内のファイルを一覧表示するには、

```
$ tar tvzf アーカイブファイル名
```

とする。オプションのtは一覧表示、vは表示内容を詳細にすることを意味する。そのほかは展開処理と同じだ。

実際のスクリプトでは、起動時オプションの処理で設定したシェル変数listやextdirの内容に応じて、これらのコマンドラインを使い分ければいい。

```
if [ -n "$list" ]; then
  tar tvzf "$f"
elif [ -n "$extdir" ]
  tar xzf "$f" -C "$extdir"
else
  tar xzf "$f"
fi
```

まず、シェル変数listの内容が空でないか調べ、空でない（すなわち起動時に-lを指定した）場合は「tar tvzf

"$f"」で一覧表示処理を行う。一方、シェル変数listの内容が空の場合は、シェル変数extdirの内容が空でない（起動時に-Cを指定した）かどうか調べ、空でない場合は「tar zxf "$f" -C "$extdir"」で展開先ディレクトリの指定を含んだ展開処理を行う。extdirが空の場合は、「tar xzf "$f"」により、カレントディレクトリに展開する処理を行っている。

さらに、シェル変数が未定義（または内容が空）の場合に、指定した文字列で置換される「${変数名:-文字列}」という置換演算子を使うと、展開処理のコマンドラインをひとつにまとめられる。

```
tar xzf "$f" -C "${extdir:-.}"
```

スクリプト起動時に-Oオプションを指定しなかった場合は、extdirが未定義になるので、「${extdir:-.}」は「.」（カレントディレクトリ）に置換される。つまり、カレントディレクトリが展開先に指定されるわけだ。

tar＋bzip2やtarのみ（無圧縮）のアーカイブファイルに対する処理は、tar＋gzipの場合のtarのオプションを一部変更するだけで対応できる。一方、LHAやZIPのアーカイブファイルを扱うコマンド（lhaおよびunzip）は、オプションの文字や並べ方がかなり異なる。たとえば、展開処理の場合、LHAでは「lha xw=ディレクトリ アーカイブファイル名」だし、ZIPでは「unzip アーカイブファイル名 -d ディレクトリ」だ。

こうしたコマンド処理を加えた完成版のスクリプトextractが**リスト2**だ。

このスクリプトでは、最初に使い方（Usage）をシェル変数usageに格納（2行目）した後、コマンドライン引数の数を調べ、1つも指定されていない場合には使い方を表示して終了する（3～6行目）。

続いて、本文中で説明した起動時オプションの処理（7～16行目）を行う。

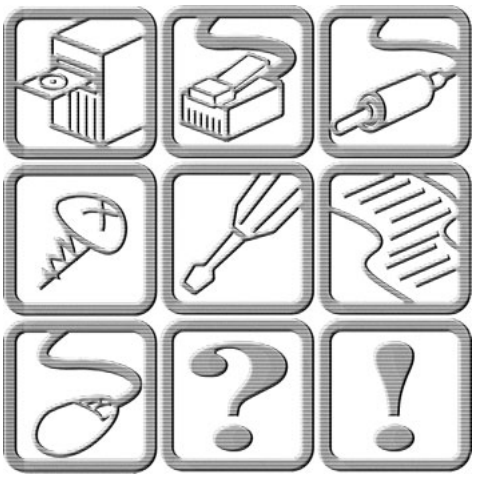
オプション処理が終了すると、コマンドライン引数にはアーカイブファイル名しか残っていない。そこで、再度コマンドライン引数の数を調べ、ファイル名が1つも指定されていなければ使い方を表示して終了する（17～20行目）。

最後に、コマンドライン引数からアーカイブファイル名を1つずつ取り出し、それぞれの拡張子に対応するアーカイバを使って、展開（または一覧表示）処理を行っている（21～55行目）。

▼リスト2　スクリプトextract

```
 1: !#!/bin/sh
 2: usage='Usage: extract [-l] [-O dir] files...'
 3: if [ $# = 0 ]; then
 4:   echo "$usage" > /dev/stderr
 5:   exit 1
 6: fi
 7: for f in "$@"; do
 8:   case "$f" in
 9:   -l ) list=yes
10:        shift ;;
11:   -O ) extdir="$2"
12:        shift 2 ;;
13:   -* ) echo "$usage" > /dev/stderr
14:        exit 1 ;;
15:   esac
16: done
17: if [ $# = 0 ]; then
18:   echo "$usage" > /dev/stderr
19:   exit 1
20: fi
21: for f in "$@"; do
22:   case "$f" in
23:   *.tar.gz|*.tgz )
24:     if [ -n "$list" ]; then
25:       tar tvzf "$f"
26:     else
27:       tar xzf "$f" -C "${extdir:-.}"
28:     fi ;;
29:   *.tar.bz2 )
30:     if [ -n "$list" ]; then
31:       tar tvIf "$f"
32:     else
33:       tar xIf "$f" -C "${extdir:-.}"
34:     fi ;;
35:   *.tar )
36:     if [ -n "$list" ]; then
37:       tar tvf "$f"
38:     else
39:       tar xf "$f" -C "${extdir:-.}"
40:     fi ;;
41:   *.lzh|*.LZH )
42:     if [ -n "$list" ]; then
43:       lha l "$f"
44:     else
45:       lha xw="${extdir:-.}" "$f"
46:     fi ;;
47:   *.zip|*.ZIP )
48:     if [ -n "$list" ]; then
49:       unzip -l "$f"
50:     else
51:       unzip "$f" -d "${extdir:-.}"
52:     fi ;;
53:   * ) echo "extract: $f is not supported." >
      /dev/stderr ;;
54:   esac
55: done
```

Try & Try [trái] [trái]



我が家のネットワークの調整（1）

文：政久忠由
Text : Tadayoshi Masahisa

最近よく目にするキーワードのひとつに『ブロードバンド』がある。

厳密な意味はとりあえず置いておくとして、これは、消費者うけを狙った広帯域伝送技術をベースにした大容量の高速通信サービスのマーケティング的なキャッチ用語として使われている。まぁ何をもって広帯域で、どこから大容量で高速なのかは非常にあいまいではあるけれど、なんとなくイメージできるような気がしないでもないし、大容量とか高速といってしまうより、実はブロードバンドといったほうがさらにあいまいさを増して、結構便利に使えそうではある。

個人的には、現行の地上波のテレビレベルの動画像を送受信できる程度の通信がブロードバンドの最低基準（いくつか手持ちのソースをエンコードしてみた感じだと1500Kbps以上、と言いたいところだけど、768Kbpsでも許容範囲かな。映像のタイプによるんだよね）だと思っていたんだけど、最近のサービスは帯域保証ではないものの、最高値は実際そのレベルにある。そういった意味では、ブロードバンドっていうのは、まんざらアドバルーンではない（あれ、B-ISDNはどこにいったのかしらん？）。

でもそれらのコンテンツサービスは始まろうとしている段階で、多くのユーザーが満足できるレベルに成長するには、相当の時間とお金が必要である。いやぁー、各人にとって、インターネット上の情報の99.999999999％は必要のないゴミといわれているが、現段階では本当にロクなコンテンツがなくて呆れてしまう。金をかければよいものが作れるわけではないが、今は金も知恵もないらしい。だいたいインターネットやパソコンにすべてが集約されるなんて幻想でしかないし、しかもハッキリ言って将来的には必要なのかもしれないけれど、近々にはそんなもの必要ないという現実もあって、そういろんなことがスンナリとは進むはずもない。少なくとも僕はそう思う。

もちろんユーザーとISP（アクセスサーバ）の間が速くなったとしても、上位のネットワークやインターネットは一人で占有できるものではなく、何千、何万、何百万人以上でシェアリングしている状態だ。数10Gや数Tbpsのネットワークが実用段階に入ったとはいえ、ユーザーから徴収できる金額を考えると、おいそれとバックボーンを増強するのは難しい。通常の電話は加入者の10分の1程度の実効設備だったと思うが、インターネットの場合は、良くて収容ユーザー×帯域の100分の1程度が基準であろう（あくまでこれは一般的な考え方であって、当然それぞれのISPでは独自の基準で収容率を決める。それ相応の金額を払えば帯域をある程度保証してくれるところもあるが、一般ユーザーの利用に見合ったものではない）。まぁ現実はもっとひどい状態のISPが多いことも確かだと思う。

やれやれADSLだ

僕は当初ADSLを導入する気はなかった。そりゃあパソコンと同じで、インターネット接続に関しても速いに越し

たことはないのだけれど、よくよく考えるとそんなに必要ないんだよね、これが。

たぶん一般的なユーザーの多くは、コストパフォーマンスを最も大切に考えていると思う。僕もその一人だ。個人向けに10Gbpsの回線を月10万円（1Mあたり10円）で、と言われても加入する気にはならないが、64Kbpsの回線が月500円だとしたら、迷わず後者を選ぶ。ここにADSL（下り1.5Mbps程度）が月6000円（何だかんだで8000円）が加わったとしても僕は64Kbpsの回線を選ぶ。しかし現実には、ISDNとISPの利用に月6000円以上かかっている。こうなるとADSLを選択するのは、好きとか嫌いとか、必要とか必要ないとかの検討以前の問題なのだ。個人的には、512Kbpsでいいから月2500円で提供してくれまいかと思っているのだが、ダメかなぁ。

さて、そんなこんなで我が家はADSLになった。手続きとか、開通への道とかの話は、申し込んで、待って、待って、待って、忘れかけた頃にNTTの机上審査がどうたらというメールが来て、その10日後くらいに開通したというだけで、特に何かがあったわけではないので省略する。ADSLモデムはDIYってことで、宅配便で開通日の当日の朝やってきたのを眠いなぁと思いながら適当に取り付けた。そしてコンピュータを接続して回線状態を確認する。pingとtracerouteで近場の地力のありそうなサーバを探して、ftpで600Mバイト程度のCDイメージのダウンロードである。186Kバイト／秒。次は割り当てられた自分のftpスペースにアップロード。30Kバイト／秒。特に問題はないようだ。この手のものは、何かトラブルがないと少々手持ち無沙汰の感があったりするけれども、本来よろこばしいことで、そうでなければ困るんだけどね。

というわけでADSLについて述べることはもうない。特に問題もなかったので、僕が利用しているADSLの伝送方式が、ITU-TのG.992.1、G.992.2、そしてそれらのAnnex A、Annex Cのどれなのかはよくわからないし、知ってどうなるってものでもないから、ITU-Tのサイトに行ってはみたものの、仕様書を購入するようなまねはしなかった（ちなみにG.992.1は96スイスフラン、G.992.2は73スイスフラン。製本しているならあれだけど、オンラインドキュメントなのに結構高いねぇ）。

ちょっと困ったこと

ADSLモデムは、面倒な設定とか中継するコンピュータ（ルータ）を別途用意したくなかったので、ルータタイプにしたのだけれど、残念なことにその設定は我が家のネットワーク構成とは相容れないものであった。しかも、今のところモデムの設定ユーティリティが提供されていないのだ。

設定されているIPアドレスやネットマスク自体はふり直せばいいことで問題ないが、ADSLモデムに簡易DHCPサーバとして機能されては困るのである。我が家のネットワークでは、DNSサーバとDHCPサーバが効果的に機能していて、どちらも欠かすことができない存在なのだ。同一セグメントに複数のDHCPサーバがあるのは非常に困る。クライアントコンピュータは、ADSLモデムのDHCPサーバの提案を受け取るとDNSサーバにADSLモデムのIPアドレスを登録する。実はこうなると各マシンのログオンに支障をきたすのだ。そう我が家はActive Directoryなのである。

こんなもの単にADSLモデムのDHCP機能を無効にするだけで問題は解消するのだけれど、結局設定することができない。まぁクライアントコンピュータといっても、全部で5台なので、スタティックに設定してもたいした手間ではない。一応、ISPに強くお願いすれば一定の手順を踏んでユーティリティの提供もなされるようでもある。

しかし、どちらにしても不毛な労力を要するだけで、僕にとってはメリットが少ない。

そこで僕は、内部的にゲートウェイを設置するという方法を選択した。

ちょうど複数のネットワークカードを搭載しているマシンがあったので、そいつにルータをやらせることにした。このマシンはWindows 2000 Professionalが稼動していた。ルータ機能を有効にするのは雑作もないことだったのだが、ADSLモデムは通信を許可するIPアドレスを制限していて、パケットが届いてもドロップしてしまっている。

つまるところNATしなければならない。ADSLモデムに受け入れてもらえるように、内々のネットワークの通信をこのゲートウェイでアドレス変換し、さらにADSLモデムでアドレス変換を行いインターネットに出ていく、という2重のアドレス変換を行うハメになってしまった。アホらしいけど仕方がない。

それでまぁWindows 2000 ProfessionalにもNAT機能があるので、それを有効にしてしばらく利用していたのだけれど、このマシンはテスト用なので常設しておくことはちと難しい。というわけで、ゲートウェイの選定を見直す必要が生じた。我が家で常時稼動しているマシンは、ドメインコントローラとLinuxのテストマシンである。どちらも必要条件は満たすが、それぞれ不安もある。ドメインコ

ントローラには大切なデータを集中させているので、さすがに余計なことはさせたくないし、Linuxのテストマシンは、常時稼動しているものの、再起動も頻繁にさせちゃうかもという不安だ（再起動のコマンドを送るのは僕なのだが）。まぁ本当のところは、考える余地なくLinuxマシンに決めていて、常設のマシンを別途作るかどうかの思案をしていたのだけれども。

ゲートウェイマシン

で、僕の結論としては、常設するかもしれない、でもしばらく運用してみてから決めようというという極めていい加減なものだ。そこで常用Linuxマシンのシステムの複製を作成し、最低限Linuxが動作する必要なデバイスだけにダイエットしたマシンをでっち上げ、そこで稼動させ、これをゲートウェイとした。適当なディストリビューションをインストールしてもよかったのだけれど、好みの設定にするのに時間がかかるので現状のLinuxシステムのクローンを使ったわけだ。もちろんホスト名やネットワークの設定は変更する必要がある。また実際には、両ネットワークインターフェイスともDHCP設定で、スタティックな設定はしていない。内々のネットワーク側は、MACアドレス指定の配布なので特に問題はないが、モデム側はモデムの気分次第となる（それなりの規則はあるが）。通常は、ゲートウェイマシンにはあるまじき行為だが、管理は楽だ。ただこれは内々のゲートウェイマシンだからそうしたのであって、する人はいないと思うけど、本当の外にさらすマシンでは絶対にやっちゃだめだ。

■ゲートウェイマシンにさせようと思うお仕事

このゲートウェイマシンには、3つのお仕事をさせる。1つ目は、NATで内々のネットワークから外が見られるようにすることと、テストも兼ねてNetFilter機能を楽しんでみること。2つ目はProxyキャッシュサービス。3つ目は日頃チェックしているソフトウェアの収集。詳細は追って説明するが、ダイジェスト的には次のような感じだ。

●NetFilter機能は、iptables v1.2.1aを使用。カーネルにはそのためのパッチを適用している。

●Proxyキャッシュは、squidを使用。お試しということで、--enable-async-ioを指定しaufs、pthreadを有効にしている。また--enable-pollも指定。ただ効果のほどは利用者が僕一人だけなので不明（ベンチマークでもやらないと…）。またデフォルトではスレッド数16が指定され、そのメモリ消費の多さと、一人で使用するレベルでは、スレッドが効果的に使われていないことに後悔する。テスト時はともかく日頃の運用はスレッド数4のものを使用。ディスクキャッシュは最大500Mバイト（aufs指定、ディレクトリ数は変更せず）、メモリキャッシュは16Mバイト、キャッシュオブジェクトは16Mバイトに調整。あと内々ネットワークをlocalnetと定義し、接続許可を与える、などを施した。

●チェックしたいソフトウェアの配布サイトをrsync、cvs、wgetなどで収集するために、とりあえず40Gバイトのディスクスペースを用意。rsyncでftp.gnu.org::ftp、rsync://rsync.kernel.org/pubなどを取り寄せてみる。gnuで2.4Gバイト、kernelで9Gバイトなど。ミラーとして公開もできないのに申し訳ないなぁ…とちょっと後悔するも、2回目以降は差分のやり取りなのでやっぱり便利。cronで12時間ごとに走らせ、結果はメールで配送。サイト全体のコピーが必要ないところは、wgetの-mオプションを使用して必要なディレクトリのみチェック。

ネットワークの調整

さて、設定はまず土台からということで、ADSLのような比較的大容量転送が可能な回線の場合のLinuxのネットワーク調整について説明しておきたいと思う（先ほどのアプリケーションレベルの設定は次回ね）。

昔から結構言われていたことなのだが、最近は特にADSLやケーブルモデムとの接続でWindows 9xのデフォルト設定では性能が出ないという話を聞く。結論から言うと、MTUのサイズが問題ということになる。これはWindows 9xのネットワークインターフェイスのMTUデフォルトサイズが従来の低速なモデムでの通信を前提に設定されているからだ。PPPインターフェイスが28.8Kbpsだったとしよう。これは、3.6Kバイト／秒である。これに対し、大昔はともかく、昨今の一般的なEthernetのMTUサイズは通常1500バイトである。10Base／100Baseのネットワークにしてみれば、このサイズは極めて小さい。しかし前述のネットワークだと、1秒間にたった3個も送れないのである。しかもネットワークは相互に通信するため、ずっと占有して送りつづけるわけにもいかない。だからPPPの実装レベルでMTUのサイズを576バイトに制限していたのだ。インターネットって上位層はTCP/IPだけど、物理層にはざまざまな選択肢があって、実際さまざまな物

理ネットワークで構成されている。そのインターネットでIPパケットが断片化されないで配送されるには、MTUが576バイトというある規格に合わせるのが適当だったということもある。

しかしWindows 9xの場合、MTU=576というのがすべてのネットワークインターフェイスに適用される設定となってしまっていた。そのためEthernetを装備してもこのMTUサイズが適用されてしまいLAN環境でも性能は芳しくなかったのである。このことがADSLの場合も如実に性能として表されてしまうというわけだ。

ではLinuxの場合はどうかというと、そんなタコな設定がなされているわけがない。MTUは、ifconfigコマンドで各インターフェイスのMTUの数値で確かめられるのでチェックしてみるとよいだろう。通常MTUを変更する必要はないが、指定したい場合は同コマンドで行えるようになっている。もちろんEthernet以外のインターフェイスでも最適値（通常は設定できる最大値）が設定されるよう実装されているので、それほど気にする必要はないと思う。

■MTUとMSSとIPフラグメント

用語の説明が後になってしまったけど、MTUと後ほど出てくるMSSについて説明しておく。

MTUというのは、最大転送ユニットの略で物理層が転送の単位とするサイズだ。通常、通信は同一物理層内で行われるので、最適値は許容される最大値ということになる。高性能な製品の中には、特別にこのサイズを大きく設定できるものもあるが、一般的なEthernetでは1500バイトだ。このユニットは、それぞれIPヘッダ、そしてTCP／UDPヘッダ、そしてデータの格納部分で構成される。

MSS（最大セグメントサイズ）は、TCPの場合のデータ格納部分の最大サイズを表している。標準的なパケットでは、IPヘッダが20バイト、TCPヘッダが20バイトなので、MSSは1460バイトとなる。後述するTCPオプションのタイムスタンプが有効な場合は、このサイズが12バイト少なくなり、1448バイトとなる。これはtcpdumpでパケットを監視した際に表示させることができるのでチェックしてみるとよいだろう。

```
test.www > linux01.3897: P 702260:703720(1460) ack
1 win 32120 (DF) (ttl 60, id 24415, len 1500)
```

TCPの場合は、このMSSサイズ、またこれを基準としたウィンドウサイズを、セッションを確立するネゴシエーションの際にやり取りするため、値に深い意味があるが、UDPの場合は、ネゴシエーションをしないので、あえて取り立たされない。

さてネットワークに流されるデータは、ネットワーク機能のレイヤを渡り歩くたびに、どんどん分割されて、最終的にMTUのMSSに区切られて回線に送出されることになる。最近のインターネットでは、MTU1500で何の問題なくルーティングされていく。これが問題になるような経路なんて探すのが大変だと思うが、そういった意味では、完全にないとは言い切れない。

MTUが1500未満の経路を通過する場合、MTUはさらに適当なサイズに分割されることになる。ここで分割されたパケットは、IPヘッダがそれぞれに付けられ、フラグメントの状態も書き込まれる。しかしTCP／UDPヘッダなどそれ以降のデータはサイズに合わせて分割される。つまりあるパケットにはTCP／UDPヘッダとそのMTUに収まるだけのデータ、その次のパケットにはそれ以降のデータ、そしてさらにそれ以降のデータという分割がなされる。このフラグメントパケットを受け取った側は断片が揃うのを待ってパケットの再構築を行う。

これを悪用したDoS攻撃があるのはよく知られていると思う。だから割り切ってIPフラグメントパケットを受け入れない設定のシステムもあったりする。実際、割り切ってしまっても問題は少ないので、ファイアウォールなどのフィルタリングで排除してしまうのだ。

で、まぁ、これに関してちょっとだけ注意しておくことがある。それはPPPoEの存在だ。ある経路間をラップして通すわけだが、その際、PPP通信のポイント間のアドレスは最低限付加される。各4バイトで8バイト分は増える。実際にはその実装次第だとは思うが、最悪の場合、パケットの分割が行われてしまう可能性がある。セッションを確立する際にMTUサイズのやり取りはないが、双方がMSSサイズを交換するので、大丈夫というわけだ。ただ、PPPoEで接続されているマシンをゲートウェイとしてその内側のマシンから利用する場合、EthernetのMTU1500のパケットがゲートウェイマシンには到着する。これを外にフォワードする際に問題となる場合があるというわけだ。でも僕自身は、PPPoEを利用していないので、本当のところどうなるのかはわからないんだけどね。チェックしてちょ。

■ウィンドウサイズと受信ウィンドウサイズの拡張

TCPは、送信した各パケットに対して受取応答（ACK）を返す手法でデータの配送を管理し、通信を信頼のあるも

のにしている。

MSSを基準に設定されるウィンドウサイズというのは、送信側が受取応答（ACK）が返ってこなくても送信するデータサイズ（数）を表している。これはセッションのネゴシエーション時にそれぞれが希望のサイズを指定するようになっている。上記の例では、32120がウィンドウサイズ、1460がMSSで、22個分のパケットデータということになる。送信側は、このサイズ分のパケットを送信した後、該当する受取応答（ACK）を待つことになる。もし既定のタイムアウトまでに返答がないようなら再送を行ったり、送信要求をキャンセルしたりする。

つまりウィンドウサイズを大きくすることでネットワークの利用効率が向上する場合があるが、それは受取応答（ACK）が何らかの理由で遅延している場合に限るのだ。

標準では、ウィンドウサイズの最大値は、TCPヘッダの該当部分の既定サイズ（2バイト分しかない）に制限され、65535となる。ウィンドウサイズはアプリケーション側で指定できるが、特に指定のない場合、Ethernetでは、MSS×12の17520がデフォルト値として使用される。

実際問題としてLANでは遅延は無視できるレベルなのだけれど、インターネットのような世界規模のネットワークだと遅延が無視できない。特に昔と比べて大容量の転送が可能になり、遅延はあるけど、大量のデータがやり取りできる状況にある。

そうなるとウィンドウサイズの最大値が65535バイトというのはちょっと小さい。

それを見越してRFC1323ではウィンドウスケールというウィンドウサイズの拡張が規定されている（一般にラージウィンドウと呼ばれる）。また同時に正確なパケットの遅延をチェックできるようタイムスタンプ機能も規定されている。

Linuxでは、これらのTCPオプション機能の有効／無効を/proc/sys/net/ipv4のtcp_Window_scaleとtcp_timestampで指定できるようになっている（デフォルトではどちらとも有効）。echoで1をリダイレクトすれば有効、0だと無効になる。

両オプション機能は、セッションのネゴシエーションの際にやり取りされ、その有効、無効が決定される。当然通信の双方が有効である場合のみ、そのセッションで利用されることになる。

```
test.ftp > linux01.2413: S (0) win 32120 <mss
1460,sackOK,timestamp 541918457 0,nop,wscale 0>
linux01.2413 >test.ftp: S (0) ack 1430485524 win
32767 <mss 1460,nop,wscale 7,nop,nop,sackOK>
```

上記はセッションハンドシェイクの最初の2つのやり取りを抜粋したものだ。

まず最初のSYN要求では、自分はタイムスタンプ、ウィンドウスケールが有効であることを接続先に伝えている。ただし自身の受信ウィンドウは拡張しないのでwscale 0とウィンドウスケール係数は0が指定されている。それに対するSYN-ACKでは、wscale 7とウィンドウスケールは有効であるものの、タイムスタンプは無効であることを伝えている。

なおウィンドウスケール係数は、標準のウィンドウサイズを左にビットシフト、つまり1なら2倍、2なら4倍する指定だ。上記の7は、128倍、ウィンドウサイズ約4Mバイトでよろしくという意味になる。

このやり取りにより、testはLinux01に対して、約4Mバイト分のデータを、該当するパケット群のACKがどれひとつとして返ってこなかったとしても送信し続けるようになるというわけだ。

■受信ウィンドウサイズを拡張する

Linuxの場合、先ほどの各オプションは有効なんだけど、実際ウィンドウスケールは0にしかならない。これは既定の受信バッファメモリサイズが次のようになっているからだ。

```
# cat /proc/sys/net/ipv4/tcp_rmem（デフォルト値）
4096    87380   174760
```

拡張してみたければ、次のような感じで2フィールド目（既定値）と3フィールド目（最大値）を指定するとよいだろう。下記だとほとんどの場合、ウィンドウスケール係数に1が設定されるハズだ。もちろん提案するウィンドウサイズとの兼ね合いなので下記の2行目のようにデフォルト値を2048バイト程度大きくすれば、ウィンドウスケール係数に1が設定されるセッションも少なくない。

```
echo 4096 174760 349520 > /proc/sys/net/ipv4/tcp_rmem
echo 4096 90000 180000 > /proc/sys/net/ipv4/tcp_rmem
```

なお、このサイズを大きくすれば必ずしもネットワークの性能効率がよくなるってものではない。遅延によるタイムアウトでのパケットの再送や送信側の待ちぼうけを軽減

できるだけで、ACKを省略したりするのとは本質的に異なるので注意してほしい。

ちなみに各種バッファサイズは次のようになっている。

```
Default UDP send buffer size... 65535
Default UDP receive buffer size... 65535
Default TCP send buffer size... 16384
Default TCP receive buffer size... 87380
```

■タイムスタンプ

タイムスタンプが有効な場合は次のようになる。

```
test.www > linux01.32782: P 778067:779515(1448) ack
506 win 31856 <nop,nop,timestamp 787002604 99704>
```

各MTUのTCPヘッダを拡張してタイムスタンプ（オプションのプレフィックス含め12バイト）が埋め込まれるため、格納できるデータサイズは1448となっている。

一応タイムスタンプを無効にすることでちょっとだけデータを追加できるようになるが、逆にタイムスタンプのお陰で効率のよい転送が行われていたような場合はかえって問題になる。設定は動的に反映されるのでそれぞれ試してみて判断するのが賢明だ。

■SACK（RFC 2018）

SACK（選択的な受取応答）は、その名のとおり受取応答を選択的に行う仕組みだ。

前述のセッションネゴシエーションのオプションにsackという表示があったと思う。これが指定されている場合、その機能は有効となっている。Linuxではデフォルトで有効だ。設定は、先ほどと同じディレクトリのtcp_sackで行える。

受取応答（ACK）は、送信されたデータパケットに対してそれぞれ返信するのが基本だ。アプリケーションによっては、ACKを1つおきのパケットやさらに省略する設定で通信する場合もある。これは信頼のおけるネットワーク、一般にはLAN環境で通信効率を向上させる目的で用いられる。たとえばWindows 2000のSMBセッションはACKの数を半分以下に軽減している。だから昔と比べると結構速い（うちの安価な構成の100Baseネットワークで平均8Mバイト／秒くらい）。

しかし基本は受け取ったパケット全てに対してACKを返すことになる。ただし送信側としては、ウィンドウサイズを再送要求の基準にしている。ウィンドウサイズ中のひとつのセグメントパケットをロストした場合でも、再送はその全体が行われるのだ。

SACKではそのような場合、全体ではなく部分要求が行える機能を提供する。

```
linux01.3897 > test.www:. [tcp sum ok] 1:1(0) ack
699340 win 64970 <nop,nop,sack sack 1 {702260:703720}
linux01.3897 > test.www: . 1:1(0) ack 699340 win 64970
<nop,nop,sack sack 2 {706640:708100}[|tcp]>
linux01.3897 >test.www: . 1:1(0) ack 699340 win 64970
<nop,nop,sack sack 2 {706640:709560}[|tcp]>
```

この通信では、経路かネットワーク機器の問題でパケットの順番がシーケンス番号のとおりに流れていない。しかしこの段階ではそのパケットをロストしたと判断するのは時期尚早だ。SACKでは、このような場合、すでに完了しているシーケンス番号を指定し、sackオプションとして受け取ったセグメントパケットを設定した受取返答を行う。

上記の場合、699340を返答の番号とし、シーケンス702260:703720を受け取ったことを知らせている。しかしその後さらにパケットの到着にずれが生じてしまい。sack 2（先ほどは1）として新たなセグメントパケットに対する返答を行っている。その後のパケットの到着によって、{706640:708100}、{706640:709560}、{706640:711020}というように、個々のセグメントパケットに対する返答というよりは、どこからどこまでという全体の状況を知らせるのだ。

これにより最終的にロストしたと判断されたセグメントパケットがあったとしても、ウィンドウサイズ分ではなく、個々のセグメントパケットに対する再送要求が可能になる。ウィンドウサイズを大きくすると、効率が向上する半面、失敗した場合のペナルティも大きくなる。SACKではそのペナルティを軽減することができる。TCP通信はストリーム的な使い方が多いのでSACKはかなり役に立つのだ。

■

ちなみにnetstat -sで再送要求が行われたセグメントパケット数が分かるので、それを確認しながら、さまざまなネットワークオプションを調整するとよいだろう。

またtcpdumpでSYN、SYN／ACKパケットを抽出してチェックする場合は、次のようなオプションをつける。HEXダンプが不要な場合は-Xはいらない。

```
# tcpdump -i eth0 -s 128 -X -vvv "tcp[13] & 2 == 2"
```

Linux日記

第20回　メール配送（7）

sendmail.cfがメールを配送するためには、適切なルールに従ってメールアドレスを書き換えたり、配送エージェントを選択しなければなりません。このルールはどのように記述されているのでしょうか。今回は、ルールセットの記述について解説します。

illustration ; Aki

文：榊 正憲
Text : Masanori Sakaki

前号の冒頭で、2001年になって筆者が最初に書いたのは、2001年4月号の原稿だと書いた。実は最初に編集氏に原稿をメールで送った時点では、これを3月号と書いていたのである。原稿受け取り確認のメールでそのことを指摘された。どうも、前回の2001年問題といい、この件といい、筆者自身の時空連続体の認識能力が低下してしまっているようだ。昔からは時間についてはいい加減な人間だったが、誤差が1カ月も出るというのはちょっと困ったものだ。

■わかりやすい時間体系

かつては会社に勤め、その後、会社は辞めたものの子供ができたということもあって、筆者は現在1日を24時間で過ごすというリズムで生活を送っている（起床／就寝時間などは世間とかなりずれているが）。しかし、ろくに学校にも行かなかった学生時代は、眠くなったら寝る、目が覚めたら起きるといったリズムで生活を送っていた（状況によっては、仕事が終わると眠れるといったこともあったが）。つまり、太陽の運行とは関係のないリズムで生活していたのである。このような生活を送っていると、だいたい1日のサイクルは30時間くらいになる（人間の体内時計のリズムは天体による1日より少し長いらしい）。そんな頃、同じような生活をしていた友人が面白いことを考えついた。新しい時間体系の構築である。1分を64秒、1時間を64分、1日を32時間とする。そして1年を256日とするのである。すると、1年の長さは3355万4432秒になる。通常の時間体系では、60秒×60分×24時間×365.25日として3155万7600秒である。1年の長さを今と同じままとすると（さすがに、寒いとか暑いといったことは、世間様と同じように感じていた。もっとも、仕事が終って家に帰るときには季節が変わっていたということもあったが）、1秒の長さは、新しい時間体系と従来の時間体系で、6%ほどしか違わないのである。つまり、体感的な1秒という間隔はほとんど今と変わらないまま、日付計算の非常に簡単な時間体系ができあがるのである。もっとも、曜日の数を8にしてしまうと、休日が減ってしまう。8曜日にして休日3日ならいいのだが。

当時、仲間うちでは、すばらしいアイデアだと賞賛されたのであるが、天体の運行に支配されていた旧人類の編集氏には「あほか！」と言われただけだった。

その頃もうひとつ仲間うちで考えていたことがある。人間の指は片手で5本、両手で10本であるが、もう少し違う進化の道を歩んで、これが片手で8本ならよかったのにということだ。そうなっていれば、現在の人類の数値体系は間違いなく8進数か16進数になっていただろう（個人的には16進数がよいと思う）。こうなっていれば、いろいろな処理の際に、10進16進変換を行わずに済んだろう（乗除算の負荷を軽視できなかった時代の話だ）。そして、100よりも256のほうが切りがいいと思っている人種が迫害されることもなか

ったろうにと思うのである。もっとも、小学生は、九九を覚える代わりにFFを覚えなければならないから、ちと辛くなるかもしれない。

■汎用ドメイン名

ある日、JPNICから手紙が来た。汎用jpドメイン名の優先登録のお知らせである。筆者は一応jpのドメイン名を1つ持っているので、汎用ドメイン名の優先登録の権利があるのだ。

今までのco.jp、ne.jp、or.jp、gr.jpなど、組織種別ごとのjpドメイン名は組織種別が第3レベルに来る形式（属性型ドメイン名という）だったが、汎用jpドメイン名は、組織名が第2レベルに来る形式である。たとえば、ascii.co.jpは属性型のjpドメインであるが、アスキーが汎用jpドメイン名を取得すると、ascii.jpという名前になる。

また、今までの属性型ドメイン名は、各組織に1つという規則があったが、汎用ドメイン名は、いくつでも取れる。部署ごとに、あるいは製品の名称ごとにドメイン名を取るといったことが可能になる。要するに、comなどのドメインと同じ形式でjpドメインが使えるようになるわけだ。

汎用jpドメイン名の最大の謳い文句は、おそらく漢字のドメイン名であろう。これは本稿の執筆時点ではまだ運用試験中で、いつ正式運用が始まるのかといったことはわからないが、とにかく「株式会社アスキー.jp」といったドメイン名が使えるようになる。

また汎用jpドメイン名は、個人でも取れる（個人で取得する際の条件はあるが）。今までのgr.jpドメイン名は2人以上のグループという規則があったが、汎用ドメイン名は1人でも取れるのである。汎用jpドメイン名を取得する条件は、申請者が日本国内に在住していることだけである。詳しくは、http://www.nic.ad.jp/を見てほしい。

さて、筆者は自分のドメイン名の優先登録権をもってはいるのであるが、優先順位は2位である。co.jpやne.jpなどで、同じ組織名を使っている場合、その組織名について、先に登録した組織のほうが優先順位が高いのである。したがって、1位の組織が申請してしまうと、残念ながら、筆者は今の組織名では汎用jpドメイン名を取れないことになる。さて、どうなることやら。

さて、sendmailの続きだ。前回は、メールアドレスを書き換えたり、メールアドレスに応じて配信エージェントを選択したりするルールの話をした。今回はルールセットの話からなのだが、その前に、ルールセットのチェックに使えるsendmailの-btオプションについて説明しよう。

■sendmail -bt

実際にルールやルールセットの動作を見るには、-btオプションを指定して、sendmailをアドレステストモードで実行すればよい。これはルールセットの評価のみを行うモードで、指定した文字列（アドレス）がルールセットによってどのように書き換えられていくかを、ルール1つごとに表示してくれる。sendmail -btと指定すると、現在のsendmail.cfを読み込んで、それに基づいてルールを適用する。自分でルールセットを記述しているのであれば、代替のsendmail.cfファイルをコマンドラインで指定することもできる。たとえば、sendmail -bt -C test.cfと指定すれば、カレントディレクトリのtest.cfファイルに記述されたルールセットに基づいて評価を行う。

-btオプションを指定してsendmailを起動すると、「>」というプロンプトを

```
$ sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 user@takobeya.com ──── ルールセット3と0でアドレスを評価する
rewrite: ruleset   3   input: user @ takobeya . com
rewrite: ruleset  96   input: user < @ takobeya . com >
rewrite: ruleset  96 returns: user < @ takobeya . com >
rewrite: ruleset   3 returns: user < @ takobeya . com >
rewrite: ruleset   0   input: user < @ takobeya . com >
rewrite: ruleset  97   input: user
rewrite: ruleset   3   input: user
rewrite: ruleset   3 returns: user
rewrite: ruleset   0   input: user
rewrite: ruleset   0 returns: $# local $: user
rewrite: ruleset  97 returns: $# local $: user
rewrite: ruleset   0 returns: $# local $: user
>
```

画面1　sendmail -btの例

Column

256倍シリーズ

アスキーでは、さまざまな分野で256倍シリーズの本を出版しているが、発売当初は、社内の営業や書籍流通の関係者から「なんで256倍なんですか？」という質問を山ほどくらったらしい。今ではそんな質問もないのだろうが、それでも256がどのような意味を持つ数字かをわかっている人間は、あまりいないのではないかと思う。

表示して入力待ちになる。ここで、ルールセットと文字列を空白で区切って入力すると、文字列に対して指定したルールセットが適用され、どのように書き換えられていくかが表示される。ルールセットは、カンマで区切って複数まとめて指定することもできる。

たとえば、“ user@takobeya.com ”という形式のアドレスを扱うMTA上で、ローカルホスト上のユーザー宛のメールアドレスをルールセット3とルールセット0で評価した場合、**画面1**のようになる。この内容は、sendmail.cf中に記述したルールセットに応じて内容が変化する。ここで使ったsendmail.cfは、WIDE CFというツール（夏ごろには説明できるかもしれない）を使って生成したものである。

最後に「$# local」という表示がある。これは、localという名前の配信エージェントが選択されたということだ。

sendmail -btにデバッグオプションを指定すると、各種付加情報も表示されるようになるので便利だ。デバッグオプションは、-dに続けて、ピリオドでつながれたカテゴリ、レベルを示す2つの数値を指定する（**画面2**）。カテゴリは0から99、レベルは0から127を指定できるので、この組み合わせは無数にある。どのように指定するとどんなデバッグ情報が表示されるかという点については、sendmailのドキュメントを参照してほしい。

```
$ sendmail -bt -d0.1          カテゴリ0、レベル1
Version 8.9.3                 バージョン番号

 Compiled with: MAP_REGEX LOG MATCHGECOS MIME7TO8 MIME8TO7 NAMED_BIND
                NETINET NETUNIX NEWDB NIS QUEUE SCANF SMTP TCPWRAPPERS
                USERDB XDEBUG
                              コンパイル時に指定されたオプションが列挙される

============ SYSTEM IDENTITY (after readcf) ============
      (short domain name) $w = mail
  (canonical domain name) $j = mail.takobeya.com
         (subdomain name) $m = takobeya.com
              (node name) $k = mail.takobeya.com
                                        予約マクロの内容
========================================================

ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
>
```

画面2　sendmailのデバッグオプションの例

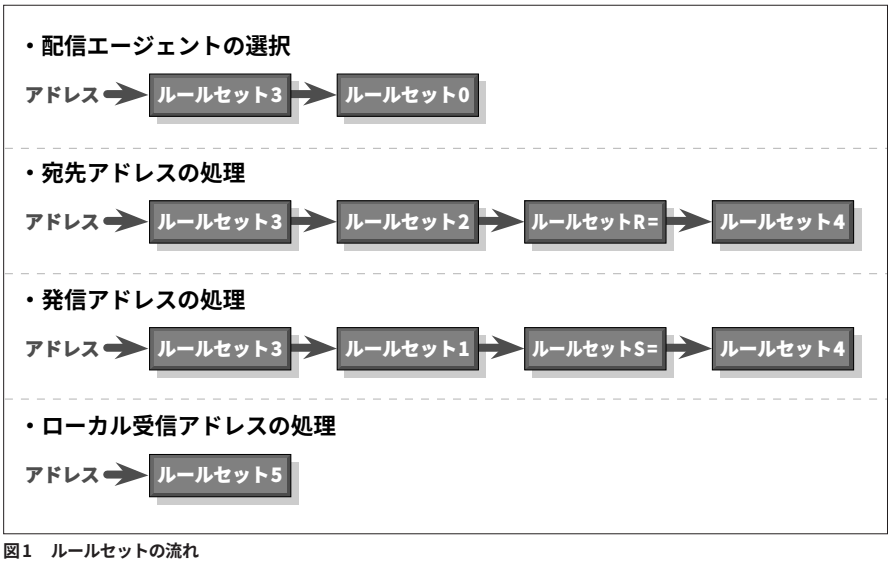


図1　ルールセットの流れ

■ルールセット

ルールセットはルールの集合体で、メールアドレスを評価し、書き換えを行ったり、配信エージェントを選択するためのマクロプログラムである。ルールセットはさまざまな用途に使われている。ルールセットの重要な仕事として、アドレスの処理と、配信エージェントの選択がある。ルールセット番号のうち、0から5は特定の機能を割り当てることが決まっており（**表1**）、6から9は将来の拡張のために予約されている。

sendmailを標準的な構成で運用する場合、ルールセットの処理の流れは**図1**のようになる。ルールセットは、メール中に指定された発信元アドレス、宛て先アドレスについて別々に適用される。基本的に、宛て先アドレスについては宛て先への配信のための処理を行い、発信元アドレスについては、組織の標準アドレス形式への書き換えといった処理を行うことになる。

図1の中で、S＝、R＝と記述されている部分は、sendmail.cf中のMコマンド（配信エージェントの定義）で指定できるルールセット番号である。

それでは、各ルールセットの役割に

番号	内容
0	配信エージェントを決定する
1	発信アドレスを処理する
2	受信アドレスを処理する
3	発信、受信アドレスの前処理
4	発信、受信アドレスの後処理
5	ローカルユーザーアドレスの処理

表1　予約ルールセット番号

ついて順に説明していこう。

■ルールセット0

ルールセット0は、宛て先メールアドレスに基づいて、どの配信エージェントを使用するのか、そしてその配信エージェントにどのようなパラメータを渡すのかを決定する重要なルールセットである。sendmailのもっとも中核となる部分ともいえるだろう。

ルールセット0は、与えられた宛て先アドレスの文字列をルールで判定しながら、必要な配信エージェントを決定する。配信エージェントが決まれば、RHSの$#オペレータを使って、Mコマンドで定義されている配信エージェントの名前を指定する。これにより、ルールセット0は、宛て先アドレスから配信エージェントを求めることができる。

基本的な考え方は、メールアドレス中のドメイン名に基づく判定である。もっとも単純な配信エージェントの選択は、次のようになるだろう。

- **メールアドレスのドメイン名が自身のドメイン名であれば、ローカル配信エージェントを選択する。**
- **メールアドレスのドメイン名がその他のドメイン名であれば、リモート配信エージェントを選択する。**

メールアドレスのドメイン名が、ローカル配信エージェントを呼び出すものかどうかは、いくつかのルールで簡単に判定することができる。たとえば、mail.ascii.co.jpというホスト上で、“ user@ascii.co.jp ”という形式のアドレスのメールをローカル処理する場合であれば、“ user@ascii.co.jp ”、“ user@mail.ascii.co.jp ”、“ user@localhost ”といったメールアドレスに対して、ローカル配信エージェントを呼び出せばいいだろう。こういったローカル配信用のドメイン名は、いくつかの予約マクロとして定義されている。たとえば、ascii.co.jpというドメイン名については$m、mail.ascii.co.jpというドメイン名については$jというマクロ参照をルール中に記述すればよい。何らかの理由で（たとえばプロバイダのMTAで、契約者のドメイン名を使ったメールサービスを運用する場合など）ほかのドメイン名についてもローカル配信を行いたいのであれば、そういった名前を直接ルールセットに記述してもよいし、適当なマクロやクラスをあらかじめsendmail.cf中で定義してもよい。

ローカル配信とSMTP配信しか行っていないMTAであれば、ローカル配信でなければSMTP配信ということになるのだが、実際にはそんなに単純ではない。宛て先に誤ったメールアドレスを記述した際に、エラー処理を行う必要があるが、このエラー処理のための配信エージェントも用意するのである。ここでいうエラーは、ネットワーク通信のトラブルではなく、たとえばメールアドレスの構文的なミスなどである。たとえば、“ user@takobeya.com ”という宛て先アドレスは正しいアドレスであるが、“ user@ ”とか“ @takobeya.com ”などは不正なアドレスである。こういったアドレスが指定された場合には、実際に配信エージェントに渡してからエラーとするよりは、事前にチェックして弾いたほうが無駄がなくていい。

こういった用途のために、特殊な配信エージェントとして、errorという名前のエラー用のエージェントが用意されている。これはSMTP配信エージェントと同じように、sendmailが最初から内蔵している配信エージェントであり、sendmail.cf中で管理者が定義するものではない。

エラーでもなく、ローカル配信でもなければ、SMTPで配信することになる。sendmailの場合、SMTP配信エージェントも内蔵されているが、errorエージェントのように予約名があるわけではない。配信エージェントを定義するMコマンド中で、各種パラメータと共に、内蔵モジュールを指定することになる。

■$#と配信エージェント

ルールセット0中のルールでは、LHSにいろいろなマッチパターンを記述し、RHSに$#オペレータを記述することに

Column

メールボックスのデータベース化

たとえば、あるメールサーバが1万人のユーザーを管轄しなければならない場合を考えてみよう。passwdファイルにこれだけのユーザーを登録するということ、メールボックスのディレクトリに1万人分のメールボックスファイルを用意することは現実的ではない。しかし、これをデータベースに置き換えたらどうだろうか？

1本のメールを1つのレコードとして扱うようにし、それをユーザー名をキーとして検索するのであれば、1万人分のメールでも簡単に扱えるようになる。プロバイダのメールサーバであれば、ユーザーに提供するのはメール機能だけである。受信したメールをデータベースに登録するローカル配信エージェントプログラムと、ユーザーにメールをダウンロードするためにデータベースにアクセスするPOPサーバを用意すれば、passwdファイルに1万人分のユーザーアカウントを登録しなくても、比較的簡単に大規模メールサーバを構築することができる。

こういったことが可能なのも、sendmailのいいところである。

よって、実際に使う配信エージェントを指定することができる。しかし、ルールセット0で求めるのは、配信エージェントの種類だけではない。

実際にメールの配信を行うには、使用する配信エージェントだけでなく、そのプログラムに渡す各種のパラメータも必要になる。配信エージェントに、宛て先となるユーザー名やメールアドレスを指示する必要があるのは明らかだ。またリモート配信であれば、送信先MTAホストのドメイン名も必要になるだろう。これは、DNSのMXレコードから得られるメールエクスチェンジャ情報から得られるホスト名かもしれないし、単にメールアドレスのドメイン部をホスト名としたものかもしれない。実際の配信では、このほかにもさまざまなパラメータが必要になる。

sendmailは、外部プログラムを配信エージェントとして使える。配信エージェントをsendmail内に持たないことによって（SMTP配信エージェントは例外だ）、実に多くのメール配信方式に対応することができる。ローカル配信をどのようにするか、モデムを使ったダイヤルアップ接続によるメール配信を、実際にどのように行うかといったことは、sendmailには関係ないのである。sendmailが持っているのは、それらの配信エージェントとのインターフェイス部分までである。このような構造になっているからこそ、sendmailはかなり設計が古いものであるにも関らず、いまだに現役でいられるのだ。

■$#オペレータ

具体的に見ていこう（ここではエラーはとりあえず無視する）。

ルールセット0でローカル配信エージェントを選択する次のようなルールがあるとしよう。

```
R$-@$m    $#local $:$1
```

LHSの$－は1個のトークンとマッチする。$mは自身の（ホスト名を持たない）ドメイン名である。つまり、ローカルサイトのユーザー名を指定されたメールアドレスが、このルールにマッチすることになる。RHSは、配信エージェントを指定する$#オペレータで、localという名前を指定している。

続けて、$：というオペレータがある。前回のRHSオペレータの説明は、$：はルールを1回だけ評価するというプレフィックスであるとしたが、この位置に記述した$：はプレフィックスではない。$#オペレータにより配信エージェント指定を行うルールでは、$：オペレータは、指定した配信エージェントに渡す宛て先アドレスを指定する。ここでは$1が指定されているので、入力文字列の1個目のトークンが宛て先アドレスとなる。

このルールは自身のドメイン名を持つ宛て先アドレスを検出するものであるから、たとえば、mail.ascii.co.jpというMTAホスト上で“masa@ascii.co.jp”という宛て先アドレスが指定されたときにマッチする。この場合、localという名前の配信エージェントが選択され、その配信エージェントには、宛て先アドレスとしてmasaという文字列が渡されることになる。

以前に説明したmail.localという配信エージェントでは、「mail.local -d masa」というように指定することで、ローカルホスト上の各ユーザーのメールボックスファイルに、標準入力から受けたメールを追加できたということを思い出そう。$：で指定した部分が、mail.localの引数に渡されることになるのである。そして、「mail.local -d masa」が実行され、masa宛のメールがメールボックスに追加されることになる。

次にローカル配信より少し複雑な、リモート配信の例を考えてみよう。次のようなルールがルールセット0中にあった場合は、どのように評価されるだろうか。

```
R$-@$+    $#smtp $@$2 $:$1@$2
```

$－は1個のトークン、$＋は1個以上のトークンにマッチする。したがって、たとえば、nobu@mail0.example.ne.jpといった宛て先アドレスにマッチすることになる（ローカルサイト宛のアドレスは、このルールが評価される前に処理されているものとする）。RHSの$#により、この宛て先アドレスを持つメールは、smtpという名前の配信エージェントで処理されることになる。配信エージェントに渡す宛て先アドレスは、$：オペレータで$1@$2と指定されている。これは実質的に、入力文字列と同じものだ。

この例では、$#と$：の間に、$@というオペレータが置かれていることに注目。$@は、LHSに置かれた場合は、0個のトークンとマッチするというオペレータになるが、RHSで$#以降に置かれた場合は、メールを送信する相手先ホストを指定するのである。この場合は、$2を指定しているので、入力文字列nobu@mail0.example.ne.jpの中のmail0.example.ne.jpという部分が指定されたことになる。これにより、メールを送信する相手がmail0.example.ne.jpと指定されるわけだ（ここではMXの話はちょっと置いておこう）。

以前に、telnetを使ってデーモンモードのsendmailと通信し、メールを送る実験を行った。このとき、telnetのコマンドラインで、接続するホスト

（実験ではlocalhostだったが）を指定する。そして、sendmailとの対話の中で、送信先のメールアドレスを指定した。$@で指定したホストは、コマンドラインで指定する接続ホスト、$：で指定した宛て先アドレスは、sendmailとの対話で指定したアドレス（エンベロープ）になるのである。以前のtelnetの例でいえば、**画面3**のようになる。

さて、ルールセット0のルールでは、$#で配信エージェントを指定し、さらにその配信エージェントのために宛て先ホスト情報、宛て先アドレス情報も指定するということがわかった。それでは、$#で参照される配信エージェントは、どのように定義されるのだろうか？

■配信エージェントの定義

配信エージェントの定義については前にも少し触れた。行頭のMは、配信エージェントの定義を意味し、それに続く文字列が配信エージェントの名前である。以降は、その配信エージェントの詳細の定義である（インデントされた行は継続行だ）。配信エージェントとして使用するプログラム、各種オプションが定義されている。

まずは、localという名前の配信エージェントの定義を見てみよう。これはsendmailを運用するうえで、必須の配信エージェントである（**リスト1**）。

P＝という式で実際に実行されているプログラム（mail.local）が定義されているのは明らかである。A＝という式は、そのプログラムを実行する際の引数を指定するものだ。mailという文字列の部分はP＝で指定されたプログラムで置き換えられる。$uは宛て先のユーザー名（宛て先アドレスではない）で、sendmailによって生成されたものである。たとえば、宛て先アドレスとして“ masa@takobeya.com ”が指定され、それがローカル配信される場合、$uはmasaとなる。

結果として、sendmailによってlocal配信エージェントが実行された場合、mail.local -d masaというプログラムが実行されることになる。以前に説明したローカル配信が行われるわけだ。

ルールセット0でリモート配信と判断され、smtp配信エージェントが呼び出される場合はどうなるのだろうか？

リスト2を見ると、P＝で指定されるプログラム名が、通常のパス名でなく、［IPC］となっている。これは、sendmailが持つSMTP配信機能を使うことを意味する。プログラムのオプションを指定するA＝では、$hが指定されている。これは、ルールセット0で判定された送信先MTAホストを示すものである。これにより、sendmailは指定されたホストに接続し、メールを配信することになる。ここでは、ユーザー名を含む宛て先アドレスは指定されていない。sendmailがSMTPで配信を行う際は、宛て先はSMTPのセッショ

```
$ telnet mail0.example.ne.jp 25
Trying 192.168.1.16...
Connected to mail0.example.ne.jp
Escape character is '^]'.
220 mail0.example.ne.jp ESMTP Sendmail 8.9.3/3.7W-2.8compat; Sat, 10
May 2001 17:28:05 +0900 (JST)
HELO mail.takobeya.com
250 mail.takobeya.com Hello mail0.example.ne.jp [192.168.1.16], pleased
to meet you
MAIL FROM: <test@takobeya.com>
250 <test@takobeya.com>... Sender ok
RCPT TO: <nobu@mail0.example.ne.jp>
250 <test@takobeya.com>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
To: nobu@mail0.example.ne.jp
From: test@takobeya.com
Subject: SMTP test

This is test mail message.
        :
        :
```

画面3　telnetでのメール送信

▼リスト1　ローカル配信エージェントの定義

```
Mlocal,   P=/usr/libexec/mail.local,  F=lsDFMAw5:/|@qrmn,  S=10, R=20/21,
          T=DNS/RFC822/X-Unix,  A=mail -d $u
```

▼リスト2　SMTP配信エージェントの定義

```
Msmtp,    P=[IPC],  F=mDFMuX,  S=31/11,  R=41/21,
          T=DNS/RFC822/SMTP,  E=\r\n,  L=990,  A=IPC $h
```

Column

Linuxのローカル配信エージェント

多くのLinuxディストリビューションでは、local配信エージェントにmail.localではなく、procmailというプログラムを採用している。そのため、ローカル配信エージェントの定義が少々異なるが、基本的な考えかたはmail.localの場合と同じだ。

ン中でやり取りされ、コマンドラインには現れない。

■配信エージェントの各種オプション

sendmail.cf中のMコマンドを見ると、P＝、A＝のほかにも、いろいろな式の指定がある（**表2**）。これらについて簡単に説明しておこう。

前述のMコマンドによる配信エージェントの指定を見てもわかるように、これらすべてを指定しなければならないわけではない。各配信エージェントについて必要なものを指定すればいい。また、はっきりいって、意味もそんなにわかっている必要もない。意味のよくわからない式については、ディストリビューションに含まれているサンプルのsendmail.cfのMコマンドを引き写してくれば、普通は問題は起こらないはずだ。

A＝とP＝は前に触れたので、ここでは、S＝、R＝、F＝だけ説明しておこう。

S＝式、R＝式は、メールアドレスの書き換えを行うルールセットの指定である。S＝式は、発信者アドレスの書き換え、R＝式は宛て先アドレスの書き換えに使うルールセットを指定する。ルールセットの指定は、1つだけ指定する場合と、スラッシュで区切って2つ指定する場合がある。リスト1に示したlocal配信エージェントでは、「S＝10, R＝20/21」という記述があった。

前に説明したように、メールのアドレス情報は、ヘッダ中に記述されたもの、エンベロープ情報としてMTA間でやり取りされるものの2種類がある。S＝式、R＝式でルールセットを1つしか記述しなかった場合は、ヘッダ中のアドレス、エンベロープ情報のアドレスとも、この同じルールセットで処理される。スラッシュで区切った2つのルールセットを指定した場合は、ヘッダ中のアドレスは後者のルールセットで、エンベロープ情報のアドレスは前者のルールセットで処理されるようになる。つまり、2種類のメールアドレスに対して別々の書き換えルールを適用できるということだ。local配信エージェントの場合、発信者のアドレスはどちらもルールセット10で処理され、エンベロープ情報の宛て先アドレスはルールセット20で、ヘッダ中の宛て先アドレスはルールセット21で処理されることになる。

F＝式は、配信エージェントを実行する際のオプションフラグを指定する。これは非常に数が多いので、いくつか抜粋して**表3**にまとめておく。詳細についてはsendmailのドキュメントを参照してほしい。

P＝式とA＝式で配信エージェントとそのパラメータを指定し、その他の式でsendmailが行うべき各種の処理を指定しているということがわかる。非常に複雑で、しかもそれをわかりにくい形で記述しなければならないのであるが、それによって得られるsendmailの自由度の高さが垣間見えたのではなかろうか？

式	名前	説明
A＝	Argv	配信エージェントプログラムに渡すコマンドライン引数を指定する
C＝	Charset	デフォルトのMIME文字セットを指定する
D＝	Directory	プログラムを実行する時のワーキングディレクトリを指定する
E＝	EOL	行の終端文字（列）を指定する
F＝	Flags	配信エージェントの各種動作オプションを指定する
L＝	Linelimit	1行の最大の長さを指定する
M＝	Maximum	メールメッセージの最大の長さを指定する
N＝	niceness	配信エージェントを実行する際のnice（プロセスの実行優先順位）値を指定する
P＝	Path	配信エージェントプログラムを指定する
R＝	Recipient	宛て先アドレスの書き換えを行うルールセットを指定する
S＝	Sender	発信アドレスの書き換えを行うルールセットを指定する
T＝	Type	DSN（Delivery Status Notification、配信状態通知）のタイプを指定する
U＝	UID	配信エージェントを実行する際のユーザーIDを指定する

表2　Mコマンドで指定するパラメータ

フラグ	説明
0	MX検索の抑止を指定する
5	エイリアスを評価した後、ルールセット5を適用する
@	ユーザー名を使って、ユーザーデータベースを検索する
A	ユーザー名について、エイリアスを評価し、展開する
b	メールメッセージの最後に空行を追加することを指定する
D	Date:ヘッダを必須とする
F	From:ヘッダを必須とする
M	Message-ID:ヘッダを必須とする
R	予約TCPポートの使用を指定する
S	指定されたUID、GIDの権限で配信エージェントを実行することを指定する
X	メールメッセージ中の行頭のドット（SMTP配信時にメッセージ終了を意味する）の書き換えを指定する
a	ESMTPを使用する
h	ホスト名の大文字／小文字情報を保存することを指定する
l	最終的な（ユーザーにメールを配信する）エージェントであることを指定する
m	複数の宛て先アドレスを同時に処理できることを指定する
w	passwdファイルのチェックを行うことを指定する
u	ユーザー名の大文字／小文字情報を保存することを指定する
x	Full-Name:ヘッダを必須とする

表3　F＝で指定するパラメータ

Linux ガベコレ

Linux Garbage Collection

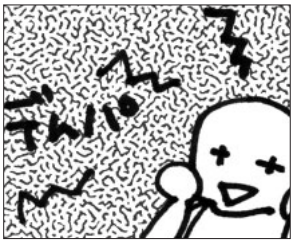
目からウロコの用語辞典

文: しのはらひろあき
Text : Hiroaki Shinohara

第13回
【セキュリティ】（せきゅりてぃ）
【マタニティ】（またに・てぃ）
【クラッカー】（くらっかー）
【ハッカー】（はっかー）
【ふせいアクセス】（ふせい・あくせす）
【アタック】（あたっく）
【ポートスキャン】

セキュリティ
【せきゅりてぃ】

「もしもし」「はい、もしもし」「Linux magazine編集部ですけれど」「えっ、こ、こんにちは。い、いまお留守にしていますが」「明らかに居留守っぽい居留守つかわないでください。いるじゃないですか」「そ、そうですね。肉体はたしかにここにあるらしいんですが、一瞬、ふっと心だけ遠くに飛んでいきたいような気がして……」「（無視）原稿どうなってますか」「原稿、原稿ですね。はい。いや、それが送りたいのはやまやまなんですが、とんでもないことになっていまして。いま、中国のほうから不正アクセス攻撃が続いているそうじゃないですか」「ああ、はい。H.U.C.ですか」「どうもあれのアタックがうちにも来ているらしいんですよ。それで原稿が進まなくて……」「アタックですか！　そりゃたいへんだ！　ポートスキャンだけですか？　それとも、もうバックドアまでしかけられましたか？」「ええ、スキャンといえばスキャンされているような。机に向かって原稿を書いていると、どうもビビビ……とどこかから走査されているような電波を感じるんです」「……」「近所に大きな鉄塔ができたんですが、あれが奴らの本拠地かもしれません。おかげで頭の中では90%できあがっている原稿が、毒電波で消去されてしまって」「……いいから原稿書け！」「は、はい、失礼しました」「ちょうどいいので、今回は“セキュリティ”をテーマにまとめてください」「セキュリティ？　ですか？」「そうです。ちなみにお茶の種類のことじゃありませんから。またいい加減なこと書いて原稿料持ってかないでくださいね」「ははは。担当さん、おもしろいこと言うなー」


毒電波受信

マタニティ
【またに・てぃ】

現在妊娠8カ月の妻は、むくみがひどくなってきたので、かかりつけの産婦人科医から塩分断ちを宣告されてきた。妊娠前からゾウのような太さだった妻の足は、むくみでゴジラの足クラスにまで成長しており、塩分断ちくらいでは手に負えない。そこで入手したのが、妊婦のむくみ止めにいいとウワサされている中国茶だ。ネットでしか手に入らないうえにやたら高い。そのぶん効果はあったらしく、ゴジラがモスラくらいにまで衰えてきた。問題はそのお茶の名前である。妊婦用だからと、よりによって「マタニ・ティー」ときた。あまりのネーミングセンスに、おもわず妻のマタにティーをぶっかけようかと思ったほどだ。「だから、お茶の話じゃない！」「はっ、見つかったか！」

おちゃ

クラッカー
【くらっかー】

もちろん、紅茶のつけあわせに最高なアレである。セレブ

なLinuxerはセキュリティーにクラッカーで優雅なアフタヌーンを過ごす、などと書いてあと84行埋めようと思ったが、最近担当の見る目が厳しいのでやめておく。ちなみに、伝説のクラッカーは「前田」さんと呼ばれる人物だという。出典は不明だが、30歳以上の世代にはもはや当たり前だの伝説とされているのでよく覚えておくこと。

ハッカー

【はっかー】

クラッカーと同義。と書いておくと、ハッカー原理主義者からクレームが来るかもしれない。それも無視して来月も同じことを書くと、ハッカー原理主義者のサイトに批判記事が載って注目されるかもしれない。注目が集まれば人気コーナー扱いになって、原稿料も上がるかも。わくわく。「妄想をふくらませるな！」「う。厳しいなぁ」

ふせいアクセス

【ふせい・あくせす】

“ふせい”なアクセスのこと。どのようにふせいなのかという点から、さまざまなバリエーションが存在する。たとえば、わが子のLinux boxのセキュリティ上の不備を心配するあまり、教育的指導の見地から厳しい態度でアクセスに臨むお父さんの「父性アクセス」。いきなり父親の書斎に呼び出され「おまえ、ホームディレクトリの下にアダルト画像溜めとくのはいいが、ほどほどにな。お母さん心配してるぞ」などと言われたらコレにあたる。また、選挙演説の遊説中、街宣カーの中で女子大生にR指定なアクセスにおよぶのが「府政アクセス」である（大阪限定）。発覚した場合には辞職の憂き目を見る可能性が高いので、初心者は注意深く行いたい。“不規則で整っていない”アクセスを指す「不整アクセス」は、このコーナーの担当編集者の会社へのアクセス時間が朝9時から夜7時までのあいだと、不規則で整っていないことを指すが、こういうことを書くと「おまえの原稿の納品のほうが不整アクセスだろうが！」などと言われかねないので触れないことにしておく。人間、自分の弱みに不正アクセスされないよう、万全の備えをしておくことが肝要だ。

アタック

【あたっく】

gooで「アッタク」を検索したら1409件ヒットした。世の中には少なくとも1409人ものうっかりさんがいるということである。可能性としては日本在住の外国人でアッタクさんという名前の方が1409人もいるということも考えられる。中国からの集中アタックも怖いが、六本木アマンド前交差点を1409人のアッタクさんが時速60kmの高速で民族大移動するほうが破壊力は高そうである。ちなみにそのままっすぐ初台方面に進むとすぐアスキー本社に当たるので（編注：当たりません）、Linux magazine編集部諸兄にあたっては避難勧告にしたがい落ち着いて退避されたい。

ポートスキャン

【ぽーとすきゃん】

スキャン中

コンピュータやルータのネットワークポートに片っ端からアクセスして、攻撃の糸口となるサービスが公開されていないか調べること。ポートスキャン自体には攻撃性はほとんどないが、攻撃の予備行為であるとして不正アクセスのひとつに位置づけるべきかどうかについて論争かまびすしい。しかし、現実世界での不法行為に照らし合わせてみれば、その解はおのずと明らかであろう。たとえば、美しい女性を見ていきなり襲いかかる。これは犯罪である。だが、知人などを介して「彼女の名前は？」といった情報を集めること。ここまでは単に恋心のなせるわざであり、否定すべきでない。声が聞きたいと思って電話をかけ、ホントに彼女が出てしまってあわてて切るようなことも何度かはあるだろう。結果的に何度も無言電話をしてしまうのもやむを得まい。気持ちがたかぶるあまり、インターネットの地図サービスで彼女の自宅周辺を表示してみるのも、許してやっていいのではないか。健康な男性なら、我慢しきれず週末や夜の空いた時間に彼女の家の近くまで訪ねていってしまうことだってあるはずだ。このあたりまで、もうまったく正常である。せっかく来たのだから、ゴミ集積所をあさり、彼女の手がかりを探し求めるのも仕方がない。ただ、ゴミを散らかしてご近所に迷惑をかけないように心がけたいものだ。などとやっているうちに、昨日の晩なにを食べたかまでわかるほど彼女について詳しくなってしまった。でも、もっと知りたい。このあたりまで、ぎりぎり許容範囲であろう。そのうち実物の彼女に近づきたくなり、毎晩女性のあとをつけて歩くようになる。そろそろ危険ライン到達である。この考察から導き出される解は、ポートスキャン行為を取り締まるべきは不正アクセス対策法ではなく、ストーカー行為防止条例だということである。

Books

「もっと知りたい」人のための　パッケージ管理&構築入門

西村めぐみ 著
ソシム
B5変形判／504ページ　　本体価格2600円

Linuxディストリビューションでは、収録するソフトウェアコンポーネントを何らかの方法で「パッケージ」にまとめている。パッケージには、お馴染みの「RPM」と「deb」、それにtar＋gzip形式でプログラムファイルをまとめたものがある。RPMとdebは、それぞれRed Hat LinuxとDebian GNU/Linux用に開発されたもので、このことからパッケージ方式によって「Red Hat系」と「Debian系」などと分類することもある。tar＋gzip方式は主にSlackware（系）で採用されている。いずれの方式にしろ、ディストリビューションを使用する際にはパッケージを扱うことになる。

本書では、各方式ごとにパッケージ管理とパッケージの作成方法を詳しく解説している。ソースからのコンパイルについても触れられているので、使用しているディストリビューションにかかわらず役に立つ情報が盛りだくさんだ。

Linux Controllerで作る簡単Linuxサーバー

太田俊哉／小田切 耕司／君島達也／中島誠一 著
秀和システム
B5変形判／240ページ／CD-ROM1枚付き　　本体価格 2800円

「Linux Controller」とは、HDE（ホライズン・デジタル・エンタープライズ）社が開発したWebベースのLinux設定ソフトウェア。ユーザー管理、ネットワーク設定、バックアップなどができるほか、Webサーバ、Sambaサーバ、メールサーバとメーリングリストまでをGUI環境で構築できてしまうという強力なツールだ。

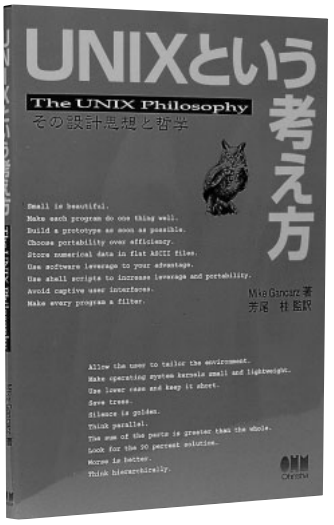
本書は、Windowsは使えるが、LinuxやUNIXにはなじみがない読者を対象に、インストールから、各種サーバアプリケーションの設定までを解説するものである。実際の操作画面を多用した構成で、解説通りに作業を進めればLinuxサーバが完成する。初めてLinuxに触れる人には心強いだろう。ただ、中級以上のLinuxerには少々物足りない内容かもしれない。なお、CD-ROMには、Red Hat Linux 6.2J、HDE Linux Controller 2.0 Standard Editionのトライアル版が収録されていて、本書の内容をすぐに試せるようになっている。

Borland JBuilder4 オフィシャルコースウェア ファウンデーション

古川正寿 著／ボーランド株式会社 監修
アスキー
B5判／312ページ／CD-ROM1枚付き　　本体価格3200円

Linuxでの開発はコマンドライン主体のチープなものが多かったが、ここへきてやっとビジュアルな開発ツールが登場するようになってきた。その中でもボーランドが昨年リリースした「JBuilder 4」は、LinuxやWindows上で動作するJavaビジュアル開発ツールとして注目されている。なにより、LinuxとWindowsで同じように開発できるというのが嬉しいところ。本誌でもJBuilderの連載を掲載しているので興味のある方も多いかもしれない。

本書は、ビジュアル開発の手法から、Javaを使ったプログラミング、コンポーネントの作成方法、スレッドなどが、初心者にもわかるように構成されている。プログラミングの経験がない場合でも、Javaのプログラムが作成できるだろう。付属CD-ROMには、Linux／Windows／Solaris版のJBuilder 4 Foundationのほか、JBuilder 4 Enterpriseの体験版などが収録されており、Linuxのビジュアル開発環境をすぐに利用することが可能だ。



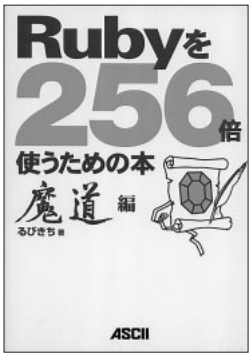
UNIXという考え方

Mike Gancarz 著／芳尾 桂 監訳
オーム社
A5判／168ページ　　本体価格1600円

コンピュータシステムには、その根底に流れている「思想」がある。それぞれの規模や用途に見合った「ものの考え方」があって、それに基づいて設計・開発・運用されている。たとえば、Windows＋x86アーキテクチャ環境のプログラマーは、そのままの発想法ではメインフレーム環境での設計・開発は行えないだろう。この例はソフトウェアだが、同じことはハードウェアについてもいえる。そして、ハードウェアとソフトウェアを結び付け、システムの中核を担うのがOSである。システムにはOS自体の「思想」が色濃く反映される。

本書は、UNIXに接する（システムを開発したり、あるいはユーザーとして利用する）際に理解しておいてほしいと著者が考える、UNIXのベースにある思想を紹介したものだ。主に開発者、特にプログラム設計に携わる人向けの内容だが、ユーザーの立場でも、コマンドを覚えるだけではわからないUNIXを理解するための考え方を身に付けるために役に立つ。

Rubyを256倍使うための本 魔道編



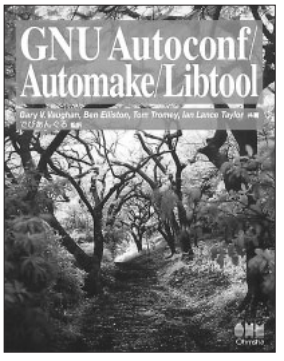
るびきち 著

アスキー

B6判／224ページ

本体価格 1200円

GNU Autoconf/Automake/Libtool



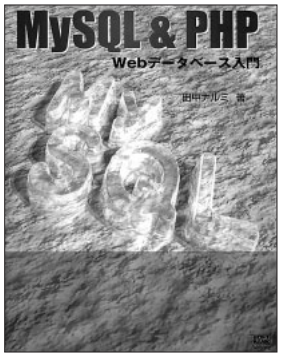
Gary V. Vaughan／
Ben Elliston ほか著

オーム社

B5変形判／424ページ

本体価格 3800円

MySQL&PHP Webデータベース入門



田中ナルミ 著

ソフトバンク
パブリッシング

B5変形判／360ページ

本体価格 2800円

Perlクックブック ーPerlの鉄人が贈るレシピ集

Tom Christiansen／
Nathan Torkington 著
田中 勝 訳

オライリー・ジャパン／
オーム社

B5変形判／816ページ

本体価格 5800円

読者の声

俺にもいわせろ！

先日、担当宅で今シーズン最後の鍋ぱーてーが執り行われました。鍋も、安くて美味しい白菜の季節も、終わってしまいました。鍋好きの担当にとっては、辛いお別れです。また、来冬に逢いましょう。そして、お花見の季節にこんにちは。今年は、どこに行こうかしらん？（笑）

魅惑の検索機能

Webで記事／CD-ROMの内容に対する細かい検索ができるといいのに……。

（はたぼさん）

㊥編集部でも記事に対する検索機能のニーズが高く、私も期待しているのですが、何しろ人手が足りず、なかなか難しいというのが現状です（泣）。データベースの構築さえ乗り切れば（というより、それがすべて）、あとはドラ○もんの世界なんですが……。

親指タイピング

携帯電話でメールを打つことに未だ慣れません……。 あのボタンでメールは……。;-)

（木村幸成さん）

㊥キーボードでの入力に慣れていると、携帯電話の入力は何ともじれったいですね。ボタンは押しやすくなっているものの、何しろ本体が小さいから入力が辛い！ 個人的には、タッチスクリーン型の携帯電話が好きでしたが、残念ながら今は販売されていません。新しい入力方式の普及が頼みの綱というところなのでしょうか？

さて、携帯電話でのメールは親指タイピングが基本ですが、担当の2倍はあるだろう親指で、携帯メールを打つ担当の父はなんとも使い心地が悪そう……世のお父さん達の中には、どんどん小さくなる携帯電話のボタンに苦労されている方が多いことでしょう。お父さん、頑張って！

バージョンアップサイクル

いつも思うのですが、最近のディストリビューションはバージョンアップの間隔が短く、ディストリビューション内の使用頻度の高いソフトなど新バージョンのものをダウンロードして、設定して、さて一通り環境が整ったところで、またディストリビューションのバージョンアップで、なんだか落ち着いて使用できていない感じがする。はっきり言って、Windowsよりもバージョンアップが激しいと思う。

（藤田善敏さん）

㊥なるほど、落ち着けないですね……そこで、提案です。ディストリビューションのバージョンアップが発表された後、すぐにインストールするのではなく、「試用感をWebとLinux magazineでじっくり調べる」という過程を追加すると程よくなるようにできている。という解釈はいかかでしょうか……だめ？

画面キャプチャ

付録CD-ROMに収録されているLinuxのインストール説明記事で インストール画面のキャプチャがありますが、これはどのようにして行っているのでしょうか。 インストールした手順の記録や他人への説明などで画面キャプチャを使用したいのですが、方法がわかりません。 よろしければ教えてください。お願いします。

（吉田光宏さん）

㊥編集部ではVMwareというソフトを使っています。VMwareを使うと、1つのウィンドウを1台の仮想マシンとして扱えます。インストール時の画面も簡単にキャプチャすることができ、とても便利なソフトです。

ただ、VMwareはお値段が少々お高いので、VMware以外の方法もご紹介します。でも、ここではヒントのみです。ちょっと考えてみてくださいね。ヒントその1．Linuxはマルチタスクです。その2．インストーラはLinuxで動いています。その3．Xのユーティリティにxwdという画面キャプチャのツールがあります。おわかりいただけましたでしょうか？

世界中のLinuxユーザーがサポートします

周辺機器との接続の規格（i-LINKなど）の現状や、Linuxでの対応と展望についてとりあげてほしい。

（栗原祐介さん）

今ある周辺機器や、これから購入する周辺機器がLinuxで使えるかどうかは、気になるところですよね。しかし、星の数ほどある周辺機器の多くは、Linuxでの動作をメーカーによって保証されていません。周辺機器ひとつひとつの動作検証については、世界中のLinuxユーザーがWeb（http://linux1394.sourceforge.net/index.htmlやhttp://www.linux-usb.org/など）で報告しあっています。有益な情報を見つけることができるかもしれません。本誌では、アバウトな企画ではありますが2000年11月号にて取り上げています。また、折をみて特集したいと思いますが、こちらもご参照くださいませ。

Linuxのお引越し

先日、ハードディスクを購入して、Linuxの引越しをしました。このときに問題となるのはLILOの再設定です。適当なインストールCDのレスキュー機能を使えばよいのですが、レスキューシステムの/dev/以下にはほとんど何も入っていません。ハードディスクをマウントするためだけにデバイスファイルを作らなければなりません。そのためにデバイスの番号を調べなくてはならず、とても面倒です。 今回は4月号付録のFisherをレスキューディスク代わりに使ってみました。すると、何ということでしょう。ハードディスクのルートパーティションが/mnt/sysimageにマウントされたではありませんか。

おお、ラッキー。 意外なものが意外なところで役に立った瞬間でした（でもFisher自体は使う予定なし♪）。

（本多洋平さん）

本多さんの環境では、たまたまうまくいったようですね。別の方法として、LILO以外のブートマネージャを試してみるのはどうでしょうか？　フリーのブートマネージャとして、GAG、GRUB、Extended-IPLなどがあります。GRUBは、OSのカーネルを直接読み込むOSブートマネージャです。また、Linuxを再構築する際に、GRUBの再インストールが不要ですので、もちろん再設定の必要もありません。今後、お引越しのご予定あるようでしたら、この機会にぜひどうぞ。

モジュールがロードされない

kernel-2.4.2をインストールしたんですが、モジュールが1個もロードされない……なぜ？

（てんさん）

モジュールがロードできない原因のひとつとして、お使いのmodutilsのバージョンが古いということが考えられます。

kernel-2.4.2のモジュールを扱うためには、modutils（この中にはinsmodやmodprobeなどがあります）のバージョンは2.4.0以上であることが必要です。それよりも古いmodutilsでは、kernel-2.4.2のモジュールを扱うことができません。modutilsのバージョンはrpm -q modutilsで確認できます。古い場合には、modutils-2.4.0以上で、かつお使いのディストリビューションに合ったものを、http://www.rpmfind.net/などで探し、バージョンアップするとよいでしょう。

疑問求む！

ここ3カ月分ほどの貴誌の付録CD-ROMを使って、貴誌の内容どおりにLinuxの機能拡張を行おうと試みました（XFree86のアップグレードや、カーネルの再構築です）。 が、ことごとく失敗に終り、元に戻せないので結局再インストール……これではWindowsと変わらないなぁ、と思いながら、悪戦苦闘してます。運が悪いのか、やり方が悪いのか……？？　はやく、いろいろできるようになりたいです。

（多胡智之さん）

誰かに説明するつもりで、自分の操作手順を書き出してみると解決につながるかもしれません。Webで検索してみるのも、ひとつの手です。どこがわからないのかを編集部に教えていただけませんか？　疑問をぶつけていただくことで、よりわかりやすい誌面にできると思います。ぜひ、ご協力ください。

スイカの塩

Linuxを始めて約3カ月。4月号付録のテスト版Red Hat Linux 7.0.9で初めて無線LAN（メルコ）が外につながりました。なんと自動認識です。今までの苦労は？　技術とは日々進歩するものなのですね。 いままで、雑誌を買いあさったりして試行錯誤のうえ、半分あきらめモードに入っていたというのに。それでも、つながることがわかり、やる気が出ました。そこで、しばらくネットを楽しんでいたら、いきなりビンゴをひきあて、あっさりと解決。一体なんだったのかと思いながらも、やっぱり嬉しかったです。

（大沼 全さん）

過去のバージョンで、試行錯誤し悩みもがくのも、新しいバージョンのすばらしさを楽しむための試練なのでしょうか？　とはいえ、できれば塩なしで楽しみたいと思う私は怠け者？

付録CD-ROMに収録した

Vine Linux 2.1のインストール

本誌付録CD-ROM収録のVine Linux 2.1はFTP（インテルアーキテクチャ）版です。非商用ソフトだけが含まれています。また、製品版を販売しているレッドハット株式会社からサポートを受けることはできません。

CD-ROMの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。

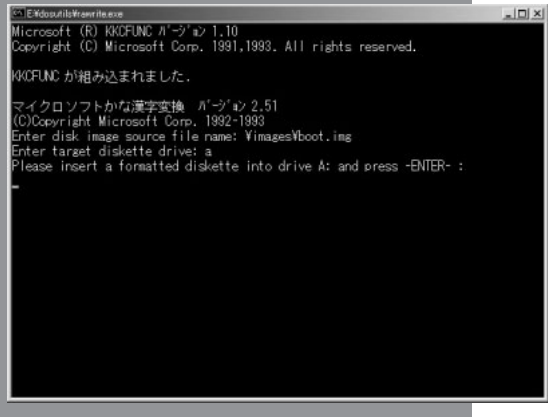
インストールの前準備

これからVine Linuxのインストールを始めますが、その途中で使用するモニタ、ビデオカード、ネットワークカードなどのハードウェア情報が必要になります。使用するハードウェアがインストーラに自動認識されない場合に備えて、ハードウェアのマニュアルを準備しておいてください。

また、インストール中に緊急時用のフロッピーディスクを作成するので、空のフロッピーディスクを1枚用意しておきます。

それではインストールを始めましょう。

1



ブート用フロッピーディスクの作成

インストールするマシンがCD-ROMから起動できる場合は、CD-ROMからブートしてインストーラを起動します。

CD-ROMから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（ここでは、フロッピーディスクドライブがA:であるとして解説します）。

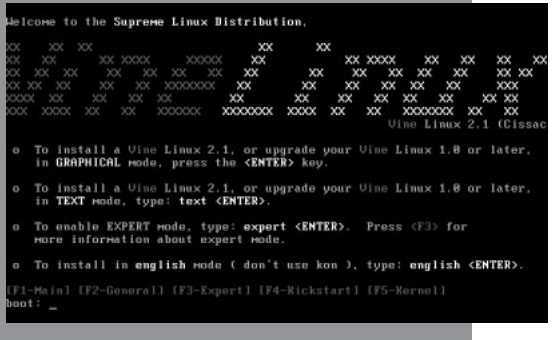
（1）Windowsのエクスプローラで、CD-ROMの「dosutils」というフォルダを開き、その中にある「rawrite」をダブルクリックします。

（2）DOS窓が開き、ファイル名の入力を促してくるので、

\images\boot.img

と正確にタイプして[Enter]を押します。さらに、フロッピードライブ名を求めてくるので、A（大文字でも小文字でもどちらでも構いません）をタイプして[Enter]を押します。最後にフロッピーがセットされているかを確認して、[Enter]を押すと、フロッピーの作成が始まります。

2



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、「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。

3

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ションを設定する」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。

インストールタイプの選択

「GNOMEワークステーション」と「全パッケージインストール」を選択すると、Windows用の領域は保存されたままで、空き領域にLinux用のパーティションが自動作成されたあとにLinuxがインストールされ、ブートローダのLILOがMBRに書き込まれます。

「サーバ」はマシンをサーバ専用機として使うためのタイプで、ハードディスク内にある既存データはすべて削除されます。

ここでは、柔軟にパーティション作成できる「カスタム」を選択して解説します。

パーティションの作成

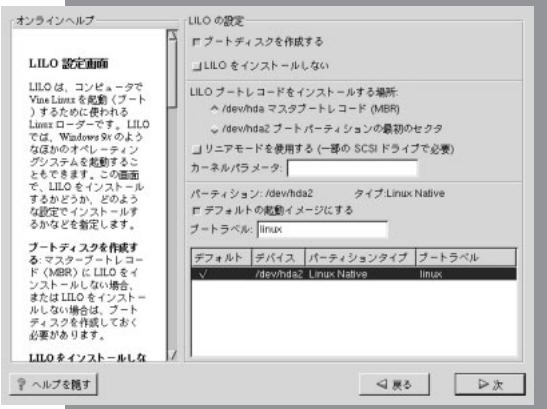
Linuxを使うには、Linuxシステム用のパーティションと、スワップ用のパーティションの2つが必要です。

画面で「追加」を押すと小さな別の画面が表示されるので、「マウントポイント」が「/」でパーティションタイプが「Linux native」の領域を2Gバイト程度、パーティションタイプが「Linux swap」の領域を、マシンが搭載しているメモリの1～2倍のサイズで作成します。たとえばメモリを128Mバイト搭載している場合は、「Linux swap」を128～256Mバイト程度作成します。

次の画面ではフォーマットするパーティションを確認して「次」を押します。

パーティションを作成したあとは、LinuxのブートローダLILOのインストール先を選択します。LILOを使ってWindows 9x／MeとLinuxをデュアルブートする場合は「/dev/hda マスターブートレコード（MBR）」を選択します。Windows NT／2000がすでにインストールされている環境では「LILOをインストールしない」をチェックして、Linuxをフロッピーから起動するのが安全です。

「ブートディスクを作成する」はチェックしたままで「次」を押します。



ネットワークの設定

家庭内LAN環境で、ISDNルータなどを使ってDHCPサーバを稼動させている場合は「DHCPを使用する」を選択します。DHCPサーバのない環境では「IPアドレス」を始めとして各項目にネットワーク情報を入力します。

さて、この「ネットワークの設定」で設定するのはネットワークカードを使って接続する場合のみで、PPPを使ったダイヤルアップ接続を設定できません。ダイヤルアップ接続する場合は本誌の特集記事を参考にして設定してください。

次にタイムゾーンを設定します。デフォルトで「Asia/Tokyo」が選択されています。マシンを日本で使うときは、このままの状態で「次」を押します。

ユーザーアカウントの設定

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。

認証の設定とパッケージグループの選択

ほとんどのユーザーはデフォルトのまま「次」を押して「パッケージグループの選択」に進んでください

次にインストールするパッケージを選択します。パッケージはおおまかなグループに分類されていますが、個々のグループの意味がわからないユーザーは、最下段の「Everything」を選択するとよいでしょう。「Everything」を選択してインストールするには、Linux用に使えるディスク領域が約1.3Gバイト以上必要です。

モニタの設定

この画面はモニタが自動検出された場合は表示されません。モニタが自動で検出されない場合はベンダーリストの中から選択します。使用するモニタがリストにない場合は、モニタのマニュアルを参考にしながら、CRTモニタユーザーは「Generic」から「Generic Multisync」の1つを、液晶モニタユーザーは「Generic LCD Panel」の1つを選択します。

この際、各項目にある周波数や解像度の数値は使用するモニタのマニュアルを参考にして選択してください。

Xの設定

多くのビデオカードは自動で検出されます（画面の例ではS3 968と認識されています）。

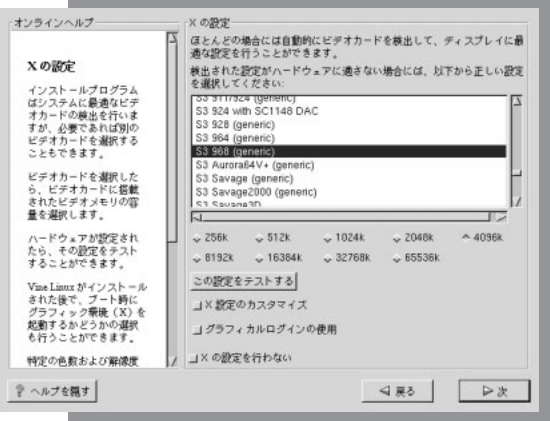
まず「この設定をテストする」を押して、Xの表示テストを行います。「このメッセージが見えますか？」と表示されたら、Xの表示テストは成功です。

デフォルトではテキスト画面でログインするようになっています。グラフィカルなログイン画面を好むユーザーは、Xをうまく表示できたときにのみ「グラフィカルログインの使用」をチェックしてください。

うまくXが表示されない場合は、「Xの設定を行わない」をチェックしてインストールを進めます。インストール後にrootでログインしコンソール上で、

```
# kon
# Xconfigurator
```

とタイプしてじっくりとXを設定するとよいでしょう。



パッケージのインストールとブートディスクの作成

手前の画面で「次」を押すとパッケージのインストールが始まります。128Mバイトのメモリを搭載したAMD K6-II/350MHzマシンで、標準的なCD-ROMドライブを使って「Everything」を選択した場合は、パッケージのインストールに10～15分程度かかります。

パッケージのインストールが終わったら、ハードディスクからLinuxを起動できなくなったときに備えて、緊急時用のフロッピーディスクを作成します。空のフロッピーをドライブにセットして「次」を押すとフロッピーの作成が始まります。

インストール終了

フロッピーディスクの作成を終えるとVine Linuxのインストールはおしまいです。フロッピーディスクをドライブから抜いて「終了」を押します（CD-ROMは自動でイジェクトされます）。

お疲れさまでした。

起動OSの選択

マシンを起動するとグラフィカルなブートローダLILOが立ち上がります。キーボードの矢印キー（↑↓）でOS名を選択し[Enter]を押すと、選択されたOSが起動します。

MBRからLILOを削除する場合は、Windows 9x／Meの起動フロッピーからマシンを起動して、fdisk /mbrというコマンドを実行します。このコマンドを実行することで、Windows 9x／Me用のブートローダがMBRに書き込まれ、マシンの起動時にLILOが立ち上がらなくなります。