ECサイト構築システムパッケージ「OneMarket PowerEC」シリーズ

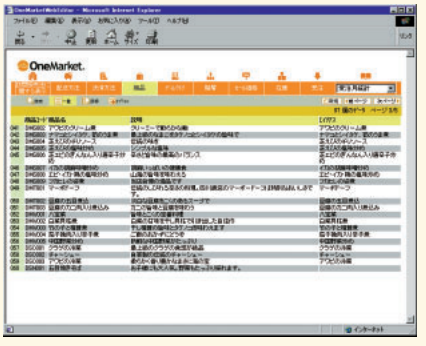
メディアヴィジョンは、ワンマーケットが開発したECサイト構築システム「OneMarket PowerEC」の販売と、同製品のASPサービスを4月23日より開始した。

今回発表されたのは、OneMarketのさまざまなオプションモジュールを選択してECサイトの構築とサーバを提供するASPサービスのOneMarket PowerEC Premium、OneMarket PowerEC Advantageと、自社管理サーバにおいて使用できるソフトウェア買い取りタイプのOneMarket PowerEC Enterprise。

同システムはデータベースエンジンとOneMarketのソフトウェアを駆使してインターネットを利用した自在な商取引を可能にしている。商品や顧客管理を担うStoreManager、遠隔地からデータベースを管理できるDatabaseManager、商品データ管理のためのUploaderの3つの管理ツールを使用してECサイトの運用が行える。リレーショナルデータベースはIBMのDB2を使用する。

価格はASPサービスの場合はサーバの使用形態によって、占有サーバ型と共有サーバ型があり、占有サーバ型のPremiumは、初期費用58万円、月額費用13万8000円から。共有サーバ型のAdvantageは、初期費用19万8000円、月額費用5万8000円から。

システムソフトウェア買い取り型のEnterpriseが380万円から。買い取り型の場合、サーバに要求される動作環境は、Pentium II／III 400MHz以上、メモリ256Mバイト以上、8Gバイト以上のHDD。OSは、Turbolinux 6.1J Server（6.5対応予定）、Red Hat 6.2J Professionalに対応。



発売日	2001年4月23日
発売	株式会社メディアヴィジョン
TEL	03-3222-4368
価格	19万8000円〜
URL	http://www.mvi.co.jp/

News Headline from 日刊アスキーLinux

www.Linux24.com

米Eazelが事業停止。「Nautilus」のゆくえは?

2001年5月17日

5月15日、Linuxのための使いやすいユーザーインターフェイスの開発を目指してきた米Eazelが事業停止を発表した。

EazelはLinuxをより簡単に操作するためのデスクトップ環境を開発すべく、1999年に設立された。創立には米Apple Computerや米America Onlineの出身者が多く加わり、設立当時は大きな注目を集めた。GNOME上で動作するファイルマネージャであるNautilusを開発しており、Ximian GNOMEでも、ファイルマネージャとして採用されていたほか、Solarisにも採用されることが決定していた。

Eazelは5月8日に「Nautilus1.0.3」を発表したばかり。CNETの報道によると、Nautilusの開発を続ける一方で2回目の資金集めに奔走していたというが、失敗してしまったようだ。16日には同社のWebサイトが閉鎖され、事業停止を知らせるメールがEazel共同創立者であるBart Decrem氏によりGNOME開発者のメーリングリストに配信されている。そのメールによると、過去6カ月にわたり資金調達を試みたが、「不幸にして、ハイテク資本市場はほとんど干上がってしまっていて、資金を確保することが不可能だった」(Bart Decrem氏のメールより)という。

NautilusはGPL(GNU General Public License)で配布されており、今後も開発を継続するかどうかが注目される。Bart Decrem氏によれば、Eazelがこれまで運営してきたサービスの一部であった、開発者のためのメーリングリストやバグレポートの収集/管理のためのサーバについては、Andy Hertzfeld氏(かつてはMac OS開発の中心的人物であり、Eazel創立当時からの従業員)がホスティングするという事になっているという。

米Easel(http://www.eazel.com/)

ボーランド、「Delphi 6」を披露

2001年5月16日

ボーランドは16日、開発者向けカンファレンス“Kylix Day”を都内で開催、あわせて報道関係者向けに、同社の開発ツール製品戦略について説明した。説明会では、米ボーランド社が8日付(現地時間)で発表したWindows向けRAD(Rapid Application Development)環境「Borland Delphi 6」のデモンストレーションが行われた。

Delphi 6は、Webサービスを容易に利用/構築するための支援機能を備えたRADツール。XMLやSOAP、WSDL、XSLなどの業界標準テクノロジーをサポートし、インターネット上でさまざまな情報の共有や交換を行うWebサービスアプリケーションを開発、配布できる。また、マイクロソフトのBizTalkや.NET、サン・マイクロシステムズのONEといったWebサービス向けプラットフォームにも対応している。

Delphi 6に搭載されている“BizSnap”はXML/SOAP対応のWebサービスを構築するためのRADプラットフォーム。“WebSnap”は、Apache、Netscape、IISといった主要なWebアプリケーションサーバに対応した、コンポーネントベースのWebアプリケーション開発のフレームワーク。“DataSnap”は、XMLやDCOM、CORBAを通じて、クライアントアプリケーションやサービスを、Oracle や SQL Server、 Informix、 IBM DB2、Sybase、InterBaseなどのデータベースと接続できる機能。

また、Delphi 6は、クロスプラットフォーム開発のためのコンポーネントライブラリー“CLX”(Component Library for Cross Platform:クリックス)を搭載する。このCLXはLinuxネイティブRAD環境「Borland Kylix」にも搭載されているもので、これにより1つのソースでWindowsとLinuxの両方で動作するアプリケーションを開発することが可能となっている。

Delphi 6は米国で6月発売予定で、Enterprise版、Professional版、Personal版の3種類が用意される。なお、Delphi 6の日本語版については、後日詳細が発表される見込みだ。

ボーランド(http://www.borland.co.jp/)

「Mozilla 0.9」公開

2001年5月10日

mozilla.orgは5月7日、「Mozilla 0.9」を公開した。リリースノートによると、このバージョンでは、以下の点が改良、追加されたという。

- プロキシ設定の自動化ツール
- “Personal Security Manager”の改善
- “MailNews”のフロントエンドの改善
- ブラウザとメールの性能向上のために改善されたキャッシュと“viewmanager”
- Javaの遅延読み込みにより、起動時のパフォーマンスが向上
- Help Viewerを新しく作成
- Image rendering libraryを性能向上のために新設計

そのほかに、「Preferences」ダイアログボックスでSSL関連の設定が可能になっているなど、細かい点で改良がなされている。

mozilla.org(http://www.mozilla.org/)
もじら組(http://www.mozilla.gr.jp/)

テンプスタッフ、インテルと協力しLinux/Javaエンジニア養成会社を設立

2001年5月10日

人材総合サービス会社のテンプスタッフは10日、ITエンジニア養成のための子会社“エクスト株式会社”を設立、インテル、イーシー・ワン、テンアートニ、ファモティクと協力して、Linux/Javaエンジニアを養成し、企業に最適な人材を提供する事業を6月1日より開始すると発表した。

エクストは、テンプスタッフに登録して

おり、アプリケーション開発の経験が2年以上ある人、もしくはITスクールなどでLinuxやJavaを勉強したことのある人を対象に、独自の選考試験を実施する。試験合格者のみがエクストのトレーニングを受けられる。未経験者は対象外となっている。

トレーニングは、LinuxとJavaにフォーカスした内容となっており、最初に集合教育を2カ月間行う。集合教育で、LinuxおよびJavaの基礎、SQLの基礎／応用、Javaの応用（J2EEのAPI、XML／XSL、EJB、FTK／WATS、フレームワーク演習など）を学んだ後、スキル評価を実施、インテルアーキテクチャ（IA）ベースのWebアプリケーション開発プロジェクトに参加させてOJT（実際の業務に携わりながらの研修）を3〜6カ月間実施する。その後、IAベースのLinux／Javaエンジニアを要望している各企業に派遣就業させる仕組みとなっている。派遣就業後も、定期的にエクストで最新技術の再教育および情報交換を実施するという。トレーニング受講者には、研修中は月額15万円、研修後派遣された場合は月額30〜40万円の給与が支給されるという。

エクストのトレーニング体制において、インテルはマーケティング支援と教育スキームの指導、技術支援、OJT受け入れを行う。イーシー・ワンはJava教育支援とフレームワークベースの教育、Javaスキル評価を担当。テンアートニはLinux教育支援とLinux技術サポート、Linuxスキル評価を担当。ファモティクはJava教育支援とフレームワークベースの教育、Javaスキル評価を担当。またテンプスタッフがマーケティング支援とプロモーション協力を行う。

エクスト（http://www.tempstaff.co.jp/ex-t/）

マイクロソフト、アプライアンスサーバ市場でLinuxに宣戦布告

2001年5月7日

マイクロソフトは5月7日、現在Linuxベースのシステムが多くのシェアを占めるアプライアンスサーバ（単機能サーバ）市場に本格参入すると発表した。

同社は、国内でPCサーバを製造するパートナー各社と協力し、Windows 2000 Serverをベースとしたアプライアンスサーバ「Microsoft Windows Powered Server Appliance」の拡販を推進するという。

Windows Powered Server Appliance は、Webサーバ「Microsoft Windows Powered Web Server Appliance」と、NAS「Microsoft Windows Powered Network Attached Storage」という2種類の製品がある。Windows Powered Web Server Applianceは、IIS（Internet Information Server）5.0をベースとし、スクリプト言語を動作させるためのフレームワークであるASP（Active Server Pages）や、ネットワーク負荷分散を行うNLB（Network Load Balancing）機能を搭載、またセキュリティ機能としてSSL（Secure Sockets Layer）やKerberos v5認証プロトコルを備えている。マルチプロセッサにも対応している。

Windows Powered Network Attached Storageは、ストレージ機能を提供するもので、Windowsファイルシステム“CIFS（Common Internet File System）”、UNIXファイルシステム“NFS（Network File System）”、NetWareファイルシステムを利用できる。RAIDディスクをサポートし、マルチプロセッサにも対応。毎秒35Mバイト以上という高速なディスクアクセスを実現するという。また、複数ストレージ間でシングルファイルシステムを利用できる“分散ファイルシステム”、ユーザー／グループ単位でディスク使用可能量を制限できる“ディスククォータサポート”を搭載する。

パートナー企業は、5月7日現在で、コンパックコンピュータ、デルコンピュータ、東芝、日本コンピューティングシステム、NEC、日本マックストア、日本IBM、日立製作所、フェイス、富士通、フリーウェイ、プロサイド、プロトン、神代（フロンティア神代）、三菱インフォメーションテクノロジー、ロジテックの16社。

マイクロソフト（http://www.microsoft.com/japan/）

TurbolinuxとLinuxcareの合併が解消!?

2001年5月2日

2月21日に、米TurbolinuxによるLinuxcareの合併が最終合意に達したというニュースが伝えられたが、5月1日、この合併が解消されたというニュースを米メディアLinuxGramが報道した。

同サイトによると、LinuxcareのCEO、Art Tyde氏は、この合併の失敗により、従業員の3割を削減することになるだろうと語っている。また、Tyde氏との共同創立者であるDavid Sifry氏とDavid LaDuke氏、Sambaの開発者として著名なAndrew Tridgell氏が会社を去ったという。一方、米TurbolinuxのCEO、Paul Thomas氏は、経済の停滞により、サービスやコンサルティングが減ってきており、合併によって急成長するという当初の理由が意味を持たなくなってしまったと語っているとのことだ。

合併解消の背後には、Linuxcareの業績悪化、Turbolinuxの事業方針の転換など、資金や運営の面で複雑な状況が絡んでいるとみられる。しかし、両社は今後とも戦略的な提携関係を続けていくとのことだ。

LinuxGram（http://www.linuxgram.com/）
米TurboLinux（http://www.turbolinux.com/）
米Linuxcare（http://www.linuxcare.com/）

ミラクル・リナックス、パートナー支援プログラム「With Miracle」を開始

2001年4月27日

ミラクル・リナックスは、Miracle LinuxとOracle for Linux製品の普及と、Linux市場の拡大を目的としたパートナー支援プログラム「With Miracle」を26日より開始、参加企業を募集している。

パートナー企業は、5月21日より公開されるWith Miracleページより、ソリューション情報の登録や検索を行える。また、パートナー専用ページより、製品情報や技術資料などを利用できる。登録はWebサイトより行える。

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

Hardware　発売日　2001年4月15日

Ultrium1規格のストレージ製品 hp surestore ultrium 215

URL　http://www.jpn.hp.com/go/surestore/

発売　日本ヒューレット・パッカード株式会社
TEL　03-3335-8333
価格　69万円〜

日本ヒューレット・パッカードは、バックアップ製品群「hp surestore」シリーズで、Ultrium1規格を採用した新製品「hp surestore ultrium 215」を4月15日より発売した。

hp surestore ultrium 215は最大容量200Gバイト、データ転送速度最大15Mバイト／秒（圧縮時）、7.5Mバイト／秒（非圧縮時）。HP、IBM、Seagateの3社が共同開発したオープン規格Ultriumを採用。薄型リールモーターで、電子回路を小型化、背面に配置することにより、既存製品、同230の約半分の高さ5.25インチ、ハーフハイトの筐体に仕上げた。

内蔵型215iと外付型215eがある。標準添付のバックアップ専用ソフトウェアTapeWare 6.2はRed Hat Linux 5.2以降、Turbolinux 4.0以降の英語環境にのみ対応。付属のドライバを用いて、tar、mt等のコマンドによるバックアップも可能。

Hardware　発売日　2001年4月18日

Midori Linux採用インターネットアプライアンス FLORA-ie 55mi

URL　http://www.hitachi.co.jp/pc/

発売　株式会社日立製作所
TEL　0120-8856-50
価格　9万7800円〜

日立製作所、情報・通信プラットフォームグループは、Midori Linuxと省電力Crusoeプロセッサを採用したインターネットアプライアンス「FLORA-ie 55mi」に、無線LAN内蔵タイプを新規開発し、4月18日から発売した。

本製品は、IEEE802.11b準拠の高速11Mbps無線LANシステムを本体に内蔵することにより、オフィスや家庭内でケーブルなしで電子メールやインターネットを利用できる。ホコリや水に耐える防塵・防滴対応。入力は、ペン入力による画面でのワンタッチ操作を始めとして、キーボード、マウスなどの各種操作機能を備えている。

主な仕様はCPU TM3200（400MHz）、メモリ64Mバイト（最大192Mバイト）、コンパクトフラッシュメモリ64Mバイト（最大128Mバイト）。10.4型TFT（800×600ピクセル）、内蔵無線LAN。上記の価格は通信・バッテリー無し、コンパクトフラッシュ48Mバイトで1ロット500台発注した場合の参考価格。

Hardware　発売日　2001年4月25日

8万9000円のエントリー・サーバを新発売 PowerEdge SC

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　8万9000円〜

デルは4月25日より同社サーバ製品「PowerEdge」シリーズの最小構成で低価格化したサーバ製品「PowerEdge 300SC」（写真）、「PowerEdge 1400SC」、「PowerEdge 2500SC」を発売した。

価格は300SCが8万9000円、1400SCが11万8000円、2500SCが24万8000円。

主な仕様は、300SCがCPU Pentium III 800MHz、メモリ64Mバイト、10Gバイト IDE HDD、48倍速CD-ROM、100Base-TXネットワーク。1400SCは、CPU Pentium III 866MHz、メモリ128Mバイト、9Gバイト Ultra3 SCSI HDD、48倍速CD-ROM、100Base-TXネットワーク。2500SCは、CPU Pentium III 933MHz、24倍速CD-ROMのほかは、最小構成は1400SCと同じだが、ケースが異なり1400SCより拡張性が高い。300SCと1400SCはケースが同じで、マザーボードが異なる。注文生産も受け付けている。

Hardware　発売日　2001年4月24日

LASER5 Linux搭載キャッシュサーバ TG／iSV-CacheServer

URL　http://www.trustguard.co.jp/

発売　トラストガード株式会社
TEL　03-3258-7111
価格　160万円（4サーバ）〜

トラストガードは同社「TrustGuard／iSV」のハードウェアに、レーザーファイブが特別にチューニングした「LASER5 Linux Cache Server Edition」を組み合わせたキャッシュアプライアンスサーバを4月24日より発売した。

4Uスペースに最大13ノードのキャッシュサーバを集積。最大10Gバイトの主記憶、0.26Tバイトのキャッシングストレージ容量を持ち、毎秒2000リクエストまでのWebサービスに対応している。Webサーバを増強しなくても、コンテンツをキャッシュサーバに蓄積してサーバの負荷を軽減する。サーバの設定はWebブラウザ上からできる。IPアドレス等はコンソールから設定する。

主な仕様はサーバカートリッジ1基当りCPU Mobile PentiumIII 500MHz、メモリ256MHz（最大768Mバイト）、20Gバイト IDE HDD、100Base-TX×2。OSのLASER5 Linux Cache Editionは、LASER5 Linux 6.4をベースにカスタマイズされたもの。

Hardware

発売日 2001年5月1日

Webホスティング限定アプライアンスサーバ hp web hosting server appliance

URL http://www.jpn.hp.com/go/serverappliance/

発売 日本ヒューレット・パッカード株式会社
TEL 03-5344-7181
価格 19万9000円～

日本ヒューレット・パッカードは、Webホスティング機能に特化したアプライアンスサーバ「hp web hosting server appliance」2モデルを5月1日より受注開始、5月中旬より出荷を開始した。

1Uラックマウントサイズの筐体でスタンダードモデルのsa1100とパフォーマンスモデルのsa1120がある。sa1100がCPU Celeron 533MHz、メモリ128Mバイト、10GバイトIDE HDDで19万9000円。sa1120がCPU Pentium III 750MHz、メモリ256Mバイト、9GバイトSCSI HDDで33万9000円。OSはいずれもRed Hat Linuxベースの専用OSを使用している。Apache WebサーバやSendmailが工場でプリセットされているため、簡単な設定だけで運用できる。管理はWebブラウザを利用して行い、セキュリティのためにSSL処理機能を搭載している。OS、アプリケーションソフトからBIOSまで、定期的に更新用アップデータが提供される。バーチャル・ドメインの設定も可能。

Hardware

発売日 2001年5月下旬

オールインワンインターネットサーバ ARIES

URL http://www.duaxes.com/

発売 デュアキシズ株式会社
TEL 03-3523-6933
価格 オープンプライス

デュアキシズは、小型・軽量な統合インターネットサーバ「ARIES」を5月下旬より発売した。販社およびシステムインテグレータを通じてのルート販売のほかOEM供給も可能。価格はオープンプライス。

製品は重さ1kgを切り、サイズ172mm（H）×109mm（W）×147mm（D）の筐体に液晶グラフィカルパネル（128×64ピクセル）、100Base-TXネットワーク、11MbpsのワイヤレスLANをサポートしたPCMCIAスロット2基を搭載。シリアル、VGA、キーボード／マウス、ヘッドフォン／LINE IN／マイク、赤外線ポートを備えており、オーディオスピーカも内蔵している。CPUにMediaGXm（200MHz）を使用し、メモリ64Mバイト、10GバイトのIDE HDDを搭載している。

OSにはLinuxを採用し、ファイル共有、プリンタ共有、メール、Webサーバ、各種ネットワーキング機能を持つ。クライアントはWindows 95／98／2000、Macintosh、UNIX、Linuxに対応。各種設定は本体正面の液晶パネルからとリモートアクセスで可能。

Hardware

発売日 2001年6月

Linuxマイクロ・アプライアンスサーバ FutureNet MA-300

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

発売 センチュリー・システムズ株式会社
TEL 0422-37-8911
価格 10万円（評価キット）

センチュリー・システムズはLinuxマイクロ・アプライアンスサーバ「FutureNet MA-300」を発表し、OEM向けに2001年6月よりサンプル出荷を開始した。

本製品は同社FutureNet ES1の上位機種にあたり、ES1の特徴である2.5インチのハードディスクとほぼ同じ70mm（W）×105mm（D）というサイズとOSとしてLinux（カーネル2.4.X）を採用する仕様はそのままにして、高速化と拡張性を高めた。

評価キットの主な仕様は、CPUにPowerPC 405GP 200MHz（最大266MHz）、メモリ64Mバイト（最大128Mバイト）、RS-232ポート、PCカード・インターフェイス、5GバイトHDDを搭載している。100Base-TXイーサネットを2ポート標準装備しており、ブロードバンド・インターネットに対応のルータとして利用できる。さらにオプションとして、光イーサネット（100Base-FX）、USB、PCカード等の拡張インターフェイスも用意されている。評価キットの価格は10万円。OEM価格については別途見積もりが必要。

Software

発売日 2001年4月17日

機能およびサポートを強化 駅すぱあと　イントラネット

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

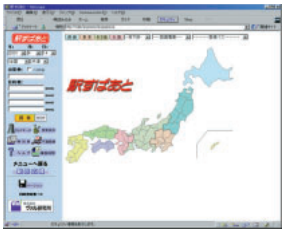
発売 株式会社ヴァル研究所
TEL 03-5373-3511
価格 30万円（同時アクセス10ユーザー）～

ヴァル研究所は機能及びサポートを強化した「駅すぱあと　イントラネット」を4月17日発売した。

これまで年間サポートとして年6回最新版を提供してきたが、今回より年12回に倍増した。

路線バスに小田急バスと神戸市営バスが追加され、路線バスの規模は公営6社、民営8社になった。新機能として定期券使用時の差額計算機能、乗り換え時間調整機能とともにJRグループおよび大手私鉄の春のダイヤ改正に対応した。10ユーザー／25ユーザー／50ユーザー／100ユーザー版の4種類を用意している。CGIインターフェイスを公開しているのでWebアプリケーションを構築できる。

サーバ環境としてはTurbolinux（6.1以降）に加え、今回Red Hat Linux（7.0以降）にも対応した。



Software　発売日　2001年4月24日

マルチプラットフォームXMLエンジン iPEX Version2.1

URL　http://www.infoteria.com/

発売	インフォテリア株式会社
TEL	03-5718-1250
価格	20万円〜

インフォテリアは、新しくSolaris、LinuxおよびWindowsに対応したXML処理エンジン「iPEX」のVersion2.1を4月24日より発売した。

iPEXは、アプリケーションソフトでXMLを扱うのに必要な機能を網羅したXMLエンジンで、開発に必要な各種機能をライブラリで提供している。使用形態に合わせて「デベロッパー・エディション」、「スタンダード・エディション」、「プロフェッショナル・エディション（PE）」の3つに別れる。

開発用は「デベロッパー・エディション」として上記Webサイトより無償ダウンロードできる。社内システムでの運用は「スタンダード・エディション」、iPEXを組み込んだシステムを第三者に販売する場合には「プロフェッショナル・エディション（PE）」が必要になる。PEには100／1000／無制限ライセンスがある。「デベロッパー・エディション」以外は、パッケージでの販売となる。

XMLパーシング、XSLTによるXMLインスタンスの変換、DOM／ネームスペースサポート、インターネット経由でのXMLデータへのアクセス機能がある。

Software　発売日　2001年5月17日

モバイル・データベース・ソフト SQL ANYWHERE STUDIO for Linux VERSION 7.0.2

URL　http://www.sybase.co.jp/product/sas_v7/

発売	サイベース株式会社
TEL	03-3512-5270
価格	6万9000円〜

サイベースは米アイエニウェア・ソリューションズ社の主力製品である、モバイル・データベース。ソフト「SQL ANYWHERE STUDIO for Linux VERSION 7.0.2」を5月17日から発売した。価格は基本パッケージが6万9000円。

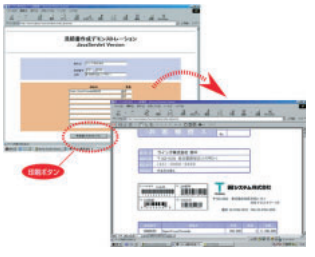
本製品によって、モバイル・データベースソフト「Adaptive Server Anywhere」をサイベース社の小型データベース「Ultra Light」と同期させる（Mobile Link）ことができる。管理ツール（SybaseCentral）は対話型SQLツール（Interactive SQL）がJavaに対応しており、これによってWindows版のツールと同じ操作環境を実現している。今回の製品では「Red Hat Linux 6.2」、「Red Hat Linux 7.0」に加えて「Turbolinux 6.1」にも対応している。

Software　発売日　2001年5月24日

帳票開発ツール「Super Visual Formade」他

URL　http://www.tsubasa-tool.com/

発売	翼システム株式会社
TEL	03-5766-2833
価格	70万円〜



翼システムは、帳票開発ツール「Super Visual Formade」と関連する帳票開発支援製品の一斉バージョンアップを行い、5月24日より発売した。

今回Javaを採用することで、Super Visual Formade for Web／PDF Java EditionとUniversal Connect／XがLinuxに対応した製品となった。今後、Report Directorをはじめとする全製品が年内をめどにJava化を予定している。

これらの製品群は帳票のグラフィカルな作成を可能にする（Super Visual Formade）と同時に、プリンタ側の持っている帳票の穴あけやステイプル処理、ソート処理、仕分けなどの操作をアプリケーション側から監視、制御する（Report Director）ことを目指している。さらに、XMLデータから自動的に帳票を作成し、それをメール配信やFAX配信させる（Universal Connect／X）ことなどにも対応した。

Software　発売日　2001年5月30日

Linuxサーバ管理ツール HDE Linux Controller 2.4 Standard Edition

URL　http://www.hde.co.jp/lc/

発売	株式会社ホライズン・デジタル・エンタープライズ
TEL	03-5738-5410
価格	2万9800円〜

ホライズン・デジタル・エンタープライズは、Linuxサーバをクライアントマシンの Webブラウザ上から設定・管理するアプリケーションの新バージョン「HDE Linux Controller 2.4 Standard Edition」を5月30日より発売した。

Webブラウザからセットアップウィザードにより、特別な知識なしにサーバ構築が可能。オールインワンサーバ、インターネットサーバ、イントラネットサーバ／ファイルサーバの中からシナリオを選択して対話形式でサーバ構築ができる。GUIによる操作なので直感的にサーバ管理ができ、従来の各種設定ファイルの編集などの煩雑な作業がいらない。今回、最新カーネル2.4に対応した。対応するLinuxディストリビューションは、Red Hat Linux 6.2／7.0／7.1、Turbolinux Server 日本語版 6.1／6.5。他のディストリビューションも検証しだい追加する予定。電話／メールによる90日間3インシデントまでのサポートがついている。

速報!

64ビットCPU Itanium ついに正式リリース

URL　http://www.intel.com/itanium/

インテルは5月29日、Itaniumプロセッサの正式リリースを発表した。これまでItaniumは、NDA（秘密保持契約）を結んだ限定された顧客にのみ提供されていた（パイロットリリース）。今回の正式リリースを受けて、PCメーカー各社からItanium搭載機が発表される予定（**表1**）。

Itaniumは、インテルとヒューレットパッカード（HP）が共同開発した64ビットCPUである。“ Merced ”というコードネームで90年代半ばから開発が続けられていた。Itaniumのアーキテクチャは、“ IA-64 ”と名付けられている。IA-64はインテルのx86（IA-32）命令とは互換性がないが、ItaniumはIA-32命令も実行できるように設計されている。

Itaniumに対応するチップセットとしては、インテルから460GXが提供される。460GXは、11個のチップで構成されており、1〜4個のItaniumをサポートする。メモリは通常のSDRAM（ECC付きPC133）を使用する。

■IA-64

IA-64は、EPIC（Explicitly Parallel Instruction Computing）というデザインを採用している。EPICはVLIW（Very Long Instruction Word）に似ているが、VLIWの持つ問題点を解決したアーキテクチャだ。名前のとおり、多くの命令を並列実行可能になっている。Itaniumは、9個の命令実行ユニットを持ち、最大6命令を同時に実行できる。

いわゆるRISC（Reduced Instruction Set Computer）形式のCPUでも命令の並列実行は可能だ。RISCの場合はプログラム実行時に命令依存関係を判断していくのに対し、EPICではプログラムのコンパイル時に同時実行できる命令群を抽出するのが特徴だ。

■対応OS

Itaniumに対応するOSの開発は、当初Linuxが先行していて、マイクロソフトのWindowsはItaniumリリース時に間に合わないと言われていた。しかし、先月になって急遽サーバ版（Windows Advanced Server Limited Edition）またはワークステーション版（Windows XP 64-bit Edition）がItanium搭載機にバンドルされることが決定した。これらはどちらもベータ版という扱いだが、マイクロソフトによるサポートが提供されるうえに、正式版リリース時には無償でアップグレードできる予定。

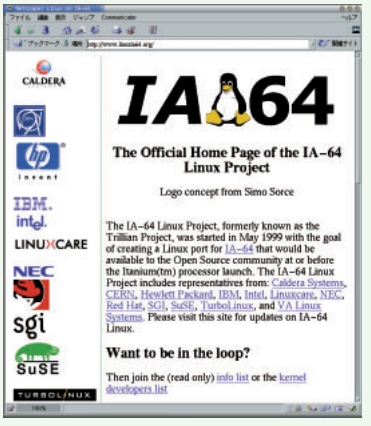
Linuxについては、Caldera、Red Hat、SuSE、TurbolinuxからIA-64に対応したディストリビューションがリリースされている（**表2**）。

■搭載機種のリリーススケジュール

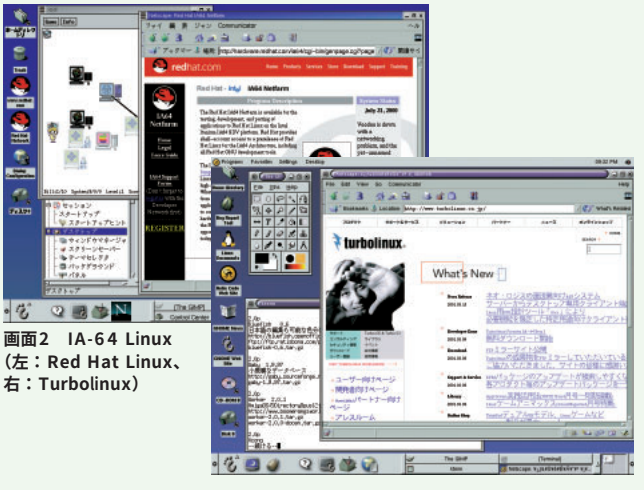
PCメーカー各社から、サーバ機を中心にItanium搭載機のリリースが予定されている。コンパックコンピュータ、デルコンピュータからは、CPUを最大4基まで搭載可能なサーバ、三菱電機からはCPUを最大2基搭載可能なサーバが発表されている。そのほか正式にリリースされてはいないが、NECや日立は8〜16基のItaniumを搭載可能なハイエンドサーバをLinuxWorldで展示した実績があり、いずれ何らかのアナウンスがあると思われる。

■ターゲットはハイエンド市場

ハイエンドのサーバ／ワークステーション市場は、サン・マイクロシステムズの製品が大きなシェアを持っている。Itanium搭載機は、低価格と強力な浮動小数点演算性能を武器に、この市場に投入されていく。



画面1　IA-64版Linuxプロジェクトの Webサイト



画面2　IA-64 Linux（左：Red Hat Linux、右：Turbolinux）

コンパックコンピュータ	ProLiant（型番未定）（7Uラックマウント型）
デルコンピュータ	PowerEdge 7150（7Uラックマウント型）
三菱電機	FT8000 モデル20Ia（フロアスタンドタイプ） FT8000 モデル25Ia（4Uラックマウントタイプ）

表1　各社からリリース予定のItanium搭載機

AIX-5L（IBM）
HP-UX（ヒューレットパッカード）
Windows Advanced Server Limited Edition（Microsoft）
Windows XP 64-bit Edition（Microsoft）
Linux ディストリビューション
Caldera
Red Hat
SuSE
Turbolinux

表2　Itaniumに対応するOS

日刊アスキー Linux Business Report

マイクロソフトが捉えたLinuxの姿

http://www.linux24.com/

最近話題になったのが、マイクロソフトが発表したLinuxに関する調査書類だ。内容は、マルチプロセッサへの対応などにおいて、Linuxと比較してWindowsが優位にあるというものだ。また、米マイクロソフト社の幹部も、ここにきて盛んにLinuxに対して攻撃とも取れる発言を繰り返している。しかし、マイクロソフトがLinuxについて言及したというのは、Linuxの存在を無視できなくなったということにほかならない。

そこで今回は、マイクロソフト製品マーケティング本部Windows製品部サーバーグループ・グループプロダクトマネージャーの松井洋一氏と製品マーケティング本部Windows製品部サーバーグループ・シニアプロダクトマネージャーの藤本浩司氏に、Linuxをどう捉えているか聞いた。

――先日の発表会では、Linuxに対しての言及がありましたが、Linuxをどう認識されていますか？

藤本氏：我々はLinuxを否定しているわけではなくて、新しいビジネスモデルとしてLinuxが育ってくるのであれば、勉強させていただこうというのが基本的なスタンスです。Windowsをご提供させていただいている立場からすると、我々は責任の所在を明確にさせていただいているOSであるということです。我々はお客様のお手元まで届けて、責任を持ってサポートさせていただいて、それによってお金をいただいているOSである、というところですね。Linuxに関しては、現時点であまり強く何かを思っているわけではないのですが、ビジネスモデルとして確立してないという点と、ディストリビューションのパッケージがいろいろとあったり、各ハードウェアメーカーから提供されるLinuxがあったりと、メーカーの思惑に振り回されないように、まとめていくことが大変そうだと思っています。

――責任の所在という点においては、Linuxはサポートをディストリビューターが行う形になっていますが、その点はどうでしょうか。

松井氏：そうですね、「Linux」と非常に大きな意味で捉えられている存在と、いまおっしゃったような、ディストリビューションとしての形態の「Linux」というものの2つについて、お客様がまだ正確なメッセージを受け取られてない部分があるのかなと思います。そして、どうしても単純な比較論みたいな話に入ってしまうというところがありますよね。1社1社のディストリビューションに対しては、各ディストリビューターが責任を持ち、我々と同じようにパッケージとして売っていますので、「Linux＝無償」という大前提はそもそも成り立たないわけです。みなさん我々と同じようなサポートの料金はとられているわけですから。

Linuxのコスト

――コストについてですが、先日の発表会では、WindowsのほうがLinuxよりもさまざまな面で結局コストがかからない、という話がありましたが。

松井氏：TCOと表現しますが、システムにかかるコストをトータルで考えてみていただきたいという話でした。さらに初期導入コストも低減するシステム提供方法として、Server Appliance Kitというものを、単機能マーケット向けに作らせていただいています。Linuxであれば、サーバアプライアンス製品を作るにしても、全部自社で作らねばならない。また、たとえばバックアップユーティリティは「フリー＝無償」ではなかったりしますし。Server Appliance Kitでは、DFSの機能ですとか、さまざまな管理ツールなどのほとんどをWindowsが持っていますので、それを呼び出してくるだけでいい。我々は、フリーといわれているLinuxに対して、Server Appliance Kitの提供というところでメリットを出させていただいている。これは「作る」立場の方々に対してメリットがあります。あとはサポートの面として、問題が起こったときに、問い合わせ窓口を明確にしておきたい、という希望があると思いますので、マイクロソフトとメーカーとできちんと受けていきます。こうしたところにWindowsのメリットがあると思っています。Linuxの技術がわかるという人がいてアプリケーションの開発を始めたけれども、やっぱりサポートにも人が必要になるし、ほかにも必要になる。マニュアルを書くにもLinuxがわかる人が要る……。そうしてどんどん経営資源が圧迫されて

しまうといったような、さまざまなご意見もいただいています。

――OSに対して対価をもらうのではなく、サービスに対して、あるいはエンジニアのスキルに対して対価をもらうというのが、Linuxのスタンスかと思うのですが、でも結局それではお金がかかってしまうということですか？

藤本氏：いや、サービスでやっていきましょうというのは、我々と一緒です。そうするとOSの対価などは微々たるものですよね。スタンスは一緒ですよ。そこで、我々はOSを作っている会社なので、OSを作るお金をいただいています。それがLinuxという風潮の中で、「サービスもタダ」という見方をされている人は多いと思うのです。そういった風潮をまずLinuxさんのほうでちゃんと打破してというか、「まずこういったビジネスモデルなんですよ」ということをお客様のほうに明確に伝えるというのも必要なのではないかなと思います。我々も基本的にサービスの会社なので、サポート、サービス料金をいただいていますが、Service Packやパッチモジュールはダウンロードで提供していますので、最終的には、うちのほうがリーズナブルと思っているんです。そしてサービスというところで考えてもらうと、無償ではないところが加わってくる。単純にいうと、Linuxの場合は、「サービスとしてのお金はちゃんとかかることを認識してください」というのを飛ばして、すべてが「無償」というところに走っているところが、「たいへんだなあ」と思っています。我々は逆にいうと、「タダじゃない」。我々はお客様の所の技術者がいなくなっても、引き続きサポートをできる体制等を持っていますので、サービス論に持っていっていただけたら、本当に嬉しいです。

マイクロソフトから見たLinuxの利点

――逆に、Linuxのいい点というのはありますか？

藤本氏：みなさんが持ち上げているところではないでしょうか。今ブームですよね。昔UNIXが一時オープンでブームになったように「全部これだ！」と。このブームというのは、我々が、LinuxとWindowsで同じシステムを構築しても、Linuxで構築したほうのみが話題になりますので、「いいなー」と思います。また、無償だということで、短期的なメリットはありますよね。ですが、「それではランニングコストはどうするの」と。今は目新しさによる注目度というのはありますよね。短期集中で今稼ぐんだということであれば、今の注目度というのをうまく利用することはできると思いますけれども、将来的にどうするのかというと疑問が出てきます。そういう観点から見ると、我々はTCOを考慮して企業規模の大小に関わらずさまざまな企業の方に利用していただけるプラットフォームを提供することでビジネスチャンスを提供してきましたし、これからも提供していきます。今までもこれからも変わらないことは、お客様の資産を継承するためのサポートや、サービスを提供しているという自負のもとで、OSの対価をもらっておりますので、無償といったところを考えているわけではありません。ソースを見たいといった要望がありましたので、特定分野のパートナー様においてはソースを閲覧可能にさせていただいています。しかしながら、お客様のほうで20万行のソースを把握するために、新たに人件費をかけて管理されるのであれば、我々にやらせていただいたほうがいいと思います。

Linux vs Windows

――風当たりは強いですか？

藤本氏：いや、そんなには強くはないですよ。ただ、（メディアが）「対Linux」と書くとヒット率が上がるのではないですか。「対Linux」と書くと、なんといいますか、オープンソースのグループを敵に回しているみたいなイメージというのがある。オープンソースの方々はがんばっているし、彼らがやっていることに対してどうこう思っているわけではないのです。やはり、Linux vs Windowsみたいな図式になってしまうところもありますが、オープンソースはオープンソースの良さがあると思っています。ただ、商業ベースに持ってきたときに、全体を通してどこにお金が発生していて自分たちがどこにお金を払わねばならないのか、という構図をお客様が全部とらえるのは難しいと思うんです。単純に、フリーのものと、お金のかかるソフトウェア、という比較をされることもあります。我々のパートナー様に、「お客様にタダのものがあるといわれて困ったことになった。なんとかしてほしい」と言われたときには、きちんと我々のメッセージを伝えて、Server Appliance Kitのようなバリューを与えるツールをご提供していきたいですね。

マイクロソフト製品マーケティング本部Windows製品部サーバーグループ・グループプロダクトマネージャー・松井洋一氏

Linuxの生みの親

Linus Torvalds氏インタビュー

「あらゆる人は自分のやっていることを楽しむべきじゃないかな」

自叙伝『それがぼくには楽しかったから　全世界を巻き込んだリナックス革命の真実』（リーナス・トーバルズ、デビッド・ダイヤモンド著　風見 潤 訳　中島 洋監修 小学館・222ページ参照）のプロモーションのため、Linus Torvalds氏が来日した。今回の来日は6年ぶりで、その間Linuxと彼を取り巻く状況は大きく変わっている。6年前の来日時にも彼と会っていた、生越昌己氏がインタビューする。

生越：やっぱり今はあまり時間がないわけですか？

Linus：時間については、なるべく自分のために確保しようという努力はします。自分の興味のないこと、面白くないことに、絶対に時間を取られないようにしています。今週LinuxWorldが開かれる日本にいるにもかかわらず、会議には出たくないので行きません。それよりも観光をしたほうがいいので。なるべく時間を作って、子供と過ごしたり一緒にバケーションしたりといった努力はしていますので、プライベートの時間が持てなくてストレスがたまるということはないんです。

生越：あまり楽しくない仕事も、時として要求されると思うんだけど、どうしているんでしょう？

Linus：まあ、おかげさまでこんなに有名になったことの利点というのは、誰も私に「あなたはこうしなければいけない」と言える立場の人がいないことです。自分が「行かない」といったら行かない。もちろん、たとえばTransmeta*の中では、頼まれたら受けなければいけない要請はある程度はありますが、基本的には自分が今回のようなツアーを要請された場合でも、自分が行きたい、なるべく楽しく過ごせるところに行くことにしています。もちろんLinuxやTransmetaでやっている技術の仕事に関しても、自分の得意なこと、好きなことしかやらないようにしています。自分の得意じゃない部分、好きではない部分はほかの人が補ってくれるということになっているので、自分は今、非常に恵まれた立場に置かれていると思います。

生越：会社で自分の好きなことをさせてもらっている。時間をLinuxに費やすことができる。そうしたら、ふだんの頭の中はどうやって切り替えているの？　たとえば私なんかは自分が今考えているプログラムのことで頭がいっぱいになっちゃって、「ほかのことをやりたいな」と思いながらそのことを頭の中に入れておくことができないんだけれども。

Linus：実は切り替えはそんなに難しくないのです。Transmetaではアドバンスド・テクノロジーグループという、次世代のそのまた次のジェネレーションの開発にあたっているチームにいます。そして、去年の夏はカーネル2.4を出すことに忙しかった。基本的にどうするかというと、1週間ごとに集中してそれを切り替える。たとえばTransmetaの仕事が忙しいときは1週間やって、でもその間Linuxコミュニティで動きがあるときはそちらのほうを1週間集中的にやることに切り替える。その時どちらに重要な動きがあるのかというところで変えるわけです。同時に1つのことだけ長期間やっていると飽きてきますので、Transmetaの仕事をしばらくやって飽きたら変えるという形で切り替えています。

生越：それをしていいというのは羨ましいな。

＊Transmeta　Linus氏が所属するx86互換低消費電力CPU、Crusoeの開発元。

Linus：自由というか……、もちろん有名になったということの利点ですが、6年前も自分の好きなことができていました。

生越：うん、それはそうだ。

Linus：当時大学にまだいることにしたのも同じ理由からで、「好きなことを自由にやりたい」と思ったからです。Linuxではない会社に入ったのも、「Linuxであなたはこうすべきだ」といろいろな人に言われたくなかったからです。大学にいると、自由で好きなことをやって、面白いことをやっていればよしとしてくれるのですが、その代わりお金にはならない。ただ、私としてはその自由がありがたかったので、お金にならなくてもまったくよかった。米国に行くことにしたのも、やはり自分がしたいこと―Transmetaで面白いことができるので決めました。そういう意味で、自分の自由というのを確保しながらやってきました。もちろん幸運じゃなかったとは言いません。ただ、誰にでも自由の確保はできると思っています。もちろん、やっていることに対して自分が秀でていたり、才能があったら、なおいいですけれども。

生越：自由というのは「選ぶ」ことだということですね？

Linus：そうです。自己の選択によって自由を得ることを常に目指して、努力しています。Transmetaに行ったときはリスクはありましたが、そのリスクを負うことにして、いい形になりました。いつも誰でもそうだと思うのですが、自分がしたいことと社会的成功を天秤にかけたとき、なによりも楽しむべきだと思うのです。あらゆる人は自分のやっていることを楽しむべきじゃないかなと思います。

生越：今日は私はいくつか質問を作ってきていて、「モチベーションを維持するには」という話も聞こうと思ったんだが、そうすると、それはあまり聞く必要がないのかな？

Linus：ほんとうに最初から―大学に行くころから、自分のしたいこと、好きなことがわかっていました。自分が好きなことがこれだけはっきりわかっていると、モチベーションを維持していく必要はなくて、自動的にモチベーションがあり続けるんです。

生越：でも、時として楽しくないこともしなくてはいけないんじゃない？

Linus：人前で話すというのが大嫌いで、本当にやりたくないんです。

生越：人前で話すというのは、楽しいときと楽しくないときがあるから。でも、楽しいときもあるでしょ？

Linus：スピーチをすることが楽しいと思えることも、なくはないですが、それは聞いてくれる人と題材によります。Linux初期の頃はテクノロジーについて話してほしいという機会が多かったので、それは個人的に興味があることですから、当然それに関して人前で話すのは、ある程度楽だったです。でも最近はLinuxの政治的な背景についても話してくれとか、社会的な現象についてとか、テクノロジーの将来についてとか……。そういうことについては興味がないので、話したくはないのです。

生越：じゃあ、最後に1つだけね。次のバージョンの作業はいつから始めていつごろ終わる予定ですか？

Linus：バージョン2.5を開始するのが、6月終わりか7月中旬です。終わりはいつということは言わないようにしています（笑）。前の場合は9カ月と言っていたのが2年間になってしまいましたし。1年周期を目標としているのですが、開発初期の新しいフューチャーを作っている段階はみんな面白く楽しくやっているんですが、それを安定化させるという作業は、本当に大変で楽しくはないのです。やらねばならないのですけれども、誰も進んでやりたいということにはなかなかなりません。次のバージョンでどの程度変わるかわからないから、2.6になるか3.0になるかはわからないけれども、それを2002年に出そうと思っています。

生越：誰も言ったとおりに出るとは思っていないので（笑）。

Linus：（笑）一応は、ちゃんと目標を目指しています。

生越：誰も信じていないし（笑）。それよりも、ちゃんとしたものが出るほうが期待されていますよ。

Linus：そうですね。これまでの2.4もそうですが、安定したカーネルを出すということについては自信があります。

2.6になるか3.0になるかはわからないけれども、次のバージョンを2002年に出そうと思っています

速報 Special Report

LinuxWorld Expo/Tokyo 2001

国内最大のLinuxイベントである「LinuxWorld Expo/Tokyo 2001」が、5月30日から6月1日まで東京ビッグサイトで開催された。初日の模様を中心に速報レポートをお伝えする。

「LinuxWorld Expo/Tokyo 2001」が、東京・有明にある東京ビッグサイト「東京国際展示場」にて5月30日から6月1日までの3日間開催された。

1999年3月に最初に開催されてから、今回で5回目となる日本でのLinuxWorldだが、回を重ねるごとに規模が大きくなり、展示会場内に設けられた特設会場での基調講演やセミナー／ワークショップには多くの聴講者が来場した。

基調講演のトップはコンパック

初日、最初の基調講演は、コンパックコンピュータの副社長エンタープライズビジネス統括本部長の河合 聰氏によって「Linuxマーケットの展望とコンパックのLinux戦略」というテーマで行われた。

河合氏は、Linuxの市場は確実に拡大しており、インターネットのインフラとして重要な位置を占めていると述べ、企業システムへ導入するための課題も、大手ベンダーによるサポートによって解消されつつあるという。有望なLinuxマーケットとして、アプライアンス市場や携帯端末向けを挙げた。

また、コンパックにとってLinuxは、1st Tire（優先度1）のOSとし、同社のProLiantサーバをLinuxの標準サーバであると位置づけた。

途中、ドリームアーツ代表取締役の山本氏による、ナレッジマネージメントシステム「Miracle Linux INSUITE @Server」のデモが行われ、コンパックのPDAであるiPAQを利用したモバイルソリューションの例として、ビール会社の営業ツールとしての利用方法を紹介した。

オラクル＋ミラクル・リナックス

今回の出展社の中で最大のブースを構えていたのが、日本オラクルとミラクル・リナックスの共同ブースである。オラクルは、大きな特設ステージでプレゼンテーションを行っており、Oracle Parallel ServerやOracle9iAS、InterMediaなどを紹介した。

ミラクル・リナックスのブースでは、Miracle LinuxとOracleの組み合わせで動作するパートナー企業のアプリケーション製品の紹介コーナーのほかに、コンパックのLinuxソリューションパッケージシリーズである「Miracle Linux File/Print Serverモデル」のデモが行われていた。リコー、キヤノン、エプソン、沖データ、日本HP、京セラミタ、富士ゼロックスのプリンタメーカー7社のレーザープリンタを並べて、Windowsマシンのプリンタサーバとして動作させていた。

ディストリビューション

ターボリナックス ジャパンからは、Turbolinux Workstation次期製品デベロッパーリリースとして、Monza（コードネーム）が公開されていた。もちろんカーネル2.4ベースで、Turbolinux Server 6.5で先に採用された新インストーラMongooseが利用される。残念ながらTurbolinux Workstation 7として登場するのは秋になりそうだ。デモとして、USBカメラから取り込んだ画像の表示、MP3の音楽をUSBスピーカから再生、Matrox G400のデュアルヘッド機能を使って2台のディスプレイに

表示を行っていた。

今年5月に、OpenLinuxを開発／販売している米Caldera Systemsが、SCO（The Santa Cruz Operation, Inc.）のサーバソフトウェア事業部とプロフェッショナルサービス事業部を買収し、Caldera Internationalとして生まれ変わった。それに対応して、日本SCOも分社を行い、カルデラとタランテラという別の会社として再スタートした。

そのカルデラからは、カーネル2.4採用のOpenLinux 3.1のベータ版と、UnixWare 7の後継製品であるOpen UNIXが参考出品された。

現時点で唯一のカーネル2.4正式採用、Red Hat Linux 7.1をリリースしたばかりのレッドハットは、今回は出展していなかった。

Itanium搭載サーバ正式発売

インテルから5月29日に正式発表された64ビットCPUのItanium搭載マシンが、ハードウェアメーカーの各ブースで展示されていた。

NECから過去のLinuxWorldで参考出品されていた「AzusA」が、Linux／Windows（開発コードWhistler）を搭載OSするExpress5800/1160Xaと、HP-UXを搭載する「TX7／AzusA」として正式に発売された。733MHzまたは800MHzのItaniumを2〜16CPU搭載する。4CPU単位のサブシステムで分割運転が可能なため、サブシステムごとに独立して動作させることができる。高さ180cmのラック仕様で重量約400kgとかなり巨大なシステムである。

日立製作所からは、2〜8CPU搭載可能なItaniumサーバ「HA8000-ex/880」が発売された。日立が開発したMLPF（Multiple Logical Processor Feature）によって、複数OS／アプリケーション

コンパックコンピュータの副社長エンタープライズビジネス統括本部長の河合 聡氏による基調講演。

大勢の聴講者が集まったオラクルのステージ。ミラクル・リナックスは、レーザープリンタを並べてSambaによるファイル／プリンタサーバのデモを行った。

Turbolinux Workstation次期製品デベロッパーリリースMonzaで、USBカメラ／USBスピーカのデモをデュアルヘッドディスプレイで表示していた。

カルデラからは、カーネル2.4採用OpenLinux 3.1とUnixWare 7の後継OSであるOpen UNIXが参考出品された。

アミュレットからLinuxPPC 2000 Q4日本語版が発売される。GNOME 1.4とKDE 2.1.1を標準搭載し、Mac OSエミュレータとして「Mac-On-Linux」が付属する。

の並列稼働を実現している。

なお、日立から参考出品ながらItaniumベースのワークステーションも展示されていた。最大2CPUで、OpenGL対応の高速3Dグラフィックスボードを採用し、3次元CAD／CG／シミュレーションを高速に処理するといった用途を目的にしているが、発売は未定とのことだ。

日本SGIは、Itaniumベースのワークステーション「Silicon Graphics 750」を発表した。ちょっと大きめなミドルタワー型ケースに、シングルまたはデュアルプロセッサのItaniumを搭載し、最適化された科学技術計算ライブラリ（SCSL）を提供する。

ぷらっとホームからも、Itaniumを搭載した「PrizeX 9600M」が発表された。4Uラックマウントタイプのケースに、Itanium 733MHz、メモリ512Mバイト、18.2Gバイトハードディスクという構成で195万円。最大2CPU、メモリ8Gバイトまで搭載可能と拡張性も十分に備えている。

会場にはなかったが、Itaniumサーバに関しては、IBMやHP、コンパック、ノーザンライツコンピュータなども発表している。

オープンソースへの支援

オープンソース／フリーソフトウェアのプロジェクトや、ユーザーズグループの活動や成果物を紹介する場所として、今回より「.org Pavilion」が設置された。

スラッシュドット、OSDN、OSDL、オープンエーシーエスジャパン、GNU/Linux on SuperHプロジェクト、Debian Project／Debian JP Project、日本GNOMEユーザ会、日本Sambaユーザ会、日本PPC Linuxユーザーズグループ、日本PostgreSQLユーザー会、

NECが発売した最大16CPUのItanium搭載サーバ「Express5800/1160Xa」。従来AzusAと呼ばれていたもの。

日立ブースには8CPU搭載可能なItaniumサーバ「HA8000-ex/880」と、参考出品ながらItaniumを搭載したワークステーションも展示された。

ぷらっとホームからItaniumを搭載したサーバ「PrizeX 9600M」が発表された。最小構成で195万円から。

日本SGIが発表したItaniumベースのワークステーション「Silicon Graphics 750」。ちょっと大きめなミドルタワー型ケースもSGIらしいデザインでスッキリ見える？

IBM pSeriesを4台使用したクラスタシステムで3Dアニメーションのレンダリングを行っていた。OSはAIXではなくSuSE 7.1のPowerPC Editionで動作していた。

フリースタンダードグループ、フリーソフトウェア財団、もじら組といった多くの団体が参加した。

なお、米オレゴン州ポートランドにLinux開発者向けの研究施設を設置している、オープン・ソース・デベロップメント・ラボ（OSDL）では、今年12月までに東京近郊に世界で2番目となる施設を開設することを発表した。

プレステ2でLinux

インターネットでの初回注文分2000台がわずか8分で売り切れた、注目のプレイステーション2用Linuxキット（β版）が、Debian Project／Debian JP Projectの合同ブース内で展示されており、大勢の人が詰めかけていた。

これは、SCE（Sony Computer Entertainment）によるものではなく、PSLUG（PlayStation 2 Linux Users Group）というユーザーズグループによって先行展示されたものであるが、一般に公開されるのは初めてのことだ。

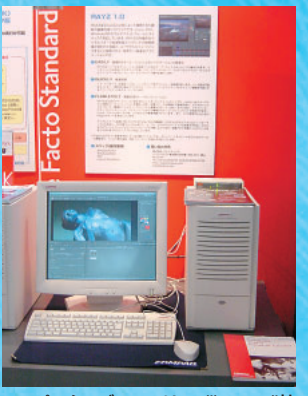
Linuxはビジネス市場へ

今回のLinuxWorldでは、カーネル2.4採用のディストリビューションのリリースが遅れているのが残念だった。ユーザーの期待に応えるべく、ディストリビューターには頑張ってほしい。

大手ハードウェアメーカーのほとんどが大きなブースを構えており、Linuxに対して本格的に取り組んでいることがわかる。IA-32サーバ製品では、OSにWindowsと並んでLinuxを選べるし、64ビットのItaniumでは、LinuxがWindowsに先行して使われている。ハイエンド分野でのOSとしてLinuxの応用が広がっていく予感を感じさせた。

ServeLinux Networks社が開発した「Xgate Wireless DSL SOHO Server」は、インターネットとはDSLで、クライアントとは11Mbpsの無線LANで接続するSOHOサーバ。

ぷらっとホームではRealNetworksのRealSystem Serverを搭載したストリーミングサーバを展示していた。1Mbpsで配信するとリアルな画像で見ることができる。



コンパックのブースでは、SiliconGrail社のRAYZという画像合成ソフトのデモを行っていた。64ビットAlpha CPUを使ったシステムの展示も多く行われていた。

.orgパビリオンでは、オープンソース／フリーソフトのプロジェクトが出展していた。ここでしか買えない、さまざまなキャラクタのTシャツの即売も。

注目のプレイステーション2用Linuxキット（β版）を一目見ようと、大勢の人が駆けつけた。

緊急レポート

PlayStation 2用 Linuxキット登場

高性能ゲームマシンであるPlayStation 2でLinuxが動く！ コミュニティの要望にメーカーが答えた結果だ。出荷を前に、緊急レポートをお届けする。

PlayStation 2（以下PS2と表記）は、最高水準のハードウェア能力を持つ家庭用ゲームマシンである。2000年3月のリリース時から、その開発環境にLinuxが利用されていることが知られていた。

今年の3月「PS2で動作するLinuxを公開してほしい」という署名活動が、日本のLinuxコミュニティにより始められた（詳細は以下のウェブサイトを参照）。

http://www.peanuts.gr.jp/pslinux/

このコミュニティの熱望に対する答えとして、PS2の発売元であるソニー・コンピュータエンタテインメント（以下SCEIと表記）は4月26日、ついにPS2用Linuxキット（β版）のリリースを発表した。

その人気は非常に高く、初期ロットの2000台は予約開始後8分足らずで完売してしまったほどだ。その後さらに6000台の追加出荷が発表され、合計8000台が出荷されることとなった。

リリース前のLinuxキット（β版）を試すことができたので、その詳細をお伝えする。なお、今回試したものは、6月20日以降に出荷されるキットとは細部が異なる可能性があることをお断りしておく。

■キットの内容

PS2用Linuxキットは**表1**に示すようにDVD-ROMと、Linuxを利用するために必要なハードウェアからなる。

ハードディスクは、PS2の本体と似たデザインの外付けタイプだ（**写真1**）。接続ケーブルのPCカード側には、100BASE-TX／10BASE-T用のコネクタがあり、ネットワークの利用が可能だ。

なお、このハードディスクは、PS2本体のPCカードスロットを利用して接続するため、PCカードスロットを持たないPS2では利用できない。具体的には、型式番号がSCPH-10000、SCPH-15000、SCPH-18000なら利用可能、最新型のSCPH-30000では利用不可だ。

画面表示には家庭用のテレビではなく、“Sync on Green”に対応したPC用の高解像度ディスプレイが必要だ。

写真1　外付けハードディスク
PS2本体とマッチした外観を持つ。開発用の「TEST」という文字が……

写真2　ランタイム画面
起動するカーネル（メモリーカード）を選択する。

写真3　デモプログラム
付属のライブラリを使用したデモプログラム。

写真4　デスクトップ
Xが起動すれば、一般的なPCと変わらないデスクトップが表示される。

最大で、1280×1024（16ビットカラー）までの解像度に対応している。

キーボードとマウスは一般的なUSB接続タイプのものだ。未確認だが、適切に設定すれば、英語配列のキーボードなども利用可能と思われる。

■ハードウェアのマニュアルが付属

キットに含まれるDVD-ROMには、Linux本体のほかに、PCのBIOSに相当するランタイム（**写真2**）や、Emotion Engine（CPU）やGraphic Synthesizer（グラフィックスコントローラ）などハードウェアのマニュアルが含まれている。

ゲームマシンに使用されているCPUやグラフィックスコントローラの仕様が正式に公開されるのは非常に珍しいことだ。PS2のこれらのハードウェアは、用途によっては、x86などの汎用CPUをしのぐ性能を持つ。Linuxキットによって、これらのハードウェアを利用したプログラムが作成できることになる。今後コミュニティの手により、数多くの利用方法が考え出されることだろう。

■ソフトウェア

PS2 Linuxは、Red Hat系Linuxをベースにしており、カーネル2.2.1、glibc 2.2.2、XFree86 3.3.6などが採用されている。そのほかPS2 Linux用に、低レベルグラフィックスライブラリや、Mesa 3Dに対応したコンソール用ドライバなどが用意されている（**写真3**）。

■見た目はやはりLinux

電源スイッチを入れ、Linuxを起動するまではPS2独自の作業が必要だが、いったん起動してしまえば、x86用のLinuxとまったく変わらないデスクトップが現れる（**写真4**）。

メインメモリが32Mバイトと少ないため、GNOMEのような統合デスクトップ環境ではスワップが多発するが、Window Makerのような比較的軽いウィンドウマネージャを用いれば、快適に動作する。

■今後の発展は、反響しだい

SCEIは、オープンソースコミュニティへのコミット継続をアナウンスしているが、今後の方向性については、キットへの反響を見て決めていくようだ。LinuxとPlayStation 2の新たな発展に期待しよう。

DVD-ROM 1枚
Linux インストーラ
PlayStation 2 ランタイム
システムマニュアル
外付け型ハードディスクドライブ（40Gバイト）
PCカードスロットで接続（SCPH-30000では使用不可）
PCカード側に100BASE-TX／10BASE-T用コネクタを持つ
USB接続キーボード（106配列）
USB接続マウス（3ボタン）
専用ディスプレイケーブル
“Sync on Green”対応のディスプレイが必要
家庭用テレビは使用不可

表1　PlayStation 2用Linuxキットの内容

interview

PlayStation 2用Linuxキットをリリースする背景を、SCEIのネットワーク事業部部長の島川恵三氏、ソニー ソフトウェア開発担当部長の阿部雅人氏にうかがった。

――PS2で動作するLinuxを求める署名活動から、今回のキットリリースまで非常に短時間でしたが。

PS2へのLinux移植は、以前から行われていました。PS2の開発環境にもLinuxが使われているので、開発チーム内に自然にそういう動きができていました。署名活動に対してすぐに対応できたのは、このためです。

――最初の2000台に加えて、追加の6000台もすぐに完売したようですが。

わずか8分で2000台の予約が決まったことも含めて、反響の大きさに驚いています。

――PS2 Linuxの開発で一番苦労された点は？

メモリが少ない（32Mバイト）ため、それに合わせて組み込む機能を選択していかねばならなかった点です。ユーザーが自由に使えるフリーメモリを多く残したほうが、良いと考えました。

――ソースコードはどこまで公開されているのでしょうか？

PCのBIOSに相当する部分や、コントローラ、メモリカード、サウンドなどのデバイスドライバなどを除いて、すべてGPLで公開されます。

――PS2 Linuxを用いてユーザーが作成したソフトの扱いは？

公序良俗や法律に反しない範囲で、お楽しみいただけます。

――海外版のPS2に対応したキットをリリースする予定はありますか？

今回のキットでは利用できないSCPH-30000への対応なども含めて、今後の展開は、今回のキットへの反響を見てから検討することになっています。

――どのようなソフトが出てくるとお考えですか？

いろいろな可能性が考えられると思いますので、今のところまったく予想できません。

――ありがとうございました。最後にコミュニティに一言お願いします。

PS2でのプログラミングを思う存分楽しんでください。

阿部　雅人氏
ソニー株式会社
ネットワーク＆ソフトウェアテクノロジーセンター
SA開発部門　基本システム開発部
ソフトウェア開発担当部長

島川　恵三氏
株式会社ソニー・コンピュータエンタテインメント
ブロードバンド事業本部
副本部長
ネットワーク事業部　部長

Distribution

新着ディストリビューション

SuSE Linux 7.1 Japanese edition

SuSE Linuxが日本語対応を果たした。SuSE Linuxはドイツ産のディストリビューションで、ヨーロッパで高いシェアを誇っている。2.4系カーネルやReiserFSを当然のようにサポートするあたりは、Linuxカーネルの開発者が多数在籍しているSuSEの高い技術力の片鱗をうかがわせてくれる。さて、ゲルマンのディストリビューションは、日本のLinuxユーザーにどんな世界を案内してくれるだろうか。

Red Hat Linux 7.1

Linux界を牽引し続けるディストリビューションRed Hat Linuxの最新版、Red Hat Linux 7.1の登場だ。Red Hat Linux 7.1は、2.4カーネルや高速WebサーバTUXなどの採用で、エンタープライズ対応はさらに強化された。シングルバイナリでの多言語対応も果たし、メンテナンスの利便性も大きく向上している。

Linux Mandrake 8.0

つねに最新パッケージで構成されるフランス産のディストリビューション、Linux Mandrakeがメジャーバージョンアップを果たした。Mandrakeに採用されたGNOME 1.4は、高機能GUIシェルNautilusを統合して、Windowsに迫るデスクトップ環境を提供している。また、サーバ環境に求められるLVMを設定するGUIツールを備えるなど、バランスよくデスクトップとサーバに対応している。

Miracle Linux Standard Edition V1.1

目指せハイエンドサーバ。Oracleデータベース対応が特徴のMiracle Linuxは、システムの安定性を確保するために、2.2カーネルを採用する。だが、Miracle Linuxのカーネルは、ジャーナリングファイルシステムReiserFSやext3に対応し、Oracleの稼動のためにパラメータが最適化され、中身は完全なサーバ仕様となっている。

SuSE Linux 7.1 Japanese edition

ドイツ産ディストリビューションSuSE Linuxの日本語版となる、SuSE Linux 7.1 Japanese edition（以下、SuSE 7.1J）が公開された。

SuSE 7.1Jは、基本システムにカーネルに2.4.2と2.2.19、ライブラリにglibc2.2.9、XにXFree86 4.0.2と3.3.6を採用している。

オリジナルのSuSE Linux 7.1（以下、SuSE 7.1）は、IA32、IA64、PowerPC、Alpha、SPARC、S/390と多くのアーキテクチャに対応しているが、日本語に対応しているのはIA32版のみだ。

国際化を目指すSuSE

公開されたSuSE 7.1Jは、日本語環境に必要なRPMパッケージを、オリジナルのSuSE 7.1に追加したものだ。

SuSE Inc.は、次期バージョン7.2以降での多言語対応を目指しており、SuSE 7.1Jはそのプレ版といった趣である。SuSE Inc.では、日本語用のRPMパッケージだけでなく、日本語のドキュメントも準備中だという。

日本語に対応した最新のGNOMEとKDE

SuSE 7.1Jが収録するGNOME 1.4には、高機能ファイルマネージャNautilusが新たに統合されている（**画面1**）。

Nautilusは、ファイルマネージャとしてだけでなく、Mozillaを内部に組み込んだWebブラウザとしての機能も備えている。

また、基本部分にCORBAというオブジェクト間通信の仕様をもとにしたライブラリを使っており、Windowsのエクスプローラや、Mac OSのFinderのようなGUIシェルとして動作する。

Nautilusがうまく機能すれば、X Window Systemを、つまりUNIX系OSをWindowsのようなデスクトップ環境として使えるようになるので、Linuxコミュニティだけでなく、Sunなどの商用UNIXベンダーからも大きく期待されている。

SuSE 7.1Jに収録されているNautilusは、メニューなどの日本語表示は可能だが、GTK+からMozillaの機能を使うgtkEmbedに不具合があるらしく、kinput2を使った日本語入力ができなかった（ただし、Mozillaを単体で使う場合は、日本語入力可能）。

このほか、SuSE独自の設定ツールYaST2（Yet another Setup Tool version.2）も日本語に対応した（**画面2**）。

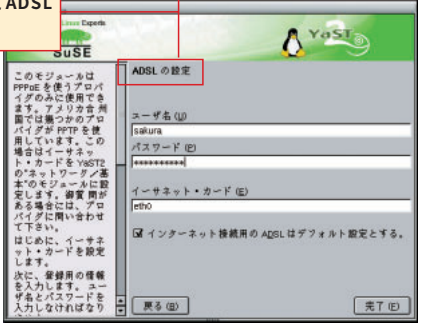
これまでのSuSE Linuxは、技術的には良いものを持っているが、日本語に対応していなかったため、日本での認知度が低かったように感じる。

だが、SuSE 7.1Jの出来を見る限り、日本での注目度も上がっていくのではないだろうか。

SuSE 7.1JのCD作成用ファイルは、（ftp://ftp.kddlabs.co.jp/Linux/packages/SuSE/suse/i386/japanese-7.1/）などからダウンロード可能だ。

画面1　Nautilusを取り込んだGNOME 1.4
SuSE 7.1Jに収録されるGNOMEは、ほぼ完全に日本語に対応している。

YaST2を使ってADSLを設定する



画面2　日本語化されたYaST2
YaST2は、SuSE独自の設定ツールだ。GNOMEやKDEと同じく日本語に対応した。

Red Hat Linux 7.1

Linuxディストリビューションの代名詞とも言えるRed Hat Linuxの最新版Red Hat Linux 7.1（以下、Red Hat 7.1）がリリースされた。先にリリースされたFTP版に続き、製品版の販売も5月11日から始まっている。

Red Hat 7.1の基本システムは、カーネル2.4.2、glibc2.2.2、XFree86 4.0.3という構成だ。

製品版のRed Hat 7.1はデスクトップ向けのDeluxeとサーバ向けのProfessionalの2種類で、おもな収録物は**表1**のとおりだ。

どちらも基本的なインストールサポートが付帯していて、Professionalユーザーは、Deluxeに付属するサポートに加えて、ApacheとBINDの設定、ソフトウェアRAIDの設定など、さらに高度なサポートを受けられる。

サポート期間はDeluxeが60日間、Professionalが90日間で、ともに件数無制限だ。

製品名	Red Hat Linux 7.1
価格	1万4800円（Deluxe） 3万4800円（Professional）
問い合わせ先	レッドハット株式会社 03-3257-0411 http://www.redhat.com/jp/

シングルバイナリで多言語対応

これまでRed Hat 7といえば英語版、Red Hat 7Jといえば日本語版だったが、Red Hat 7.1からはRPMパッケージが各国語で共通となったため、「J」の表記を持つRed Hat Linuxが姿を消した。

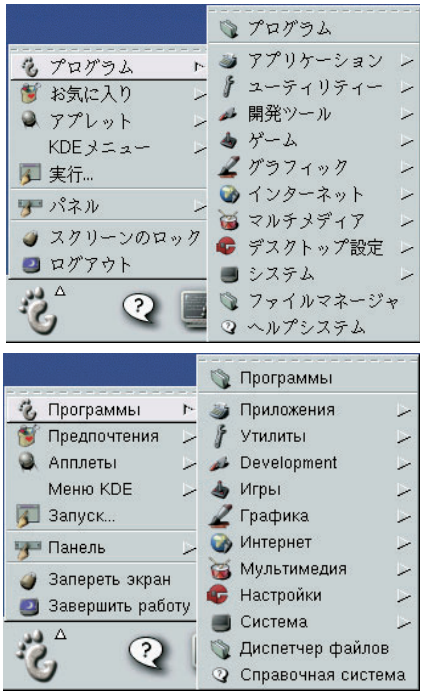
つまり、Red Hat 7.1のCD-ROMを持っていれば、英語はもちろん日本語やロシア語など多くの言語に対応した環境を利用できるというわけだ。

GNOMEのスタートメニューが、ログイン画面で選択した使用言語で表示されるのがその例だ（**画面1**）。

RPMパッケージが各国語で共通になると、日本人を含む英語圏以外のユーザーが、セキュリティ関連のアップデートパッケージを、本家とタイムラグなしに利用できるという大きなメリットができる。また、「このRPMパッケージは日本語に対応してるんだっけ？」と迷わなくてよいのが嬉しい。

FTPサイトにはjaがつくISOイメージファイルが存在するが、これはkonなどを収録してテキストモードのインストール中に日本語を表示させるためのもので、基本的な構成は同じである。

本誌付録CD-ROM収録のRed Hat Linux 7.1はFTP版です。非商用ソフトだけが含まれており、製品版を販売しているレッドハットからのサポートを受けることはできません。



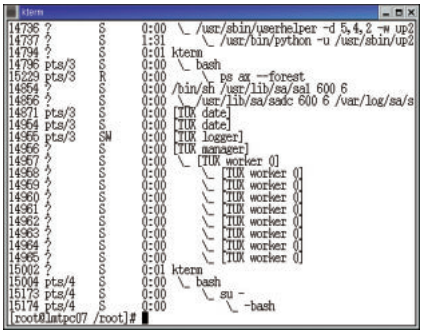
画面1　多言語対応したGNOME
グラフィカルログイン画面で選択した使用言語でメニューが表示される。

2.4カーネル採用

Linuxディストリビューションでも

収録物	概要
インストールCD 2枚	各国語共通のインストーラ
ソースCD	ソースRPMパッケージを収録
Power Tools CD	EtherealやRubyなどの追加パッケージを収録
ドキュメントCD	ダイヤルアップ接続やサウンドドライバなどの設定方法を解説
Loki Game CD	SimCity 3000などのデモ版を収録
デスクトップ用アプリケーションCD 2枚	・Wnn6 ・DynaFont ・HancomWord ・PatitionMagic SE などの商用ソフトを収録
サーバ用アプリケーションCD 2枚	ColdFusionなどのサーバ向け商用ソフトを収録
Developers Module Archive CD	PerlやPythonの開発時に便利なモジュールなどを多数収録

表1　製品版Red Hat 7.1のおもな収録物
CD-ROMのほかにもインストールマニュアルが収録される。



画面2　psコマンドの実行結果
動作中のプロセスを表示すると、WebサーバTUXがカーネルモードで動作していることがわかる。

っとも基本となるカーネルは、2.2.16から2.4.2にバージョンアップされた。2.4カーネルの採用で、大容量メモリやReiserFSへの対応をはじめとしてRed Hat Linuxのハイエンド対応はより強化されている。

■SMP環境でネットワーク性能が向上

Linuxは「ロック」という機構で、複数のCPUがカーネル内の同じリソースへ同時にアクセスしないように工夫されている。

カーネル2.4では、カーネル内にあるネットワーク関連のリソースへ対するロックが、より細やかなものに変更され、2.2カーネルに比べて、複数のCPUを搭載するマシンがより高いネットワークパフォーマンス出せるようになった。

■高速WebサーバTUX

Red Hat 7.1には、もっともポピュラーなWebサーバApacheに加えて、Red Hatが開発するTUXというWebサーバが追加された。

TUXはネットワークドライバと同じようにモジュールとしてLinuxカーネルに組み込まれる。つまり、Linuxカーネル自身がWebサーバとして機能するわけである（**画面2**）。

現時点では、CGIの動作に関してApacheが優位を保っているが、TUXはWebページのデータをカーネルのキャッシュとして扱えるので、静的なWebページを処理する場合に、特に高いパフォーマンスを発揮する。

2.4カーネルとTUX Webサーバを取り込んだRed Hat 7.1は、SMP環境で高速なWebサーバとして動作するだろう。

TUXの性能は、SPECweb99（http://www.spec.org/osg/web99/results/web99.html）のベンチマークなどを参考にしていただきたい。

なお、カーネル2.4にはkhttpdというモジュール化されたWebサーバが含まれているが、Linuxカーネルのメーリングリストでは、khttpdの代わりにTUXを組み込んではどうかという意見も出ている。また、TUXとApacheは、それぞれに別々のポートを割り当てて共存させることも可能だ。

■さらに強化されたUSBサポート

2.4カーネルは標準でUSBをサポートしている。店頭で売られている周辺機器は、プリンタやスキャナをはじめとしてUSBタイプが主流になりつつあるので、デスクトップユーザーにとっても2.4カーネルの採用はうれしいだろう。また、USBを含むホットプラグ機器の対応も強化されている。

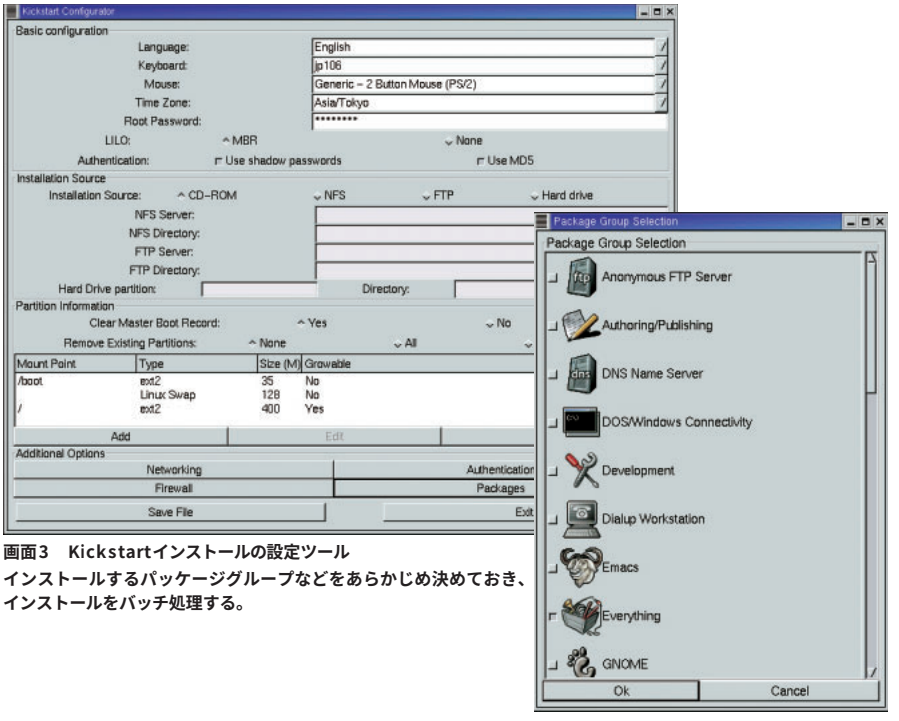
ぐいぐいKickstart

1〜2台のマシンにLinuxをインストールするのと違って、企業などで10台を超えるマシンにLinuxをインストールするのは結構骨の折れる作業だ。

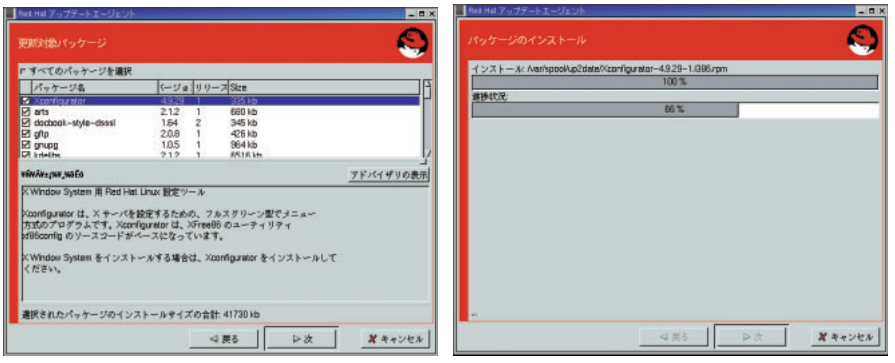
Red Hat Linuxは以前からKickstartインストールという、半自動化されたインストール機能を備えていたが、設定ファイルの作成が難しいためか、それほど注目されることがなかった。

Red Hat 7.1にはKickstartインストールの設定ファイルをGUIで作成できるツール（**画面3**）が追加され、より簡単にKickstartインストールできるようになっている。

この設定ツールの項目は、Red Hat 7.1のインストーラとほぼ同じ構成だ。



画面3　Kickstartインストールの設定ツール
インストールするパッケージグループなどをあらかじめ決めておき、インストールをバッチ処理する。



画面4　Red Hat Networkを利用するためのツール
ネットワーク経由でパッケージリストを更新して、アップデートパッケージをダウンロード後にインストールする。

パーティションやインストールするパッケージグループなどを選択し、[ks.cfg]というファイルを作成してフロッピーに保存する。

あとは、CD-ROMなどでRed Hat 7.1のインストーラを起動して、起動時に表示される[boot:]プロンプトに[linux ks=floppy]と入力してやれば、ネットワークカードやビデオカードの設定をのぞいて、ほぼ全自動でRed Hat 7.1がインストールされるという段取りだ。

アップデートの必要性

今年のはじめにWU-FTPDなどの持つセキュリティホールをついて、「Ramen」というワームが猛威をふるった。この「Ramen」から身を守るためのアップデートパッケージはかなり以前から配布されていたが、アップデートパッケージのリリースや、アップデート方法を知らないユーザーが多かったせいか、アップデートされていない多くのRed Hat Linuxサーバが「Ramen」の被害にあった。

Red Hatは効率よくシステムをアップデートするために、Red Hat Networkというサービスを本格的に稼動し始めている。製品版購入者は、ユーザー情報とマシンのシステム情報を登録したうえでサービスを利用する。マシン1台でのサービス利用は無料だが、2台目以降のマシンでの利用には追加料金が必要となる。

このサービスを利用するには、まずrhn_registerというアカウント作成ツールを起動して、ユーザー情報とマシン情報など登録する。このあとup2date-configでアップデートに関する設定を行い、アップデートツールup2date（**画面4**）を起動すれば、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、Red Hat Linuxを最新状態に保てるというわけだ。

なお、このサービスは米国向けのもののみが提供されている。

Webブラウザに Mozillaも収録

サーバ向けの拡張が目立つRed Hat 7.1だが、デスクトップ環境にも大きな変化がある。

Red Hat 7.1には、WebブラウザとしてNetscapeのほかにMozilla 0.7も採用されている（**画面5**）。Red Hatは開発体制がオープンであるということを理由に、Mozilla 1.0がリリースされ次第、Mozillaを標準のブラウザに採用する予定だという。

Mozillaはソースコードが公開されたNetscapeをもとにして開発されているブラウザだ。スタイルシートなどを正しく表示できるのが特徴で、開発時から多言語対応をめざしているため、Netscapeに比べて日本語入力などの動作が安定している。

ただ、Red Hat 7.1に収録されているMozillaは日本語ロケールデータが含まれていないため、メニューなどを日本語表示できない。このあたりは、標準のブラウザになるまでに解決を期待したいところだ。

画面5　新たに追加されたブラウザMozilla
Netscapeも同梱されているが、レンダリング能力ではMozillaが勝る。

Column

Red Hat Linuxでらくらく使うXFS

近頃Linux界隈ではファイルシステムが熱い。多くのLinuxディストリビューションではext2というファイルシステムが採用されているが、カーネル2.4.1からはジャーナリング機能を備えたReiserFSが取り込まれるなど、ext2以外のLinuxで使えるファイルシステムが着々と増えつつある。

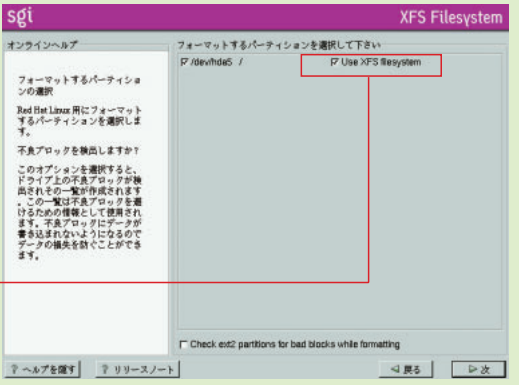
Linuxで使えるジャーナリングファイルシステムとしては、Hans Reiser氏の開発するReiserFS、IBMのJFS、SGIのXFSの3つがポピュラーだ。この中から5月1日にLinux版が正式リリースされたXFSを紹介する。

XFSはグラフィックワークステーションのベンダとして知られるSGI（Silicon Graphics, Inc.）が開発するジャーナリングファイルシステムで、ReiserFSと同じくB+ツリーというアルゴリズムでメタデータを管理する。XFSはSGIのIRIXというUNIX OSで使われてきたので、ファイルシステム自体の動作実績は豊富だ。

2.4系のLinuxカーネルに対するXFSのパッチと、XFSを使うためのツールはGPLで配布されている。XFSの使用には、XFSパッチをあてたLinuxカーネルの再構築と、ファイルシステム作成用ツールなどのコンパイルが必要だ。

しかし、SGIはXFSに対応したRed Hat 7.1用のインストーラ（本誌には収録されていない）を公開しているので、これと本誌付録のRed Hat 7.1のCD-ROMを合わせて使えば、カーネルを再構築をせずにXFSを使ったLinuxを簡単にインストールできる。

画面1　SGI製のインストーラ
インストーラの上側はSGIのロゴが入った紫色だ。XFSに対応したRed Hatは、MBRにインストールされたLILOか、レスキュー用のディスクで起動する。GRUBはまだXFSに対応していないが、現在XFS対応に向けた開発が進行中である。



ここにをチェックすると、パーティション作成場面で［Linux native］と指定したパーティションがXFSでフォーマットされる。

文末のFTPサイトにあるISOファイルをもとに作成したCD-Rからマシンを起動すると、**画面1**のようなRed Hat 7.1のインストーラが立ち上がる。作業手順は、オリジナルのインストーラとほとんど同じなので、Red Hatをインストールできる読者ならば、XFSを使ったRed Hat 7.1も簡単にインストールできるだろう。

さて、気になるXFSのパフォーマンスだが、SGIのインストーラを使って構築したRed Hat 7.1上で、ext2、ReiserFS、XFSの簡単なベンチマークをとってみた。テスト環境は600MHzのCyrixIIIと64Mバイトのメモリを搭載したマシンだ。

グラフ1は、Linuxカーネルを含むディレクトリのコピー、削除、アーカイブにかかる時間を測定した結果を、グラフ2はBonnie++というベンチマークソフトを使って、120Mバイトの連続したファイルの入出力について測定した結果を示している。

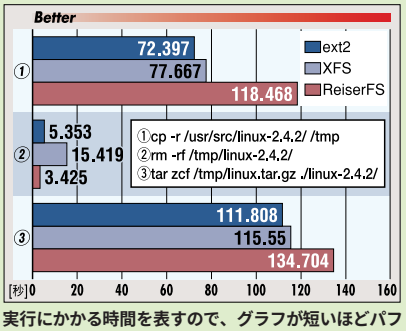
Linuxカーネルを含むディレクトリには、約6000個のファイルとディレクトリが含まれていて、総容量は約100Mバイトだ

2つのグラフを見ると、XFSは大量ファイルを削除するオペレーション以外では、ReiserFSよりもおおむね優れたパフォーマンスを出していることがわかる。また、**グラフ1**を見るとジャーナリング機能を持たないがゆえにext2がパフォーマンスを発揮するオペレーションでも、ext2に匹敵する結果を出していることがわかる。

さてさて、Linuxのファイルシステムはこれからどんどんおもしろくなりそうだ。

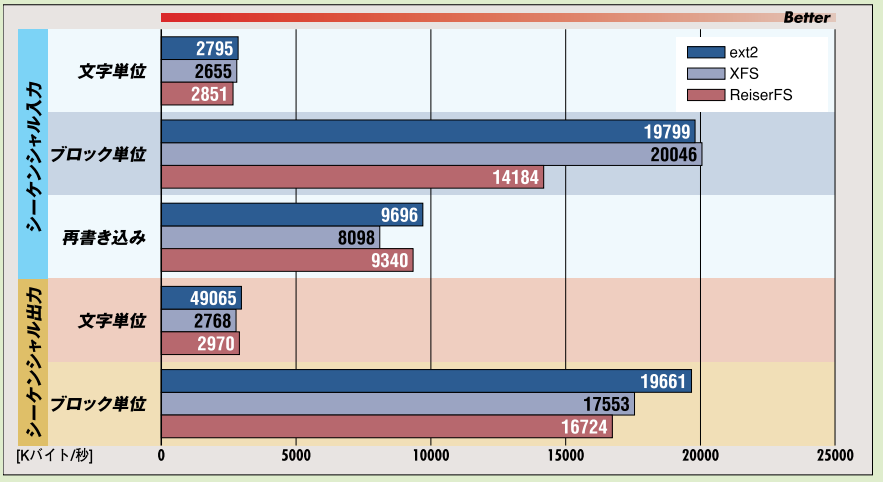
■SGIのXFSプロジェクトページ
http://linux-xfs.sgi.com/projects/xfs/
■SGI製のRed Hat 7.1用のインストーラ
ftp://oss.sgi.com/projects/xfs/download/Release-1.0/iso/RH7.1-SGI-XFS-1.0.iso

グラフ1　大量のファイルをあつかう



実行にかかる時間を表すので、グラフが短いほどパフォーマンスが良い。

グラフ2　Bonnie++を使ったベンチマーク



グラフが長いほどパフォーマンスが良い。

Linux Mandrake 8.0

フランス産のLinuxディストリビューション、Linux Mandrake 8.0（以下、Mandrake 8.0）が4月19日にリリースされた。

基本システムには、カーネル2.4.3、glibc2.2.2、XFree86 4.0.3を採用している。また、GNOME 1.4とKDE 2.1.1を採用して、デスクトップ環境は最新の構成だ。

GNOME 1.4登場

Mandrake 8.0は、ほかのディストリビューションに先駆けて、GNOMEの最新バージョン1.4を収録している。

■Nautilus

GNOME 1.4と、ほかのディストリビューションで採用されているGNOME 1.2との違いは、高機能ファイルマネージャNautilus 1.0（**画面1**）を標準で収録していることだ。

Nautilusは、ファイルマネージャとWebブラウザの機能を持ち、WindowsのIEのように一種のシェルとして機能するアプリケーションだ。

Nautilusの開発には、Macintoshのインターフェイス設計を手がけた元Appleの開発者も関わったので、デザインやインターフェイスは非常に先進的なものとなっている。

Nautilusを起動すると、まず表示されるアイコンの持つ情報量の多さに驚かされる。画像ファイルのアイコンは画像のサムネイルが表示されるし、テキストファイルのアイコンにはファイルの先頭から数行の文章が表示されるなど、アイコン自体がプレビュー機能を備えているのだ。

また、一覧表示された状態の小さなアイコンも個別に拡大できるので、アイコンの表示モードに関係なくプレビュー機能を利用できる。

このほかにも、MP3などの音楽ファイルがあるディレクトリで［View as Music］を選択すれば、XMMSなどを起動することなく音楽ファイルを再生可能だ。この機能は、音楽ファイルのイントロ部分を知りたい時に便利だ（**画面2**）。

残念ながら、先頃Nautilusを開発したEazelが倒産してしまったが、オープンソースのNautilusは、GNOME projectのよって開発が続けられていくだろう。

■Evolution

NautilusがIEならば、Ximian（旧Helix Code ）が開発するEvolution（**画面3**）は、メール、スケジュール、アドレスなどを一括管理するOutlookのような存在である。

Evolutionは、POPだけでなくIMAPもサポートしている。

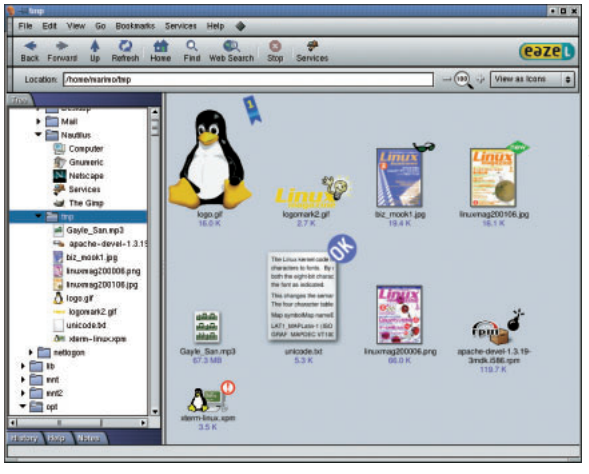
コントロールセンター

Mandrakeの特徴である統合設定ツールは、名前をDrakeconfからコントロールセンターに改めて、さらに使い勝手が向上している。

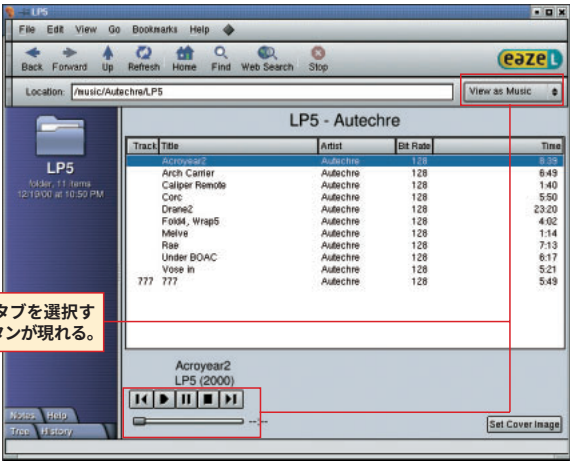
■Windowsライクな Xの設定ツール

Mandrake 8.0のコントロールパネルで［Display］を選択すると、Windowsの［画面のプロパティ］に似た画面が表示される（**画面4**）。

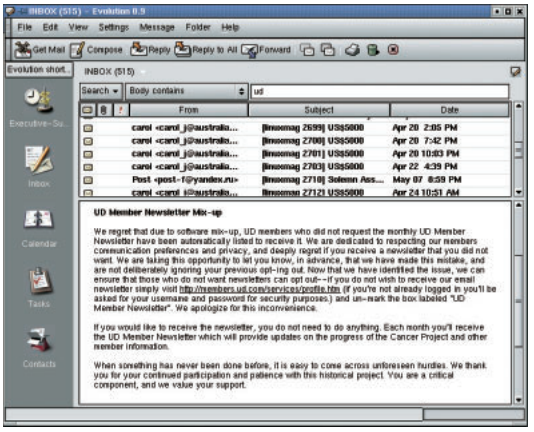
この設定ツールは、XFree86 4.0.3



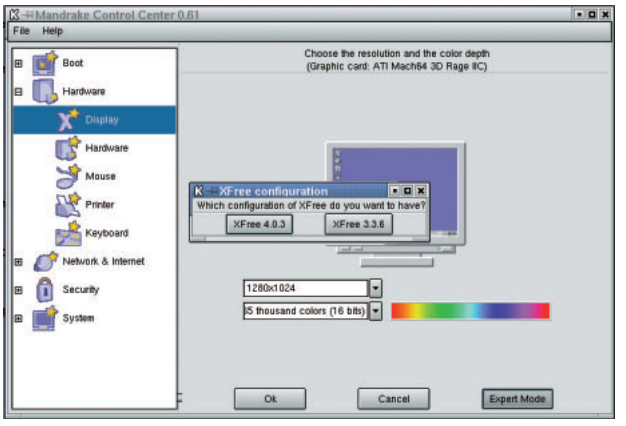
画面1　高機能ファイルマネージャ Nautilus
Nautilusはファイルマネージャと Webブラウザの機能をもつ。



画面2　Nautilusの［View as Music］モード



画面3　多機能メーラEvolution
Evolutionは、メールの管理だけなくスケジュールなどの管理もできる。



画面4　コントロールセンターから呼び出したX設定ツール
Mandrake 8.0のX設定ツールは、XFree86 4.0.3とXFree86 3.3.6の両方に対応している。

とXFree86 3.3.6の両方に対応している。設定内容を有効にするのにマシンの再起動が必要なのが難点だが、使い勝手はまずまずだ。

■LVMをGUIで設定

2.4カーネルには、複数のハードディスクをまたいで、複数のパーティションをひとつのドライブとして扱うLVM（Logical Volume Manager）という機能が加わった。

LVMを使って作成した仮想的なドライブは、一度作成した後でもパーティションを追加していけるので、ドライブサイズを柔軟に変更可能だ。

システム運用中にディスクの空き領域に余裕がなくなっても、どんどんドライブのサイズを増やせるので、とても便利である。

いくつかのディストリビューションはLVMに対応しているが、どのディストリビューションでも、LVMの設定にはコマンド操作が必要だ。

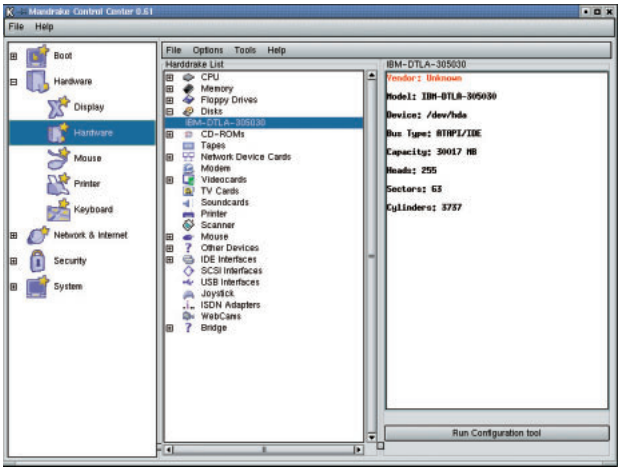
しかし、Mandrake 8.0では、コントロールパネル（**画面5**）から呼び出すDiskDrake（**画面6**）というツールを使えば、LVMをマウス操作で設定できる。

DiskDrakeは、Mandrake 8.0のインストール時にも、パーティション設定場面で使われており、インストール時でもLVMを設定可能だ。

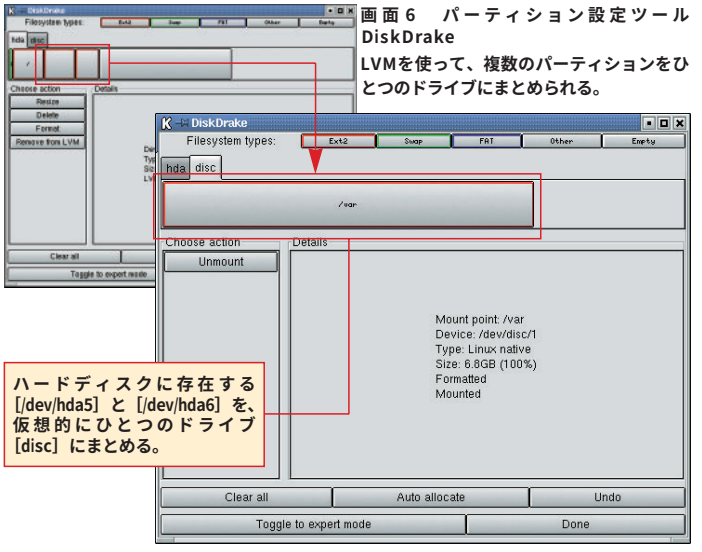
強く望まれる日本語対応

デスクトップ環境も、設定ツールも、完成度の高さでは、ほかのディストリビューションの一歩先をいくMandrake 8.0だが、惜しむらくは日本語にほとんど対応していない。

Nautilus（Webブラウザ）とEvolution（メーラ）は、Linuxのデスクトップ環境においてキラーアプリになる可能性を秘めているので、Red Hat Linuxのようにシングルバイナリで多言語対応して、ぜひ日本語環境として使えるように拡張して欲しいところだ。



画面5　DiskDrakeを呼び出すコントロールセンター
コントロールセンター起動して、[Hadrware] — [Hadrware] と進み、ハードディスクを選択して [Run Configuration tool] を押すとDiskDrakeが起動する。



画面6　パーティション設定ツールDiskDrake
LVMを使って、複数のパーティションをひとつのドライブにまとめられる。

Miracle Linux Standard Edition Version 1.1

ミラクル・リナックスは、Oracleデータベースに最適化したMiracle Linux Standard Edition Version 1.1（以下、Miracle 1.1）を5月25日に発売した。

Miracle 1.1は、**表1**のOracle製品をサポートしている。また、基本システムにカーネル2.2.18、glibc2.1.3を採用して、**表2**のようなサーバアプリケーションを収録している。

Miracle 1.1には、Webブラウザを使ってLinuxを設定するHDE Linux Controller 2.0 Express Edition、Oracle8i Enterprise Edition R8.1.7（試用版）、サーバ構築用のマニュアル、30日間のインストールサポートが付属する。

このほか、Miracle 1.1とは別に販売されているMiracle Linux for SambaとMiracle Linux for PostgreSQLの機能もすべて備えているので、Oracleのプラットフォームとしてだけでなく、低コストのデータベースサーバやPDF文書を作成できるファイルサーバとしても機能する。

オラクルプラットフォームに最適

Miracle 1.1のGNOMEのメニューには、Oracleインストール用のウィザードが登録されいる（**画面1**）。このORANAVIというインストールウィザードを使えば、Oracleを簡単にインストールできるので、セットアップの段階で時間を浪費せずに済むだろう。

また、Miracle 1.1のカーネルは、バージョン2.2.18ながら、ReiserFSやext3といったジャーナリングファイルシステム、LVM（Logical Volume Manager）、Raw I/O、大容量メモリの使用などにも対応して、Oracleプラットフォームとしての足回りも十分だ。

このほかにも、PHPを使ったApacheとOracle／PostgreSQLの連携が検証済みなので、Webデータベースの開発をすぐに始められるようになっている。

Apache 1.3.19
Samba 2.0.7-ja-2.2
PostgreSQL 7.0.3
PHP 3.0.18-ja-2
qmail 1.0.3
ProFTPD 1.2.1
WU-FTPD 2.6.1

表2　Miracle Linuxに収録される主なサーバアプリケーション

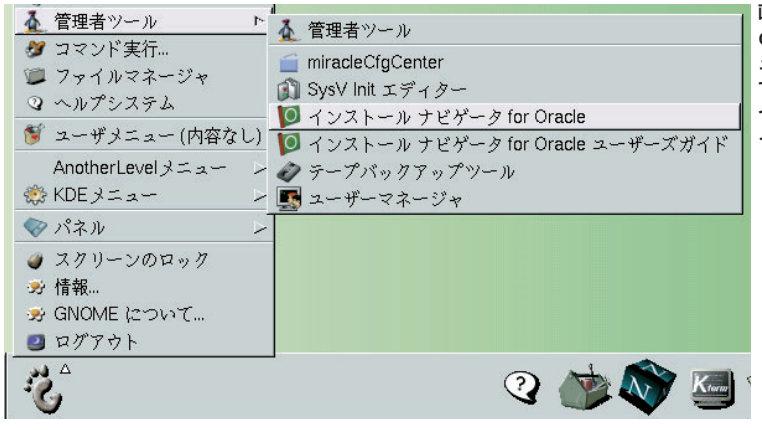
Kylixバンドル版もリリース

ミラクル・リナックスはStandard Editionに商用ソフトをバンドルした、Miracle Linux with Oracle8i/Kylixも発売した。

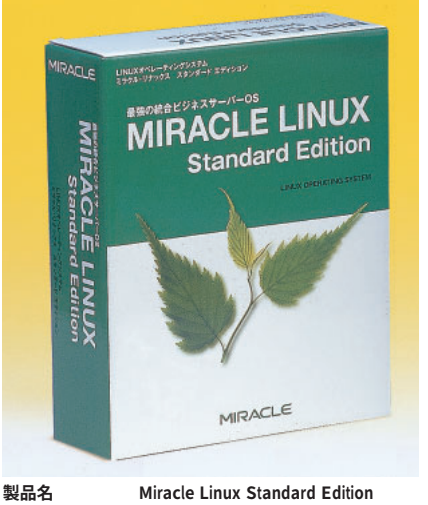
この製品には、Oracle8i Workgroup ServerとKylix Server Developerがバンドルされるので、本格的にビジュアルなデータベース開発を決めているユーザーは、こちらの製品を検討するのもよいだろう。

Oracle8i R8.1.5 Workgroup Server for Linux
Oracle8i R8.1.6 Workgroup Server/Enterprise Edition for Linux
Oracle8i R8.1.7 Workgroup Server/Enterprise Edition for Linux
Oracle Internet Application Server 8i R1.0
Oracle9i Application Server R1.0.2、R1.0.2.1

表1　Miracle Linuxでサポートされるオラクル製品



画面1　GNOMEのメニューに登録されているOracleのインストールウィザード



製品名	Miracle Linux Standard Edition Version 1.1
価格	5万円
問い合わせ先	ミラクル・リナックス株式会社 03-5562-8300 http://www.miraclelinux.com/

Distribution ▶▶▶

Kondara MNU/Linux Mary RC1公開

Kondara MNU/Linuxの次期バージョンとなる、Kondara MNU/Linux Mary RC1がリリースされた。

Maryは現在RC1フェイズまで進んでおり、次期バージョンリリースに向けて最終的なバグフィックスを行っている段階だ。

Kondara Mary RC1は、カーネル2.4.4、glibc2.2.2、XFree86 4.0.3を採用しており、標準でIPSecにも対応している。

また、デスクトップ環境には最新のGNOME 1.4、KDE 2.2.0、WebブラウザMozilla 0.9を採用しており、サーバ用アプリケーションも、BIND（9.1.2）、Squid（2.4）、OpenSSH（2.9）、PostgreSQL（7.0.3）などと、こちらも最新バージョンが収録されている。

Kondara Mary RC1では、現行バージョンのKondara 1.2で採用されているFTPサーバのWU-FTPD、メールサーバプログラムのsendmail、インターネットスーパーサーバのinetdが、それぞれProFTPD、Postfix、xinetdにリプレイスされた。

しばらくバージョンアップされなかったKondaraは、最新バージョンがリリース間近となっている。

Kondara MNU/Linux（http://www.kondara.org/）

Turbolinux Monza Developer's Release公開

ターボリナックスジャパンは、デスクトップ版の次期バージョンとなるTurbolinux Monza Developer's Releaseを公開した。

Monza（モンツァ）は、2.4系のカーネル、glibc2.2.3、XFree86 4.0.3などを基本システムとして採用している。また、GNOME 1.4とKDE 2.1の採用で、デスクトップ環境も一新されている。

このほかにも、MonzaはUSBのCD-ROMドライブやモデム、デジタルビデオ編集（IEEE 1394）、テレビチューナーなど、一般家庭で需要の大きな用途について特に強化されている。また、IPv6やADSL接続にも対応して、ネットーワークまわりもチューニングが進んでいる。

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

Plamo Linux 2.2リリース

こじまみつひろ氏が中心となって、Slackwareをベースに独自に日本語拡張を行っているPlamo Linuxの最新版Plamo Linux 2.2がリリースされた。

Plamo Linux 2.2は、カーネル2.2.19、glibc 2.2.2、XFree86 3.3.6を採用している。カーネルはジャーナリングファイルシステムReiserFSに対応済みだ。また、Plamo LinuxにはNEC PC98版も用意されており、こちらで採用されているカーネルはバージョン2.4.3である。

今回のバージョンアップは、既知の問題解決と、環境設定ファイルのチューニングが主なものである。Plamo Linux 2.2のCD作成用ISOファイルは、（ftp://ftp.kddlabs.co.jp/.7/Linux/packages/plamolinux/CD-Image/）などからダウンロード可能だ。

Plamo Linux（http://www.linet.gr.jp/~kojima/Plamo/）

ミラクル・リナックス、今後のロードマップを発表

Miracle Linuxを開発するミラクル・リナックスが、今後の開発方針と新製品のロードマップを発表した。

世界中で200ものディストリビューションが乱立するなかで、ミラクル・リナックスは、今後Red Hat Linux用とTurbolinux用のソフトウェアが動作するプラットフォームとして開発を進めていくという。

また、FSGやLPIが目指しているLinuxの標準化をサポートして、Linux関連の資格を統一することで、Linux業界を盛り上げていく方針も打ち出している。

ミラクル・リナックスは、Oracle対応製品だけでなく、Sambaサーバなどに特化したサーバ用ディストリビューションの開発にも力を入れており、今後もメールサーバやWebサーバに特化したディストリビューションをリリースしていく予定だという。

Miracle Linuxは、システムの安定性にこだわり、2.4カーネルがリリースされた現在でも、2.2系のカーネルを採用しているが、2.4カーネルの安定性が実証され次第、Miracle Linuxにも採用して、今年の下半期をめどにバージョン2.0としてリリースする予定だ。

このほかにも、2.4カーネルを採用したMiracle Linux 2.0からは、Intelが開発する64ビットCPU、Itaniumに対応した製品もリリースするなど、ハイエンド市場への対応も進めていく。

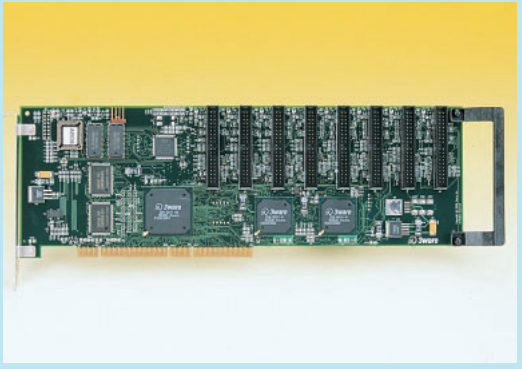
ミラクル・リナックス（http://www.miraclelinux.com/）

Products

52 64ビットPCIに対応したハイエンドIDE RAIDカード
Escalade 3W-7800

56 複数台のLinuxシステムを1台のサーバで一括統合管理
Caldera Volution

64ビットPCIに対応したハイエンドIDE RAIDカード



Escalade 3W-7800

ハイエンドのIDE RAIDカードとして知られる3ware社のEscaladeに、新たに3W-7000シリーズが加わった。64ビットPCI、Ultra ATA/100対応など既存の製品に望まれていた機能が盛り込まれている。

製品名	Escalade 3W-7800
価格	評価版のため、販売はされない
問い合わせ先	株式会社アスク TEL 03-5215-5650 http://www.ask-corp.co.jp/

Escalade 3W-7800は、3ware社が開発したIDE RAIDカードである。同社は今までにも3W-6000（6200、6400、6410、6800）シリーズというIDE RAIDカードを発売していたが、3W-7800はその上位に位置するハイエンド製品である。

なお、今回紹介する3W-7800は、評価用に作られた製品であり、本製品がそのまま発売されることはない。代わりに3W-7810という製品が7月以降に出荷予定である（店頭予想価格は、7万円前後）。3W-7810と3W-7800の差異は本体のサイズで、3W-7800のフルサイズから、一般的なハーフサイズに変更される。そのほかのスペックは、すべて共通だ。

ハードウェア

3W-7800の最大の特徴は、64ビットPCI(33MHz)に対応していることだ。このため、32ビットPCIを利用する一般的な製品の2倍に相当する最大266Mバイト／秒の帯域を利用できる。ハードディスクの性能も大きく向上しており、3W-7800のようなハイエンド製品では、64ビット化は必須ともいえるだろう。

64ビットPCIをサポートしたマザーボードやPCは、数が限られるうえにかなり高価だが、3W-7800を利用するユーザーにとっては許容範囲であろう。現在のところ、64ビットPCIが利用できるマザーボードには、ServerWorks社のServerSet IIIシリーズのチップセットが多く用いられている。

パッケージの内容は、**表1**のとおりだ。必要なものがすべて含まれており、あとはハードディスクとケースを用意するだけでよい。

ボード上に全部で8個のIDEコネクタを持ち、各コネクタに1つずつのハー

ドディスクを接続する。最新の規格であるUltra ATA/100に対応している。利用できるRAIDは、以下の4方式だ。

RAID 0（ストライピング）
RAID 1（ミラーリング）
RAID 10（RAID 0＋RAID 1）
RAID 5

そのほかにJBOD（Just Bunch Of Disks）にも対応している。JBODは、複数のディスクを単一のディスクとして用いるものだが、RAID 0のように並行しての読み書きは行わない。

RAIDの設定は、起動時にホットキー（Alt＋3）を押し、BIOSを呼び出して行う。インターフェイスはシンプルで、用いるハードディスクを選択し、RAIDの種類を選択していけばよい。

3W-6000シリーズでは、XILINX社のFPGAをRAIDコントローラ／ディスクコントローラとして用いていたが、3W-7800では、それぞれ専用のチップを利用している（**写真1**、**2**）。またEscaladeの特徴であるSRAMを用いたキャッシュメモリは、3W-7800でも採用されており、合計10Mビットの SRAMが搭載されている。

「IDE RAIDカード」と称した製品は数多く売られているが、RAID処理の一部をCPUで行うソフトウェアRAID形式が大部分である。Escaladeシリーズは、OSからは1台のディスクとして扱えるハードウェアRAIDであり、ドライブのホットスワップ、ホットスペアもサポートしている。安価なIDEハードディスクを用いて、本格的なRAIDを実現できる製品である。

Escalade 3W-7800本体
マニュアル類
ドライバ（CD-ROM×1、FD×5）
80芯IDEケーブル×8
電源分岐ケーブル×4

表1　パッケージ内容

ソフトウェア

対応するOSは、Windows 98／Me／NT 4.0／2000、Red Hat Linux 6.1／6.2／7.0、SuSE Linux 6.3／6.4／7.0であり、ドライバが付属している。

バージョン2.2.15以降のLinuxカーネルには、3W-6000／3W-7000シリーズ共通のドライバが標準で含まれているため、そのほかのLinuxディストリビューションでも利用可能であろう（動作保証はない）。

また、管理・運用のためのユーティリティである“3DM Management Utility”が付属している。これを用いると、RAIDボリュームやRAIDを構成する個々のハードディスクのモニタリングや、マシンを運用しながら、ディスクの追加／取り外しなどが行える。インターフェイスはWebブラウザを利用しており、ネットワーク上のほかのマシンからもアクセス可能だ。

ベンチマーク

3W-7800の性能を確認するために、64ビットPCIをサポートするPCを用意し、RAID 0構成時の転送速度を測定した。実験用PCのスペックは、表2のとおりだ。

なお3W-7800は評価版のハードウェアであり、今回の結果が実際に市場に出る製品（3W-7810）の性能を表しているわけではないことをお断りしておく。

“/”（ルート）パーティションには、マザーボード上のプライマリIDEコネクタに接続したMaxtor 34098H4を用いた。またRAID環境用には、Seagate ST330620A（Barracuda ATAIII）を4台用意した。

ベンチマークソフトには、Bonnie++ 1.01c（最新版）を用いた。Bonnie++は、POSIX標準のC関数を用いてファイルシステムにアクセスし、その転送速度とCPU使用率を測定する。単一のファイルを用いた転送速度の測定以外に、多数のファイルを生成する測定項目（ファイルシステムの性能測定に利用）がある。

今回はディスクやインターフェイスの性能に深く関係する、ブロック単位のシーケンシャル読み出し／書き込みの項目を利用した。

写真1　RAIDコントローラ
型番は200-0017-00。

写真2　ディスクコントローラ
型番は200-0016-01。1チップで4ポートをサポートするので、2個搭載されている。

マザーボード	SUPERMICRO SUPER 370DLE
チップセット	ServerWorks ServerSet III LE
CPU	Intel Pentium III 933MHz（FSB 133MHz）
メモリ	PC133 SDRAM（Registered、ECC）256Mバイト
ハードディスク	Maxtor 34098H4（DiamondMax VL40）40Gバイト Seagate ST330620A（Barracuda ATAIII）30.0Gバイト×4
グラフィックスカード	ノーブランドS3 ViRGE VX
OS	Red Hat Linux 6.2J
カーネル	2.2.14

表2　テストに使用したPCのスペック

Bonnie＋＋は多くのコマンドラインオプションを持つが、通常は、テスト領域のサイズ(Mバイト単位)、測定するディレクトリ、そしてログファイルの名前を以下のように指定する。

```
$ bonnie++ -s 600 -d /raid >test
```

上の例では、/raidディレクトリに600Mバイトのファイルを作成して、テストを行い、結果をtestというファイルにリダイレクトしている。キャッシュの影響を受けないように、ファイルのサイズを実メモリの2倍以上の600Mバイトに指定した。測定は3回行い、平均値を測定値としている。

付属していたCD-ROMに含まれるドライバ（1.02.00.004）では、動作が不安定だったため、3ware社のWebサイトから最新版のドライバ（1.02.00.006）を入手し利用した。

64ビットPCIの効果あり

グラフ1、2は、それぞれ3W-7800を実験機の32ビットまたは64ビットのPCIスロットに挿した場合の結果である。ディスク1台ではRAIDが構成できないため、1台での計測値は実験機とは別のマシンで測定した結果である。実験機のマザーボード上のIDEインターフェイスを利用しなかったのは、ServerSet Ⅲ LEがUltra ATA/33までしかサポートしていないからだ。グラフからもわかるとおり、今回用いたハードディスクBarracuda ATAⅢは、単体での転送速度が33Mバイト／秒を越えている。

書き込み速度については、ディスク2台のストライピングで、ディスク1台の場合のほぼ2倍の性能が得られていることがわかるだろう。3台以上になると、32ビットPCIでは頭打ちになってしまう。それに対して64ビットPCIでは、傾きはゆるやかになるが4台まで増やしても転送速度が向上していく。ディスク4台では、32ビットPCIの理論上の転送速度（133Mバイト／秒）を超えており、64ビット化の効果が十分得られている。

一方、読み出し速度については、どちらの場合もかろうじてディスク1台の時よりは速いという結果に終わった。特に32ビットPCIの場合には、台数が多いほど速度が低下した。原因は不明だが、評価版のハードウェアでもあり、ドライバの熟成もまだまだ十分でないためと考えられる。

3DM Management Utility

EscaladeシリーズがほかのIDE RAIDカードと一線を画すのは、ハードウェアの性能だけでなく、強力なユーティリティを備えているからだ。もちろん3W-7800にも、3ware社の3DM Management Utilityという管理ソフトウェアが付属する。Windows版だけでなくLinux版も用意されており、Webブラウザを使ってRAIDボリュームモニタリングや、メンテナンスを行うことが可能である。

このプログラムはデーモンとして動作しており、3W-7800のデバイスドライバと通信することで各種の機能を実現している。cshスクリプトで書かれたインストーラを起動すると、次に示す質問が表示されるので、環境に合わせて答を入力する。

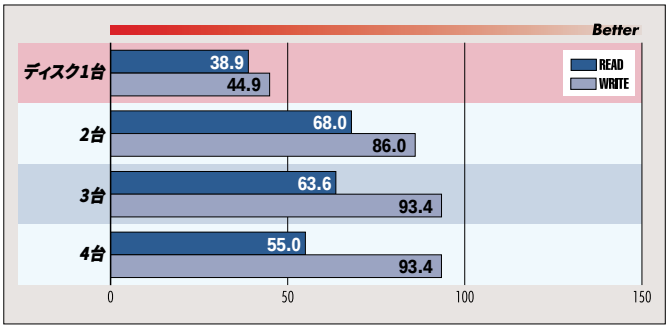
- **イベントをメールで知らせるか？**
- **メールサーバの指定**
- **メールを送るアドレス**
- **メールの送り元**
- **イベントでビープ音をならすか？**
- **Webインターフェイスで使うポート**
- **ドキュメントのインストール先**
- **ディストリビューション**

以上で、管理ツールが自動的に動作するようになる。

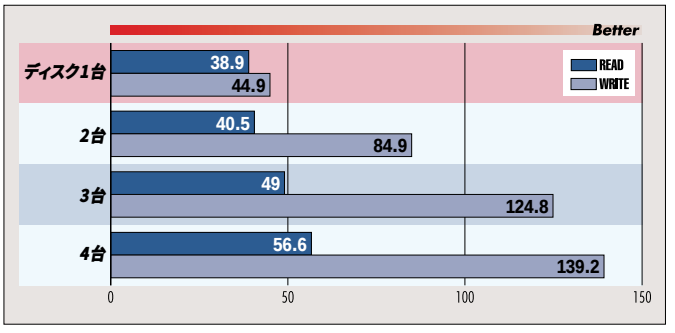
強力な管理ツール3DM

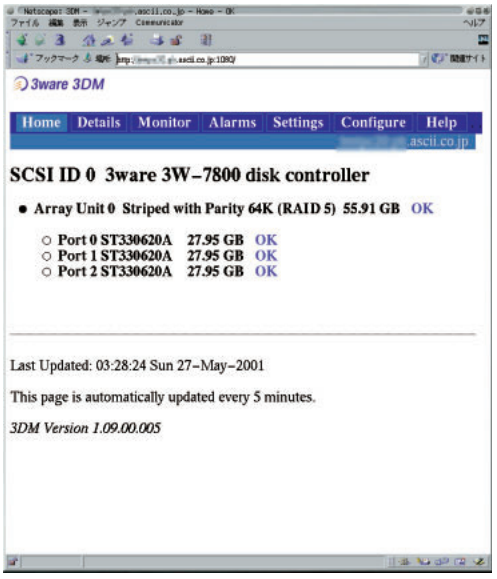
Webブラウザを用いてEscaladeが動作しているマシンにアクセスする

グラフ1　Bonnie++（Mバイト／秒）32ビットPCIスロット

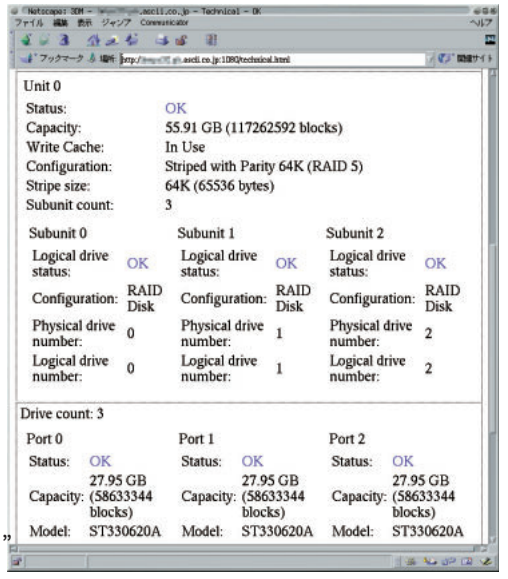


グラフ2　Bonnie++（Mバイト／秒）64ビットPCIスロット



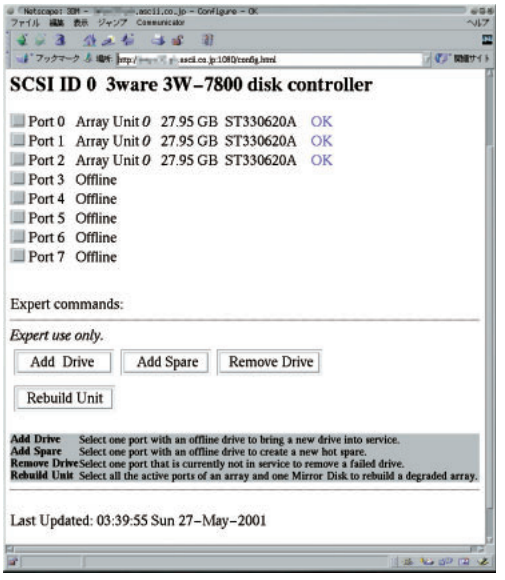


画面1　3DM Management Utility
管理ツールのトップ画面。すべてのRAIDボリュームと個々のハードディスクの情報が表示されている。デフォルトの設定では、1080番のポートが用いられる。



画面2　“ Details ”画面
RAIDボリューム、ドライブの詳細な情報の確認は、“ Details ”画面を参照する。

画面3　“ Monitor ”画面
正常に動作している場合には、このように緑色の表示が、障害が起きると赤色に変わり異常を伝える。同時にメールなどで管理者へ警告が送られる。



画面4　“ Configure ”画面
システムを動作させたままで、RAIDボリュームのメンテナンスが可能だ。

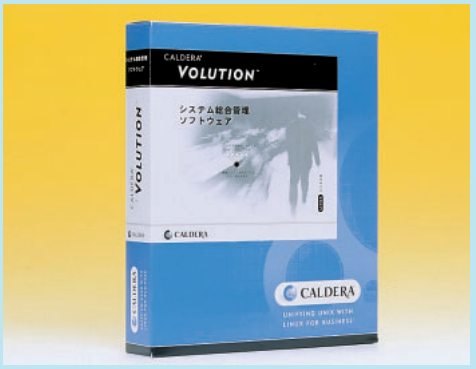
と、**画面1**のようなページが開かれる。これは、管理ツール3DMのトップページだ。ハードディスク3台を用いて、RAID 5を構成していることがわかるだろう。“ Details ”のページでは、さらに詳細な情報を表示することができる（**画面2**）。

またRAIDボリュームの状況を示す“ Monitor ”ページは、90秒ごとに自動更新されるので、まさに“ モニタリング ”用途に使えるだろう（**画面3**）。RAIDボリュームを構成するディスクに障害が発生すると、この画面上で警告が発せられると同時に、あらかじめ設定しておいたアドレス宛にメールで報告が送られ、イベントログにも記録が残る。

このツールの目玉となるのが“ Configure ”画面だ（**画面4**）。この画面から、RAIDボリュームへのディスクの追加／削除や、スペアディスクの追加が行える。さらに、問題の生じたディスクを交換したあとに必要な、RAIDボリュームの再構築もこの画面から実行できる。

障害発生後にRAIDボリュームの再構築は、BIOS画面からも行える。しかし、BIOSで再構築している間は、システムはダウンしたままだ。これに対して、3DM上から再構築を行えば、ディスクのパフォーマンスこそ低下するものの、システムのダウンタイムは最小限に抑えられる。

Escalade 3W-7800は、ホットスワップなど本格的なRAID機器と同等の機能を持ちながら、安価なIDEハードディスクを利用できるため、全体のコストを安く抑えられる。高性能／低コストを両立した製品だ。



Caldera Volution

ISPや企業／学校などで、Linuxサーバの大規模利用が増えてきている。しかし、バージョンアップやセキュリティホールへの対策のために、全部のLinuxシステムをメンテナンスするのは大変である。管理者の工数を減らすためにも統合管理ツールの導入を検討したい。

製品名	Caldera Volution
価格	50万円（10ノードライセンス）～
問い合わせ先	カルデラ株式会社 TEL 03-5486-3906 http://www.jp.caldera.com/

カルデラから「Caldera Volution」（以下Volution）というLinuxの統合管理ソフトが発売されている。Volutionは、複数、それも数十台以上といった台数の多いLinuxシステムを、一括して集中管理するためのソフトウェアである。

Volutionは、Webブラウザから管理用サーバに接続することで操作を行うようになっており、各Linuxシステムのハードウェア情報の表示や、システム負荷やプロセスの動作状況を監視したり、RPMパッケージのインストール／アンインストールをスケジューリングしながら実行することができる。

システム情報データの管理方法は、ディレクトリサービスによって実現しており、LDAP（Lightweight Directory Access Protocol）やノベルNDS eDirectory、ネットスケープiPlanetに対応する。また、システムサービス取得には、SLP（Service Location Protocol）を使用している。

そして、Volutionの特徴は、OpenLDAPやOpenSLP、XML、SNMPといったオープンスタンダードに準拠した技術を利用していることだ。

Volutionの価格は50万円で、これには、Volutionソフトウェア、ノベルのeDirectory 8.5（100ノード）、OpenLDAP、セキュアWebサーバ、および10ノード管理までのライセンスが含まれている。追加ノードのライセンスは別途購入する必要がある。

OpenLinux eDesktop 2.4
OpenLinux eServer 2.3／2.3J
OpenLinux eServer 2.3.1（アップデート版）
Red Hat Linux 6.1／6.2／6.2J
Turbolinux Workstation 6.0／Server 6.1J
SuSE Linux 6.4
Linux Mandrake 7.1（クライアントのみ）

表1　Volutionがサポートするディストリビューション

インストールはXで簡単

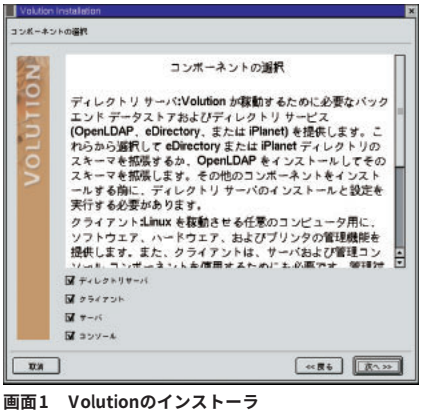
Volutionは、代表的なディストリビューションのほとんどに対応している（**表1**）。

インストールは、rpmコマンドでも行えるが、X Window System上で対話的に行うことができる。

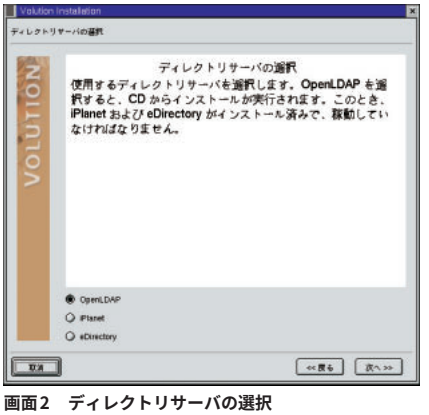
VolutionのCD-ROMからインストーラを起動し、ライセンスアグリーメントに同意すると、コンポーネントの選択画面になる（**画面1**）。Volutionの管理サーバでは、表示されている4つのコンポーネントを選び、次に進むと、ディレクトリサーバの選択となる（**画面2**）。ここでOpenLDAPを選ぶと、OpenLDAPの設定画面で、LDAPサーバの設定を行うことができる（**画面3**）。その後、ほかのコンポーネントの設定を行っていけばインストールが完了する。

各クライアントシステムには、インストーラのコンポーネントの選択（**画面1**）から「クライアント」だけを選んでインストールするか、RPMパッケージを利用してインストールすればよい。常駐するエージェントがVolutionサーバと通信することで、一元管理を実現する。

インストールが終了したら、Webブ

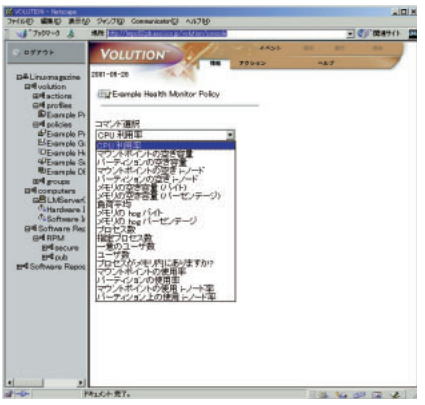


画面1　Volutionのインストーラ
インストールするコンポーネントの選択を行う。

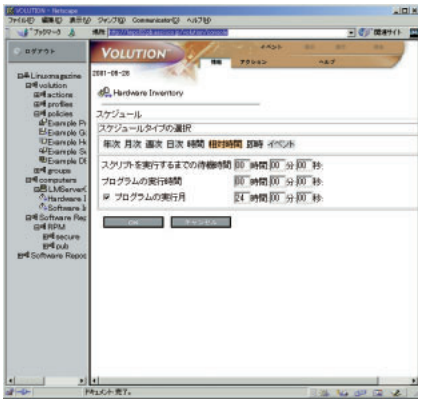


画面2　ディレクトリサーバの選択
OpenLDAP、iPlanet、eDirectoryのディレクトリサーバを利用可能。

画面3　OpenLDAPの設定
LDAPの組織名（o）、組織単位（ou）、共通名（cn）などを設定する。



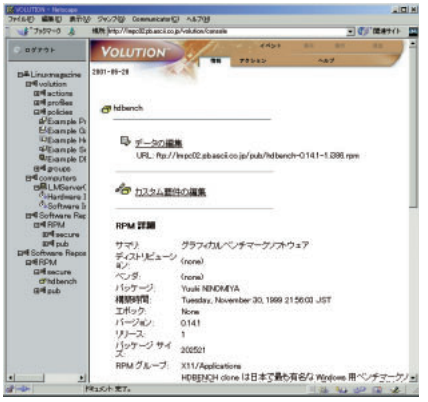
画面4　ヘルスモニタポリシーの編集
監視したい内容を選択する。指定値を超えるとログに記録することが可能。



画面5　アクションのスケジュールの設定
定義したスクリプトの実行スケジュールを指定する。

```
# cp hdbench-0.14.1-1.i386.rpm /home/ftp/pub/
# /opt/csm/bin/rpmadd -u ftp://lmpc02/pub -s "ou=Software Repository,o=Linuxmagazine"
-d LDAP:///lmpc02:389 -n "cn=admin,o=Linuxmagazine" /home/ftp/pub/*.rpm
Enter password for cn=admin,o=Linuxmagazine:
Count of RPMs to create:  1
1 - Creating object for /home/ftp/pub/hdbench-0.14.1-1.i386.rpm ..... successful
```

画面6　rpmaddコマンドでデータオブジェクトを作成



画面7　Software Repositoryに定義されたRPM
rpmaddコマンドによって作られたオブジェクトが配布可能になる。

ラウザからVolutionの管理コンソールへ、http://サーバ名/volution/consoleといったURLを指定して接続する。ユーザー名とパスワードを入力し、ログインすると管理画面が表示される。

Volutionでは、ハードウェアインベントリ情報の収集、ソフトウェアインベントリ情報、システムモニタリング、プリンタコンフィグレーションなどが行える。

たとえば、システムモニタリングを行うには、ヘルスモニタのポリシーに監視したい内容を追加する（**画面4**）。追加した内容には、通知ログ／警告ログ／注意ログのそれぞれに記録する場合の条件のしきい値を設定することができる。

続いて、そのポリシーを実行するためのアクションを定義する。アクションには、Volutionが備えている設定スクリプトや実行コマンドを設定し、実行間隔のスケジュールを指定する（**画面5**）。

こうして設定したポリシーやアクションを、実際のコンピュータ／グループに結びつけることで、決められた作業が指定したコンピュータで実行される。

RPMの自動配布／インストールも可能

Volutionでは、各サーバマシンへRPMファイルを転送し、自動的にインストールすることができる（削除も可能）。この機能を利用して、セキュリティアップデートのためのRPMパッケージを、全部（または一部）のサーバに対して一斉に適用することが可能である。

手順としては、まず配布したいRPMパッケージをサーバ機のAnonymous FTPでアクセスできる場所（/home/ftp/pubディレクトリ以下）にコピーする。次に、Volutionに含まれるrpmaddコマンドで、データオブジェクトを作成する（**画面6**）。

VolutionのWebコンソールで、［Software Repository］—［RPM］—［secure］と見ていくと、さきほど追加したパッケージが表示される（**画面7**）。あとは、特定のシステムかグループにリンクさせて、ソフトウェアの配布を行うアクションを実行すればよい。

また、WebブラウザからVolutionサーバへの接続や、LDAP情報のやり取りには、セキュリティ対策のために、SSLを利用した暗号化を行うことができる。

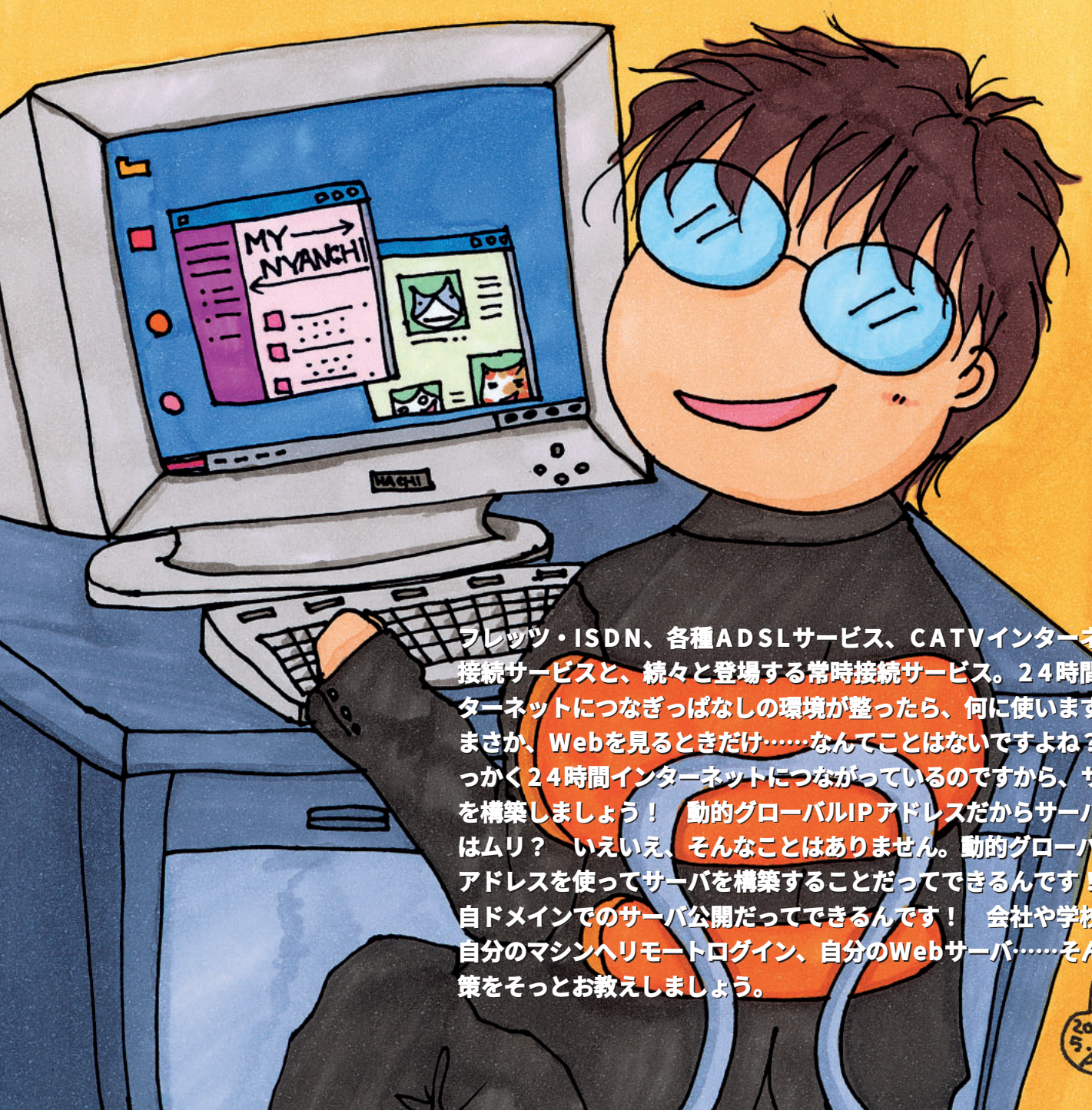
最初のバージョンということもあって、まだユーザーインターフェイスなどで不便な部分もあるが、複数のLinuxシステムを導入している場合には、バージョン管理などの手間のかかる作業を軽減させ、管理者の負担を減らしてくれる便利なツールとなるだろう。

インターネット公開術

ADSL、ISDN、CATVでサーバデビュー！ SSHで安全リモート管理、今すぐできるマイWebサーバ、ほか

文：香取精二＋渡邉利和＋しのはらひろあき＋編集部
Text : Seiji Katori + Toshikazu Watanabe + Hiroaki Shinohara + Linux magazine
illustration : Aki

サーバらくらく



フレッツ・ISDN、各種ADSLサービス、CATVインターネット接続サービスと、続々と登場する常時接続サービス。24時間インターネットにつなぎっぱなしの環境が整ったら、何に使いますか？　まさか、Webを見るときだけ……なんてことはないですよね？　せっかく24時間インターネットにつながっているのですから、サーバを構築しましょう！　動的グローバルIPアドレスだからサーバ構築はムリ？　いえいえ、そんなことはありません。動的グローバルIPアドレスを使ってサーバを構築することだってできるんです！　独自ドメインでのサーバ公開だってできるんです！　会社や学校から自分のマシンへリモートログイン、自分のWebサーバ……そんな秘策をそっとお教えしましょう。

常時接続サービスの種類と形態

常時接続サービスを利用してインターネットサーバを構築するためには、最低限動的グローバルIPアドレスが必要だ。まずは、導入する常時接続サービスを確認しよう。フレッツ・ISDNやADSLサービスの場合はほとんど問題はないが、CATVなどではプライベートアドレスしか使えない場合もある。まずは常時接続サービスが提供するサービスを確認してサーバが構築できるかどうかを確認しよう。

静的（固定）グローバルIPアドレス（以下、静的アドレス）であれば、独自ドメインサーバの構築は簡単だ。しかし、複数のユーザーでグローバルIPアドレスを使い回す動的（ダイナミック）グローバルIPアドレス（以下、動的アドレス）では、IPアドレスが接続のたびに変更されてしまう。このため、インターネット上からサーバのIPアドレスを特定することができないので、通常は独自ドメインによるサーバを構築することはできない。

しかし、動的アドレスとはいえ、IPアドレスが変更されるまでの一定期間はIPアドレスは「固定」されているわけであり、このIPアドレスを使ってサーバを特定することは可能だ。つまり、「固定」されている期間であれば、割り当てられた動的アドレスでサーバに接続することができるわけである。

この「固定」されている期間は、DHCPのリース期限と呼ばれるプロバイダが設定した有効期限である。とはいえ多くの場合、リース期限は自動的に延長され、回線が切断されるまでは同じIPアドレスが利用できる。ということは、回線を切断しなければ静的アドレスと同じように使えるのではないかと思えるが、実際には回線トラブルや、プロバイダ側から強制切断を行うこともあるので、そうそういつまでも同じIPアドレスが利用できるわけではない。しかし、少なくとも24時間単位ぐらいで考えれば、同じIPアドレスが利用できると考えていいだろう。

この「固定」の間に、割り当てられたIPアドレスをなんらかの方法で外部に公開すれば、外部からサーバに接続することができる。たとえば、接続する相手に電話で伝えたりするのもひとつの手段だ。しかしこれではほとんど実用にならないだろう。そこで、割り当てられたIPアドレスを自動的に外部に公開するような仕組みをサーバに持たせたらどうだろう。IPアドレスが変更されるたびに、自動的に外部にIPアドレスを伝えるような仕組みを用意すれば、インスタントサーバとして稼働させることができるわけだ。

この仕組みは意外と簡単に実現できる。また、このような仕組みを提供し

写真1　ADSLモデム（ブリッジタイプ）
東京めたりっく通信で提供されたADSLモデム、Arescom NetDSL 880。ブリッジタイプのADSLモデムだ。現在、東京めたりっく通信ではブリッジタイプの提供は行われていない。

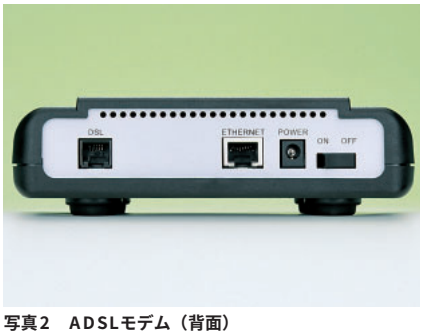


写真2　ADSLモデム（背面）
左のADSLインターフェイスをADSLスプリッタと接続する。右が10BASE-Tのポート。

ているダイナミックDNSと呼ばれるサービスも提供されている。これらについてはこの特集の最後に紹介することにするが、このような仕組みを使えば、動的アドレスしか提供していない常時接続サービスでもある程度のサーバの運用が可能になるのだ。

常時接続のサービスの種類

比較的サービスエリアが広く現実的な常時接続サービスは、現在、大きく分けて次のような3種類がある。

①フレッツ・ISDN

NTT東西が提供するISDN回線を利用した回線接続サービス。回線接続サービスのみなので、別途プロバイダとの契約が必要となる。サービスエリアが広く、回線接続料金やプロバイダ料金も低額で手軽に利用できるが、回線速度が64kbpsと遅い。

プロバイダを自由に選択できるため、選択できる付加サービスの幅が広いのも特徴。

②ADSL

NTTのアナログ回線（メタルケーブル）を利用した回線接続サービス、またはインターネット接続サービス。上りと下りの速度が違うのが特徴（通常、上りのほうが遅い）。下りで1.5Mbps程度の高速通信が可能だが、NTTの収容局からの距離が遠い場合やノイズなどで速度が得られないこともある。また、NTTの引き込み回線によっては利用できないなど、現状ではサービスエリアが少ない。

ADSLサービスには、フレッツ・ADSLのように単なる回線接続のみのサービスと、東京めたりっく通信のように、インターネット接続込みの2種類がある。回線接続のみのサービスは、別途プロバイダとの契約が必要だ。どちらか一方のサービスを提供している業者と、両方のサービスを提供している業者がある。

③CATV

ケーブルテレビの回線を使ったインターネット接続サービス。ケーブルテレビ業者のサービス地域内に限定されるため、サービスエリアが最も限定される。回線速度はケーブルテレビ業者によってまちまちだが、比較的高速なのが特徴。インターネット接続サービスなので、プロバイダを選ぶことはできない。また、グローバルIPアドレスを提供せず、プライベートIPアドレスのみを提供している場合も多い。この場合は残念ながらサーバを公開することは不可能だ。

それぞれのサービスは一長一短なので、どれが一番良いとはいえない。いずれにせよ、地域によって選択できるサービスが限られるので、自分の地域で選択できるサービスを選択することになるだろう。ただし、外向けのサーバを構築するのであれば最低限、グローバルIPアドレスが提供されていることが前提となる。この点にだけは注意してほしい。

ADSLの接続形態

ADSLはADSLモデムと呼ばれる接続機器が必要だ。このADSLモデムはブリッジタイプとルータタイプの2種類に大別される。ブリッジタイプはネットワークを中継するだけのもので、通常のモデムをイメージすればわかりやすい（**写真1**）。接続にはPPPoE（Point to Point Protocol Over Ethernet）という認証プロトコルを使うため、PPPoEクライアントとなるソフトウェアや機器が必要となる。いわゆる接続ソフトといわれるものや、ブロードバンドルータがこれに相当する。

ブリッジタイプでは、ゲートウェイ

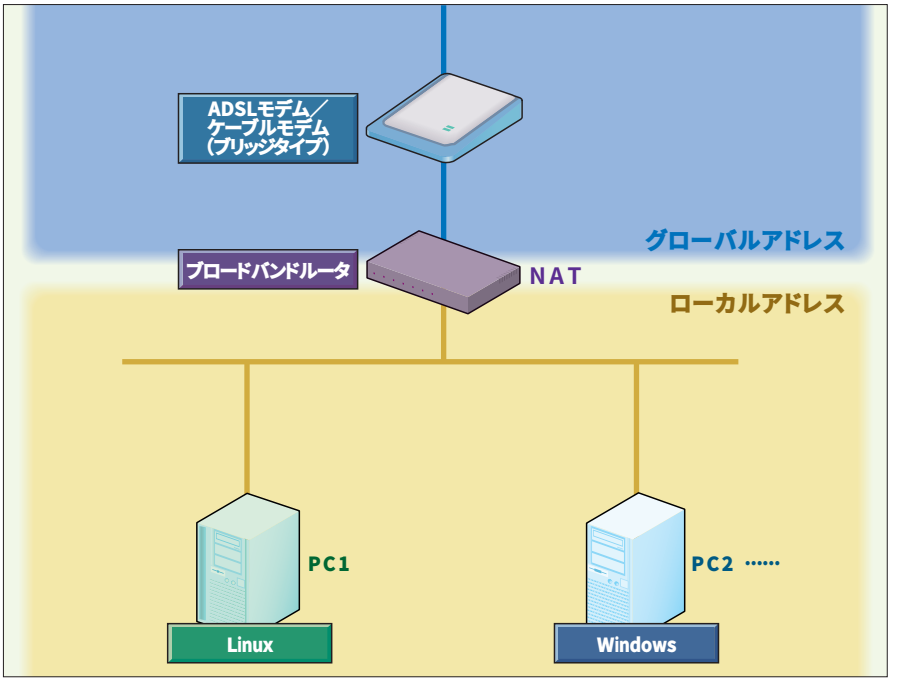


図1　ブロードバンドルータを設置したネットワーク

はPPPoEクライアントが動作するマシンか、ブロードバンドルータ（**写真3**）となる。ブリッジタイプは機能が単純なためネットワークの構成に柔軟性があるが、PPPoEのソフトウェアの設定などが煩雑なため、ブロードバンドルータを設置するほうが簡単だ（**図1**）。

ルータタイプは、ADSLモデム自体がIPアドレスを持ち、PPPoEの機能をADSLモデムが提供する。このため、ゲートウェイはADSLモデムとなる。こちらはダイヤルアップルータを思い浮かべるとわかりやすいだろう。接続はADSLモデムが自動的に行うので手軽に使えるが、ADSLモデムの機能によってはサーバを公開することができない場合もある。また、ルーティングを行うことによって、ADSLモデムの外側と内側でネットワークが分割される。この点は注意が必要だ。

ルータタイプは、IPフィルタリング機能やIPマスカレード機能を持っているため、ファイアウォールなどを簡単に構築できるというメリットもある。しかし、Linuxでファイアウォールを構築する場合など、ルータタイプのこれらの機能が逆にネットワークを構成を複雑にしてしまう（**図2**）。

なお、ブリッジタイプとブロードバンドルータを組み合わせると、ルータタイプと同等の機能を実現することができる。LinuxでPPPoEクライアントの機能とルータとしての機能を持たさせた場合も同等だ。こちらのほうが柔軟なネットワークを構築することができる（**図3**）。

ただし、ADSL業者によってはブリッジタイプまたは、ルータタイプのどちらか一方の接続方法しか提供していない場合もある。

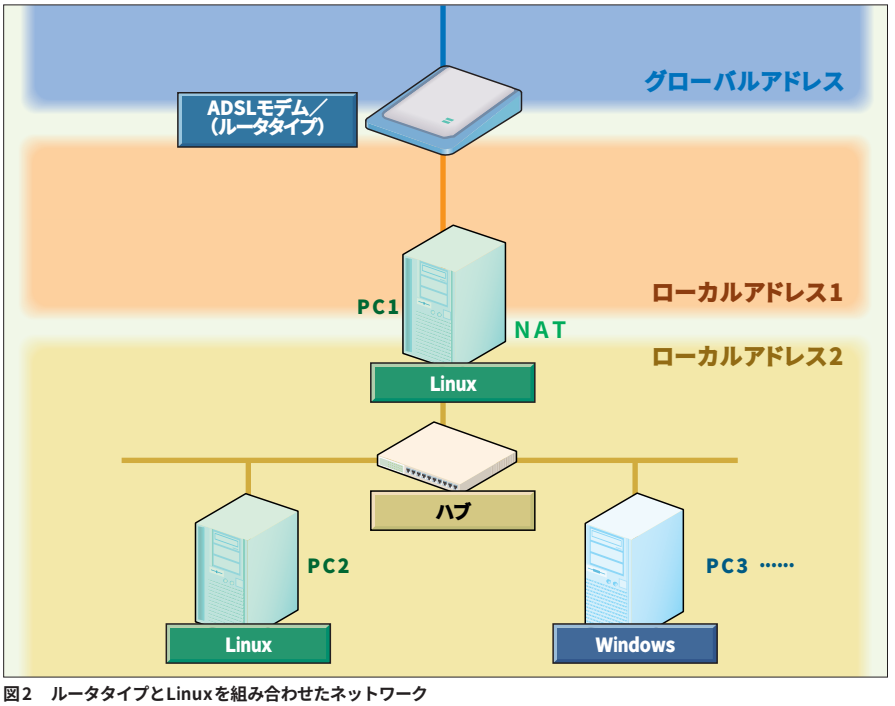


図2　ルータタイプとLinuxを組み合わせたネットワーク

■LinuxでのPPPoE接続

PPPoEのクライアントとルータの機能をLinuxで実現するためには、2枚のネットワークカードが必要となる。この場合、Linuxマシンをファイアウォールとして機能させることもできる。

Linux用のPPPoEのクライアントソフトは、ADSLの回線業者やプロバイダから提供されるものを使用すればいいが、提供されない場合はRoaring Penguin Software（http://www.roaringpenguin.com/）がフリーで公開しているRP-PPPoEなどを使う。

RP-PPPoEはRPMパッケージで提供されているため、インストールも非常に簡単だ。セットアップもadsl_setup

写真3　ブロードバンドルータ
LINKSYS BEFSR81。ブロードバンドルータ（DSLルータ）は、PPPoEクライアント機能が内蔵されているため接続が簡単になる。IPアドレスの共有や、ファイアウォールとしても機能する。

写真4　ブロードバンドルータ（背面）
この機種は8ポートの10BASE-T/100BASE-TXのスイッチングハブが内蔵されている。ルータ兼ハブという構成だ。

を実行することで対話的に設定することができる。

なお、PPPoEを使う場合、IPアドレスはDHCPで割り当てられるのではなくPPPoEによって割り当てられるので、ネットワークカードの設定でDHCPでIPアドレスを受け取らないように設定しておく必要がある。設定ファイルを直接変更するのであれば、/etc/sysconfig/network-scriptsディレクトリにある、ifcfg-＊（＊部はネットワークインターフェイス名）ファイルに、

```
ONBOOT=no
BOOTPROTO=none
```

と記述しておく。

回線の接続と切断は次のように行う。

```
# adsl-start
# adsl-stop
```

システム起動時に自動的に接続するようにする場合は、次のようにして起動時にRP-PPPoEが実行されるようにしておく。

```
# chkconfig --add adsl
```

あるいは、/etc/rc.d/rc.localに次の一行を追加しておく。

```
/usr/sbin/adsl-start
```

Roaring Penguin SoftwareのWebサイトには、詳しい接続方法も公開されているので参照するといいだろう。

CATVの接続形態

CATVもADSLと同様にケーブルモデムといわれる接続機器が必要となる（**写真5**）。CATVの場合、ブリッジタイプのものがほとんどだろう。ただし、接続自体はケーブルモデムが行うので、ADSLで必要となるPPPoEクライアントのような接続のためのソフトなどは必要ない。このため、接続自体は非常に簡単だ。ハブにマシンを接続する感覚で接続することができる。IPアドレスは通常、CATV業者から割り当てられたIPアドレスをDHCPで取得し、Linuxやブロードバンドルータに割り当てて使うことになる。

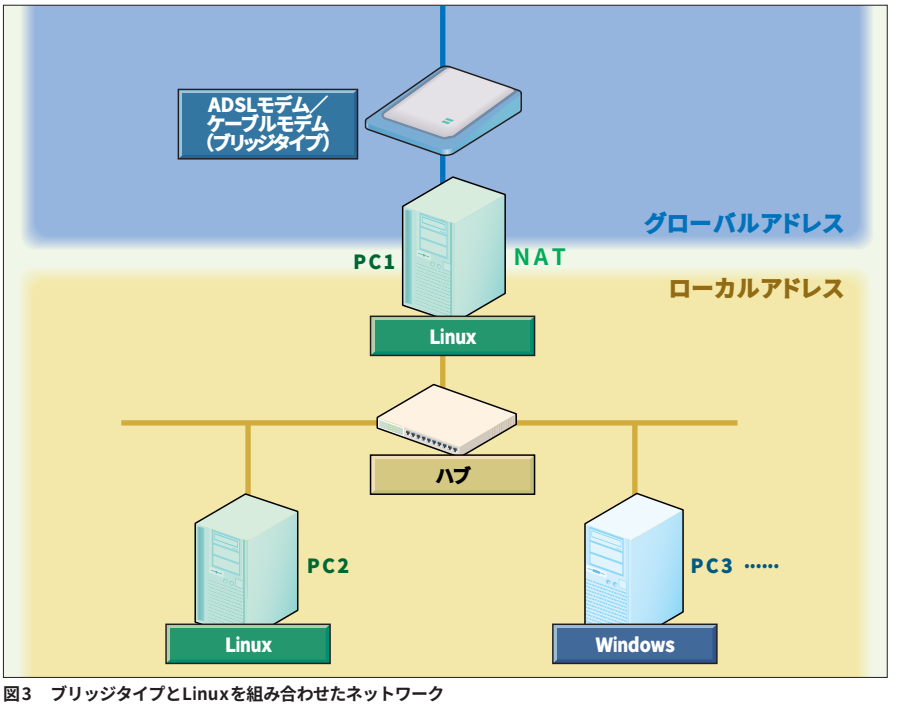


図3　ブリッジタイプとLinuxを組み合わせたネットワーク

写真5　ケーブルモデム
NECのモデキャットCM5510T。CATV用のケーブルモデム。NEC製だが、CATV業者へのOEM供給のみのようだ。

写真6　ケーブルモデム（背面）
中央が10BASE-Tコネクタ。下がRF端子で、ここにCATVのRF同軸ケーブルを接続する。上のコネクタはメーカー調整用で通常は使われない。

サーバの基本設定を押さえる

常時接続環境が整ったら、クライアントでの利用だけでなく、Webやメールなどのサービスを提供するサーバの公開を検討してみよう。このパートでは、「Webサーバ設定の基本中の基本」、「メール環境をより便利するためのちょとしたアイデア」、そして「安全な接続のためのSSHの基本設定」を紹介する。実際にサーバを公開する前のウォーミングアップとして参考にしてほしい。

インターネットサーバといえば何といってもWebサーバ。その中でも、最大のシェアを誇る、世界標準ともいえるサーバプログラムがApacheだ。当然、ほとんどのディストリビューションに標準で含まれている。なお、今回は付録CD-ROMに収録したRed Hat Linux 7.1（以下Red Hat 7.1）をベースに解説するが、基本はどのディストリビューションでもあまり違いはない。

なにはともあれApache起動

ほとんどのディストリビューションでは、インストール時にApacheをインストールするかどうかを選択できるはずだ（Red Hat 7.1のインストーラではApacheという名前は直接出てこないが、「Webサーバのインストール」を選択すればApacheがインストールされる）。サーバとして使うつもりなら、最初からインストールしておこう。もちろん、RPMパッケージを使って、あとからインストールすることも可能だ。

```
# rpm -ihv apache-1.3.19-5.i386.rpm
```

最近は、サーバソフトをインストールをしても、システムの立ち上げ時に自動起動してくれないディストリビューションが増えてきた。Red Hat 7.1もこういったディストリビューションのひとつだ。これは、不要なサーバはできるだけ起動しないようにする、というセキュリティ上の考え方によるものだ。サーバソフトウェアは、システム管理者がきちんと意識して起動することが望ましいのだ。また、設定の変更をしたときなど、システムを立ち上げ直さずにサーバのみを再起動したいこともあるだろう。

まずは、Apacheの起動、停止、再起動の方法を覚えておこう。なお、起動、停止や、設定ファイルの変更などはrootになって作業する必要がある。

起動

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

停止

```
# /etc/rc.d/init.d/httpd stop
```

再起動

```
# /etc/rc.d/init.d/httpd restart
```

この中でも、特によく使うのは再起動だ。restartオプションを使えば、いったんApacheを停止し、書き換えた設定ファイルを読み込んで再起動という一連の動作を、1回のコマンド実行で行うことができる。

では、まずstartオプションでApacheを起動して実際に動作しているか確認してみよう。自分のマシン名が確定していない場合は、以下のURLにアクセスすればよい。もちろん、すでにネットワークに接続して、ホスト名、ドメイン名が確定している場合は、そのURLに直接アクセスしてもよい。

```
http://localhost/
```

ブラウザには**画面1**のようなテストページが表示されるはずだ。テストが成功したら、とりあえずstopオプションで停止しておく。Apacheのテストページが表示されるWebサーバを公開するのはかなり恥ずかしい。せめてトップページを作ってからにしたほうがいいだろう。

運用前にこれだけはしておこう

Apacheの設定は、httpd.confというテキストファイルに書かれている（Red Hat 7.1では“ /etc/httpd/conf/httpd.

conf ”)。設定の変更はこのファイルを編集して行う。

ただし、今回は面倒な設定はできるだけ後まわし。最小限の設定でWebサーバの運用を開始することにしよう。

■webmasterのアドレス

Webサーバを公開する前にこれだけはやっておきたいのが、Web管理者のアドレスの設定だ。このアドレスはサーバの動作に何らかの問題が発生したときに、ApacheがWebブラウザに対して送信するもの。サイトの訪問中にエラーが発生した場合、見ていた人がこのアドレスに対してメールを送ってくることもあるかもしれない。初期設定は“ root@localhost ”になっているから、要修正だ。これは“ ServerAdmin ”というキーワードで設定する。httpd.confをエディタで検索し、たとえば次のように修正しておこう。

```
ServerAdmin webmaster@example.com
```

■データを置く場所

データ用のディレクトリの初期設定は、Red Hat 7.1では“ /var/www/html/ ”となっている。“ http://www.ascii.co.jp/ ”などとURLを指定すれば、www.ascii.co.jpというマシンの/var/www/html/に置いたファイルを見ることができるわけだ。これは、httpd.conf内の“ DocumentRoot ”というキーワードで指定されている。

/var/www/html/は、最初はrootのみが書き込み可能になっている。自分だけで使うマシンなら、常にrootで作業するというのもありかもしれないが、インターネットに公開するサーバでは、お勧めできない。

そこで、root以外でもデータの更新をできるようにしよう。Webの管理を行うグループを作って、そのグループに属しているユーザーに書き込みの権限を与えるのがよいだろう。

ここでは仮に、Web管理者のグループをwebadmin、管理者のユーザー名をuser1とすると、コマンドは以下のようになる。

```
# groupadd webadmin
# usermod -G webadmin user1
```

複数の人間がWebを管理するなら、それぞれのユーザー名を指定して、usermodコマンドを繰り返せばよい。ちなみに“ G ”を小文字にしてしまうと、user1のもともとのグループがwebadminに変更されてしまう。大文字のGを使えば現在のグループはそのままに、さらにwebadminにも所属できるようになる。打ち間違えないようにしよう。

次に、データディレクトリのグループをwebadminにして書き込み権限を与える。

```
# chgrp -R webadmin /var/www/html
# chmod g+sw /var/www/html
```

これでwebadminグループに属するユーザーは、Webのデータを自由に変更できるようになったはずだ。

■ファイルのパーミッションにも注意

さらに、気をつけてほしいのが、実際にWeb公開用のディレクトリに置くデータファイルのパーミッションだ。これらの場所に置かれたファイルを読み込んで、Webブラウザに送るのはApacheである。つまり、Apacheがこれらのファイルを読める状態になっていないといけない。

Apacheがこれらのファイルを読めない場合、ブラウザには「Forbidden」と表示されるはずだ。そういった場合は、そのファイルをApacheが読むことができるようにパーミッションを変更する。

```
# chmod o+r file
```

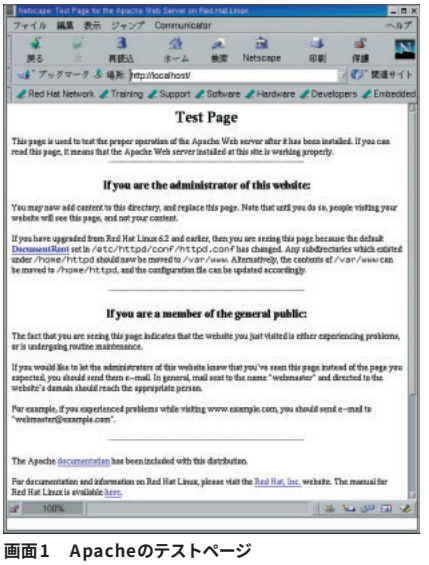
また、ディレクトリに実行属性を与えることも忘れないようにしよう。

```
# chmod o+x directory
```

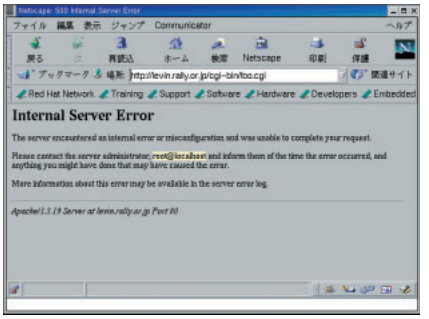
さて、これでとりあえず、Webを公開する準備は整ったはずだ。

システムの設定

ここまでの設定が完了して、Webコ



画面1　Apacheのテストページ



画面2　エラー時には通知先のメールアドレスが表示される

ンテンツが準備できたら、いよいよApacheの起動。ただ、Linuxを再起動するたびに、“ httpd start ”コマンドを使ってApacheを起動するのも面倒な話だ。そこで、システムのブート時に、Apacheも自動的に起動するようにしておきたい。

Runlevelに応じた起動スクリプトを設定するわけだが、ここではコマンドラインツールの「chkconfig」を使って設定してみよう。

```
# chkconfig --level 35 httpd on
```

上記のコマンドで、Runlevel 3（テキストログイン）とRunlevel 5（Xからのグラフィカルログイン）でのブート時にApacheが自動的に起動するようになる。

CGI、SSIを使うには

自分でサーバを持ったら、使ってみたいのがやはりCGIやSSIだ。プロバイダによっては、CGIやSSIの使用に制限があったりして、自由にならないことも多い。自分のサーバならば、人に迷惑をかける心配なく安心してCGIを動かすことができる。もっとも、サーバをダウンさせるようなことがあったら、自力で復旧しなければならないので、やっぱり気をつけるにこしたことはないけれど。

■CGIの基本

CGI（Common Gateway Interface）は、掲示板などでよく使われる、Webサーバ上で動作するプログラムだ。本来はサーバ上で動くプログラムであれば、シェルスクリプトであろうが、C言語であろうが、なんでもかまわないのだが、Perlスクリプトが使われることが多い。

▼リスト1　CGIのサンプルシェルスクリプト－sample.cgi

```
#!/bin/sh
echo content-type: text/html
echo ""
echo "<html><body>"
date
echo "</body></html>"
```

CGIを利用するには、プログラムを置くディレクトリやファイル名などを、httpd.confによって設定しておかなければならない。Red Hat 7.1の初期設定では、CGIプログラムを置けるディレクトリは“ /var/www/cgi-bin/ ”となる。

リスト1は、時刻を表示するシェルスクリプトの例だ。ごく簡単なものなので、リストを見てもらえば、だいたいの意味はわかるだろう。ではまず、ファイルを作って、以下のように実行属性をつけて、実行してみよう。

```
# chmod +x sample.cgi
# ./sample.cgi
```

現在の時刻を含んだHTMLデータが出力されるはずだ。

このスクリプトファイルを/var/www/cgi-bin/にコピーすれば、CGIの準備は完了……ときたいところだが、その前にApacheがこのファイルを実行できるように設定することを忘れないように。

```
# chmod o+x sample.cgi
# cp sample.cgi /var/www/cgi-bin
```

準備ができたら、Webブラウザから“ http://localhost/cgi-bin/sample.cgi ”にアクセスしてみよう。**画面3**のように表示されればOKだ。

CGIの基本として、次の3つのポイントを押さえておこう。

①実行の結果として、表示するべきデータを出力すること
②ApacheがCGIプログラムを実行できるようになっていること
③設定されたディレクトリにCGIプログラムが存在すること

■SSIの基本

SSI（Server Side Include）は、サーバ側が送信するデータの中に別のファイルの内容やプログラムの出力などを挿入する機能で、アクセスカウンタなどでよく使われる。

CGIの結果を挿入するには、以下のような行を追加する。

```
<!--#exec cgi="cgi_file" -->
```

リスト2は、先ほどのCGIの内容を取り込むために、SSIを使ったHTMLファイルの例だ。なお、Red Hat 7.1ではSSIを使う場合、ファイル名の最後を“ .shtml ”にしなければならない点に注意。

このファイルを、/var/www/html/の下にコピーして、Webブラウザからアクセスしてみよう。CGIの出力内容を取り込んで表示されているのがわかるだろう（**画面4**）。

CGI、SSIの設置のポイント

CGIやSSIはサーバ側でプログラムを動作させる関係上、セキュリティホールになりやすい。プログラミングに自信がない人は、インターネット上で公開されている定評のあるものを使うほうが無難だ。もっとも、慣れないうちは既存のCGIプログラムさえも、設置に戸惑うものだ。CGIの設置の第一歩は、当然だがドキュメントをよく読むこと。これにつきるのだが、一般的な

perlのCGIを設置するポイントをいくつか挙げておこう。

①プログラムの1行目のPerlのパス

プログラムの1行目にはPerlのパスが書かれている。“ #!/usr/local/bin/perl ”となっていることが多いが、ほとんどのLinuxディストリビューションでは、これを“ #!/usr/bin/perl ”に書き直す必要がある。

②変数の内容の修正

データファイル名などは、変数として設定される場合が多い。Perlでは変数の頭には“ $ ”が付くので、下の例のように変数に値を代入している行のチェックを忘れないようにしよう。

```
$data_file = "/path/to/datafile.txt";
```

③修正ミス

変数の内容を書き換えるときに、うっかりダブルクォート（"）や行末のセミコロン（;）を消してしまっているなどというのも、ありがちなミスだ。こういったミスはなかなか発見できないことが多いので気をつけたい。

ユーザー用のディレクトリ

いままではWebデータを、/var/www/の下にまとめるという前提だったが、ユーザーごとにデータディレクトリを用意することもできる。

ユーザー用のデータディレクトリは、各ユーザーのホームディレクトリの“ public_html ”が初期設定になっている。“ http://www.ascii.co.jp/~foo/ ”などと指定すると、www.ascii.co.jpというマシンのユーザーfooさんのデータ（/home/foo/public_html/以下にあるデータ）が見えるようになる。

ユーザーごとにpublic_htmlというディレクトリを作成し、そこにファイルを置けば、それだけでOK。ただし、ファイルとディレクトリのパーミッションはこれまでと同様、Apacheが読めるようにしておくのを忘れないこと。

なお、このユーザー用のディレクトリでもCGIやSSIを使いたい場合には、それぞれ以下の設定をhttpd.confの最後に加えればよい（実際に追加する記述は**リスト3**を参照）。

- **public_html/cgi-bin/の下にあり、名前の最後が“ .cgi ”になっているファイルをCGIファイルとして使えるようにする**
- **public_html/の下でSSIを使えるようにする**

Apacheの活用情報

ここまでに行った設定で、とりあえずはApacheを立ち上げてWebページを公開することができるはずだ。

▼リスト2　SSIのサンプルHTMLファイル－ssi.shtml

```
<html>
<body>
Now
<br>
<!--#exec cgi="/cgi-bin/sample.cgi"
-->
</body>
</html>
```

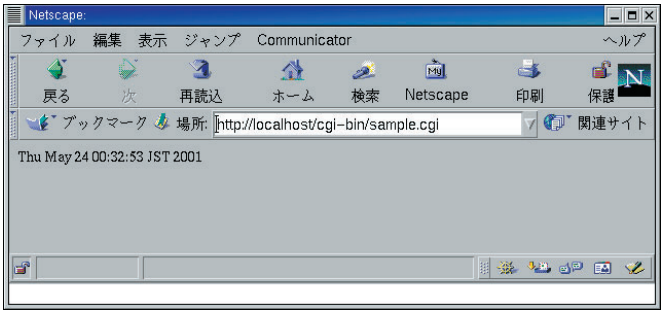
▼リスト3　httpd.confへの追加内容

```
## CGIの設定
AddHandler cgi-script .cgi
<Directory "/home/*/public_html/cgi-
bin">
    Options ExecCGI
</Directory>

## SSIの設定
<Directory "/home/*/public_html/">
    Options Includes
</Directory>
```

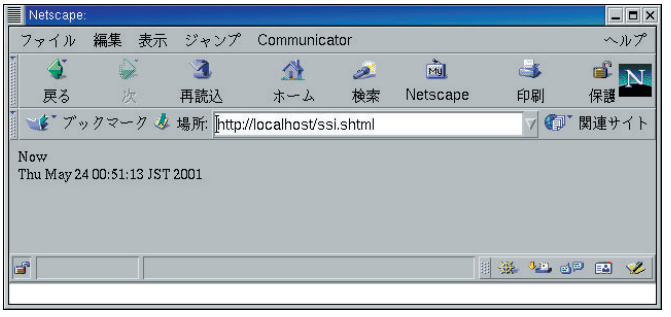
Apacheに関する、さらなる情報が欲しい場合は、Red Hat 7.1の2枚目のCDからオンラインマニュアル(apache-manual-1.3.19-5.i386.rpm）をインストールしよう。

このマニュアルは、“ http://localhost/manual/ ”というURLで読むことができる。英語の苦手な方は、株式会社インフォサイエンスで翻訳しているWeb上の日本語マニュアル（http://japache.infoscience.co.jp/）を利用するとよいだろう。ベースになっているバージョンは少し古いものだが、実用上は問題ない。

（香取精二）



画面3　CGIの実行結果



画面4　SSIの実行結果

独自IMAPサーバの運用

自前でサーバを立てるという話になれば、まず名前が挙がるのはWebサーバかメールサーバというところだろう。いずれも、個人でも利用頻度が高いため、「自分の好きに使えるサーバがあればさぞ便利だろう」と誰でも考えるはずだ。

LANにはメールサーバがあってほしい？

本誌読者ともなれば、当然自宅にはLANが敷設されており、常時複数台のマシンが稼働していることと思う。筆者宅もそういう状況なのだが、この場合、自宅にメールサーバがあれば、としばしば考える。というのも、ISPのメールサーバは複数のマシンから使う際に便利なように、と配慮してくれるほど親切ではないことが多いからだ。

筆者宅には、24時間常時稼働のマシンが1台、そのほかに比較的使用頻度の高いマシンとしてミニタワー型PCが3台とノートPCが1台ある。ユーザーは筆者1人だが、作業内容やその日の気分で5台のうちのどれか1台（または複数台）を使っているわけだ。ここで問題になるのが、メールの確認作業だ。スペースに余裕がないということもあって、ディスプレイ／キーボード／マウスは1組で、これを4台切り替え可能なKVMスイッチに接続して使っている。つまり、独立した画面を持っているのはノートPCだけで、残りの4台のマシンは随時切り替えて利用しているのだ。

切り替え自体は別段面倒な作業ではないのだが、それでもやはり、作業をいったん中断することになるため、場合によっては億劫に感じることがある。できれば、そのとき利用しているマシン上でメールのチェックができると便利だと思うわけだ。

とはいえ、プロバイダのメールサーバを利用している場合、迂闊にこういうことをすると後々困ることになる。POPでメールを受信する場合、受信したマシンのディスクにメールが保存されることになるため、複数のマシンでバラバラに受信処理を行うと、メールも各マシンに分散して保存されることになる。あとで参照する際に、どのマシンに保存してあったのかわからなくなり、あちこち探し回る羽目になったりして、不便なことこのうえない。

運用上の工夫として、いろいろな手段を試してみたが、どれもあまり具合がよくない。たとえば、1台のマシンを保存用に決め、残りのマシンはすべて参照専用として、「サーバ上にメールを残す（削除しない）」という設定でアクセスする、とかだ。この場合、保存用のマシンにメールをダウンロードする間隔が問題になる。あまり頻繁にダウンロードすると、結局参照用マシンからアクセスする前にメールが削除されてしまって意味がなくなるし、あまり間隔を長くしてしまうと今度はサーバ側にメールが溜まりすぎてしまう。プロバイダによっては、規定量をオーバーしてトラブルになることもあるだろう。

外部のPOPサーバを利用しつつ、LAN環境で便利なメール環境を構築するのはなかなか困難なのである。

独自メールサーバを立てる？

とはいえ、メールサーバも自前で持ってしまおう、というのはやや極端だ。実のところ、「あれば便利かも」という程度の発想では、メールサーバを立てたりしないほうが無難だといえる。

本気でメールサーバを立てようと思うと、独自ドメインを取得して運用することになるだろう。しかし、それにはコストもかかるし、何より運用管理が面倒だ。メールに関してはトラブルも多い。セキュリティにも配慮する必要がある。今どき中継動作はデフォルトで禁止されているのが普通だが、うっかり無用の中継を許可してしまい、スパムだのウイルスだのの配布に利用されたりすれば、世間の迷惑でもある。

こうした「メールサーバを運用する手間」を考えると、安易に独自サーバを立ち上げるのは危険でもある。特に筆者の場合、「画面を切り替える手間が惜しい」程度の惰弱な動機である。より手間がかかると容易に想像できる独自メールサーバの立ち上げには到底踏み切れるものではない。せっかくISPがメールサーバを運用してくれており、安価に利用できるだから、何も好んで苦労する必要もあるまい、というのが正直なところだ。

とはいえ、やはりPOPによるアクセスではやや自由度が低いのが気になる。そこで浮上してくるのがIMAPサーバというわけだ。

IMAPでは、メールをサーバ側に保存しておき、メールクライアントはサ

ーバ上のメールファイルを参照することになる。つまり、IMAPを利用すれば、複数のクライアントを任意に使い分けたところで、メールファイルの分散といった問題とは無縁である。

現実には、IMAPサービスを提供してくれるISPはあまり多くはない。メールをサーバ側に保存する都合上、サービス提供側の負担が重くなってしまうので、これもやむを得ないところだろう。ならば、自前でIMAPサーバを立てたらどうだろう。Linuxの場合、多くのディストリビューションではIMAPサーバが標準で含まれている。これをISPのメールサービスと組み合わせ、ISPからPOPでメールをダウンロードし、LAN内でIMAPで公開する、という構成にすればとりあえず丸く収まりそうだ。というわけで、この構成を簡単に実現する方法をご紹介しよう。

POPによるメールのダウンロード

ISPのメールサーバからメールをダウンロードするには、POPクライアントを利用することになる。とはいえ、POPクライアント独自の形式でメールを保存されては応用が効かなくなって困るので、ここは汎用性の高いツールとしてfetchmailを利用しよう。

fetchmailは、POPなどのプロトコルを利用してメールサーバからメールをダウンロードするためのツールである。ただし、メールクライアントではないので、単体でメールを読み書きできるわけではない。標準では、メールサーバからダウンロードしたメールをローカルのTCPポート25番、すなわちSMTPサーバに放り込む、という動作をする。通常、TCPポートの25番ではメールサーバプログラム（sendmailやqmail、Postfixなど）が待ち受けているので、ここから先は通常のメールと同様のメカニズムを利用して配信が行われる。POPで受信したメールをあたかもSMTPで送られてきたように見せかけるためのプログラムだといってもよいかもしれない。基本的な動作としては、一般的なメールクライアントに備わっている自動受信機能の部分だけを取り出したようなものだ、ということもできるだろう。

対外的に公開され、インターネットから直接メールを受け取ることができるメールサーバを立ち上げるのは前述のようになかなか面倒な作業となるが、LANに対してのみサービスを提供するメールサーバを立ち上げておいて、これとfetchmailを組み合わせる、という方針であれば、とりあえず今回想定している目的にかなうだろう。

なお、実は単に「ISPのメールサーバからメールを取得し、IMAPで参照できるようにする」というだけの動作が実現しさえすればよいのであれば、sendmailだのqmailだのといったいわゆるメールサーバソフトウェア（正確にはMTA）が動作している必要もない。そこで今回は、極力余分なことはせず、最小限の手間で目的の動作を実装する、という方針で臨みたい。

前提となる環境の確認

今回利用するfetchmailは、オープンソースという言葉を有名にし、『伽藍とバザール』を執筆した、かのEric Raymondが作ったソフトウェアである。たいへんよく知られたツールであるため、たいていのディストリビューションに含まれているものと思う。筆者が利用したのは、本誌6月号の付録CD-ROMに収録されたTurbolinux Server 6.5（FTP版）なので、ここで紹介する手順はこのディストリビューション上でのものとなるが、やっていることはごく基本的なことばかりなので、ディストリビューションによる違いはあまり大きくはないと思う。

Turbolinux Server 6.5ではインストールクラスとして「基本システム」、「インターネットサーバー」、「イントラネットサーバー」、「すべて」の4種類が用意されるが、筆者の場合は「イントラネットサーバー」を選択した。特に理由はないのだが、何となくLAN内部向けのメールサーバとして運用するという性格上、ふさわしいインストールクラスだと感じたため、これを選んでみた。

なお、インストール対象としたPCはCPUがAMD Athlon 650MHz、メモリ256Mバイトというもので、ネットワークカードは1枚だけ載せてある。現状LAN内で何か特別なサービスを提供しているマシンではなく、実のところTurbolinux Server 6.5を新規インストールし、今回紹介するメールシステムに関わる設定だけを行っている。

今回利用したツールは、fetchmailのほかにprocmail（MDA）とimapd（IMAPサーバ）があるが、いずれも標準でインストールされているものを使ったため、ちょっとした設定変更を行ったのみで、ソフトウェア／パッケージのインストール作業すら不要であった。

次に、前提となるメール環境について紹介しておこう。

まず、ユーザーは1人、もしくはごく少人数だと想定している。ユーザーが複数いる場合でも、ISPのメールアドレスをそれぞれ独自に所有しているものとする。ISPの1つのメールアドレスをLAN内の複数のユーザーで共有する、という利用法もあるにはあるが、この場合はメールをLAN内の適切なユーザーに振り分けるための仕掛けが必

要となり、少々作業が複雑になる。

今どきISPのメールアドレスを確保するのは月額数百円程度で済むことが多いため、妙な小細工を弄してメールアドレスを共有することにあまり需要もないだろうと思う。筆者の場合、自宅で筆者1人が利用している環境ということもあり、この前提はごく自然である。

また、ISPのメールサーバへのアクセスはPOP3を使うものとする。fetchmailはPOP2、POP3、IMAP2bis、IMAP4、IMAPrev1、ESMTPのETRN拡張など、多彩なプロトコルに対応しているが、ISPがIMAPを提供しているなら、独自のIMAPサーバを立ち上げる必要性が薄れるし、いまどきPOP2のみのサポートという例もまずないだろう。

さらに、今回はメールの参照だけを実現し、メールの送信は考慮しない。というのも、筆者はWindowsクライアント上でメールを参照することを考えているのだが、Windows用のメールクライアントでは受信メールサーバ（POP3サーバ）と送信メールサーバ（SMTPサーバ）を別々に指定できるのが一般的であり、筆者の環境では送信メールサーバとしてISPのメールサーバを指定してしまうのが一番簡単な方法だからである。

同様の理由で、メールの振り分け処理も今回は考慮していない。procmailを利用すれば高度な振り分け機能が実現できるが、筆者にとっては使い慣れたWindows用メールクライアントの機能を使って実行するほうが簡単に思えるからだ。ちなみに、筆者は主としてBecky!というメールクライアントを利用している。

実はこの仕様は「ローカルでsendmailまたはqmailを動かさないようにしよう」と考えたことから生じている。理由はたいしたことではなく、単に面倒だ、あるいは、最小構成で簡単に利用したい、という2通りの表現ができる。要は、手間を惜しんだということだ。逆に、こうしたMTAソフトウェアを動作させればメールの送信や振り分け処理なども一元化でき、すべてLinuxサーバ上で実現できることになるので、この辺りはそれぞれ趣味に応じて工夫していただきたいところだ。筆者の場合、日常的な作業をWindowsマシンで実行することが多いため、LinuxサーバはWindowsではやりにくい処理を実現するために利用する、という位置づけになっている点はご了解いただきたい。

fetchmailの設定

では、具体的な設定に移ろう。前述のとおり、Turbolinux Server 6.5ではfetchmailはあらかじめインストール済みなので、ちょっとした設定を行って起動するだけで利用できる。

fetchmailはrootが設定して起動するというよりは、個々のユーザーが必要に応じて実行する性格のツールである。というのも、ISPのメールサーバに対するアカウント名とパスワードというきわめて個人的な情報を設定する必要があるためだ。

今回の構成では、ユーザーも管理者も同一で、結局のところ1人だけ、という環境なのでこの違いはさして大きなものではないが、ユーザーが複数いる場合でも個々に設定を行い、個別に実行するのがよいだろう。

設定は、ユーザーごとに設定ファイル.fetchmailrcを作成し、これをホームディレクトリに置けばよい。各ユーザーがfetchmail（Turbolinux Server 6.5では/usr/bin/fetchmail）を起動すれば、~/.fetchmailrcを読み込んで動作する。ユーザーのホームディレクトリのファイルを暗黙で読み込むことから、fetchmailコマンドを実行するユーザーが誰かということが問題になる点に注意しよう。一般ユーザーとして設定を行ったあと、rootでfetchmailを起動する、といった手順では具合が悪いわけだ。

参考までに、筆者が作成した.fetchmailrcファイルの内容を**リスト1**に紹介しておく。ここでは、ローカルのユーザーIDが“user1”だとしている。では、内容を順次紹介していこう。

まず最初の“set daemon 600”は、デーモンモードでの起動の指定である。デーモンモードで起動すると、fetchmailはバックグラウンドで動作し、一定の時間間隔でメールサーバにアクセスし、メールが届いていればそれをダウンロードしてくる。通常は、このモードで利用するのが便利だろう。起動したあとでユーザーがログアウトしてもfetchmailは動作し続けるので、一度起動すればシステムが動いている限り再起動の必要はない。ただし、システムをシャットダウン／リブートした際には改めて起動しなくてはならないの

▼リスト1　.fetchmailrcファイルの設定例

```
set daemon 600
set postmaster user1
set no bouncemail
poll mailserver.ascii.co.jp proto pop3:
        user "userid" with password "passwd", is "user1" here wants mda
"/usr/bin/procmail -d user1"
keep
```

は、いうまでもないが。メールチェックの間隔は秒数で指定する。この例では、10分おき(600秒)を指定している。

“ set postmaster user1 ”は、エラーメールの通知先などに使われるメールアドレスの指定である。デフォルトではpostmasterが利用される。この手のツールの設定ミスで管理者にエラーが通知されるのは迷惑だろうという考えで、ここではユーザーIDである“ user1 ”を指定しているが、実のところ「ユーザーも管理者も同一で1人しかいない」という環境であれば明示的に指定しなくてもたいした違いはないと考えて差し支えないだろう。

“ set no bouncemail ”はメールの送信者にエラーメールを差し戻すという動作を禁止するためのものだ。今回の環境ではメールの送信に関しては考慮していないので、メールの差し戻しはしないでほしいためこうしてある。この場合、エラーメールはpostmasterに送られることになる。

次のちょっと長い文が、設定の核心部分になる。部分ごとに分けて内容を確認していこう。

まず“ poll mailserver.ascii.co.jp proto pop3: ”の部分で、メールサーバとアクセスプロトコルを指定している。おわかりとは思うが、“ mailserver.ascii.co.jp ”の部分はダミーであり、実際には各自ISPから指定されたPOPサーバ名を指定する必要がある。一方、“ proto pop3 ”の部分は、アクセスプロトコルにPOP3を使っている場合はこのままで大丈夫だ。このほか指定可能なアクセスプロトコルについて知りたい場合は、fetchmailのマニュアルを参照していただきたい。

次に、“ user "userid" with password "passwd", is "user1" here ”の部分だ。ここでは、ISPのメールアカウントとパスワードを、ローカルのユーザーIDに対応づける。ISPのメールサーバの接続情報が、

- **POPアカウント名：userid**
- **POPパスワード：passwd**

となっているという想定だ。一方、ローカルのメールアカウントはログインIDと一致させておく。

残る“ wants mda "/usr/bin/procmail -d user1" ”は、今回の構成で利用するやや特殊な設定だ。前述したように、標準の設定ではfetchmailはリモートのメールサーバからメールをダウンロードしたあと、ローカルホストのTCPポート25番に接続してそのメールを転送する。しかし、mdaオプションを利用すると、TCPポート25番への転送の代わりに、指定したコマンドにデータを渡すことができる。ここでは、procmailに-dオプションをつけたものを指定している。

procmailは、MDA(Mail Delivery Agent)と呼ばれるソフトウェアで、sendmailからメールを受け取ってローカルでの配送を実行する。このほか、ルールに従ってメールの配送先を決定する、という機能も持っているが、今のところはこの機能については利用していない。単に、fetchmailでダウンロードしたメールを、/var/spool/mail/user1に書き込むために利用しているだけだ。このため、書き込み先として“ -d user1 ”と明示的に指定し、振り分けを行わないようにしている。

最後に“ keep ”とあるのは、ISPのメールサーバからメールを削除しないように指定するためにつけてある。通常はこの指定は不要で、ダウンロードしたメールはISPのメールサーバから削除すべきだろう。今回は、実験用途ということと、外出先からISPに直接接続する場合も想定してメールを削除しない設定にしてみた。ただし、この状態で放っておくとISPのメールサーバのディスク容量を圧迫することになり、契約内容によっては利用可能な容量の上限に達してしまい、それ以上のメールの受信ができなくなることもあるので注意したい。

今回の設定のポイントは、fetchmailからprocmailへ直にメールを渡し、それをスプールに書き込む、というごく単純な処理に限定したことだ。これによって、sendmail(またはqmail)を動かす必要がなくなり、気にすべき作業はごくわずかになる。

この設定内容を~/.fetchmailrcに記述したら、忘れずにファイルパーミッションを「600」に変更しておこう。この指定により、ファイルの所有者(この例ではuser1)以外は、このファイルの内容を見ることはできなくなる。これは、ISPのメールアカウントとパスワードが平文で記述されているため、ぜひとも必要な処置である。fetchmail自体も、~/.fetchmailrcファイルのパーミッションを調べ、600でなければ処理を中止することになっているため、忘れずにパーミッションを変更しておきたい。

設定が完了したら、あとは単にfetchmailを起動するだけでよい。必要なオプションは~/.fetchmailrcに記述してあるので、コマンドラインでは特別なオプション等は必要ない。

IMAPサーバの準備

fetchmailを起動すると、ISPのメールサーバからメールがダウンロードされ、ローカルのスプール(ここでは、/var/spool/mail/user1)に次々と書き込まれていく。この内容は、コマン

ドラインのmailコマンドを使って簡単に確認することも可能だ。ただし、実際に試してみた際には、Subjectに日本語コードを直に入れてあるメールなどに遭遇すると表示が乱れたりしてどうも具合がよくない。やはりIMAPを使ってリモートからアクセスするほうが融通が効いてよいだろう。

IMAPも、Turbolinux Server 6.5では標準でインストールされるので、利用にあたってもたいした作業は必要ない。実は、デフォルトでは動作しないように止めてあるので、これを解除すればよいだけの話だ。

/etc/inetd.confファイルを見ると、IMAPもちゃんとあらかじめエントリが用意してあり、単にコメントアウトされているだけだとわかる。そこで、作業としては、

- **/etc/inetd.confファイルをエディタで開く**
- **IMAPのエントリ行を探し、行頭の“#”を削除してコメント解除する**
- **inetd.confファイルをセーブする**
- **inetdをリスタートする**

という手順で完了だ。

ただし、デフォルトの設定では、imapdはtcpd経由で起動されるように設定されている。この場合、ホストごとのアクセス制御がかかるが、デフォルト設定では、いっさいのリモートマ

Column

IMAPサーバにインターネットからアクセスする

ここで、ローカルのLAN内部に用意したIMAPサーバにインターネットからアクセスする場合に検討すべき要素について簡単に概観しておこう。

まず、今回利用したIMAPサーバでは、特別な設定は行っていない。この点で、IMAPサーバ自身は、アクセスがLAN内部からのものかインターネットからのものかを見分けることはしていないといえる。もっとも、これは多くのサーバソフトウェアで共通の仕様であり、だからこそtcpdを利用したアクセス制御などが用いられるわけだ。さらにセキュリティに関してちゃんと配慮しておきたいのであれば、ブロードバンドルータの設定や、専門のファイアウォールを用意するなどといった何らかの手段が必要になるだろう。

■IMAPサーバのユーザー認証

初めに、IMAPサーバ単体でのセキュリティがどの程度のレベルか、ということを考えてみよう。

IMAPでは、標準ではクリアテキストによる認証を利用する。つまり「ユーザーIDとパスワードが直接ネットワーク上を流れる」ということになる。IMAPでは、クリアテキストによるID／パスワードの送出以外にもより高度な認証メカニズムが利用できるようになっているが、利用するためにはサーバ／クライアントの双方で同じメカニズムをサポートしている必要がある。

今回の例では、特別な認証メカニズムは設定していないので、クリアテキスト認証を利用していることになる。つまり、メカニズムとしての安全性は、APOPなどをサポートしていない多くのISPのメールサーバと同等レベルで、経路によっては盗聴の危険があると考えるべきだ。ただ、LAN内にIMAPサーバを立てる場合、ユーザーIDは任意に設定可能である。ISPのメールアカウントのIDと（IMAPサーバへの接続時に使われる）ローカルのログインIDは完全に独立しているため、ユーザーIDとパスワードの組をいつでも任意に変更できる。このため、こまめな変更によってセキュリティを高める努力をすることは、ISPのメールサーバよりも多少容易に実行可能だといえる。

■IMAPサーバのセキュリティを強化する

IMAPサーバをインターネットからアクセス可能とし、かつセキュリティを高めるためには、どのようなやり方があるだろうか。

まず基本は、IMAP以外のアクセスを遮断することだろう。IMAPは標準でTCPポート143番を利用するので、これ以外のポートへのアクセスを禁止し、IMAP専用サーバとするのが第一歩である。ユーザーIDに関しても、任意に設定できるというメリットを活かし、IMAPアクセス専用のIDを用意し、最低限の権限しか付与しないようにしておくことも必要だと思われる。また、インターネットからアクセスする際に利用するマシンのドメイン名などがある程度限定できるのであれば、tcpdのアクセス制御を利用して接続可能なホストを極力少なくすることで安全性を高めることが可能かもしれない。

今回使用したTurbolinux Server 6.5の標準のimapdは、ワシントン大学が開発したimap-2000である。このサーバは、標準でCRAM-MD5という認証メカニズムをサポートしている。設定ファイル（/etc/cram-md5.pwd）にユーザー名とパスワードを記述しておく、という比較的簡単な手順でパスワードを暗号化できるが、メール本文は暗号化されない点に留意しておきたい。設定の詳細については、サポート用のWebサイト（http://www.washington.edu/imap/）が参考になる。

imap-2000は、SSLやKerberosといったより高度な認証メカニズムもサポートしている。これらの機能を利用するよう設定を変更してリコンパイルし、対応するIMAPクライアントを利用することでセキュリティを強化することも可能だと思われる。興味のある方は挑戦してみていただきたい。

さらに、ログイン時のID／パスワードやメールの内容そのもののの漏洩に備えるのであれば、セッション全体を暗号化するのが有効だ。たとえば、LAN内部に中継サーバを設置し、SSHを利用してインターネットからこの中継サーバにログインしたあと、中継サーバでIMAPクライアントを起動する、といった方法が考えられる。こうすれば、インターネット上を流れるパケットについてはすべてSSHで保護されることが期待できるので、安全性はかなり向上する（SSHについては、74ページからの「**SSHでセキュリティを固めろ！**」も参照のこと）。

シンからのアクセスを受け付けないようなキツい設定になっているので、これを多少緩和してやらないとまるでアクセスできない。

この設定には、/etc/hosts.allowファイルにアクセスを許可するホストのアドレスをリストアップするという方法で行うのが簡単だろう。とりあえず現時点ではインターネットからのアクセスは許可しないことにして、筆者の場合は利用しているプライベートアドレス全体に対してアクセス許可を出す（192.168.0.というエントリを加える）という方法を採った。

これで、LAN内のクライアントからはIMAPを使ってサーバに保存されたメールを参照できるようになった（「見るだけ」ということではなく、削除も可能である）。IMAPサーバに関しては特に設定は行っていないが、デフォルトでちゃんとスプール上の自分のメールファイルを開くし、フォルダを作成した場合には自分のホームディレクトリに作成されるので、何も問題ない。

LAN内のどのクライアントからも同じ構成のメールファイルにアクセスする、という初期の目標は達成できたわけだ。

拡張のための検討

さて、LAN内の複数のマシンでメールを一元管理する、という目標は達成されたが、設定の手間を惜しむため、制約事項もいくつかある。ここでは、この制約を解消し、より便利な環境に発展させていくことを考えてみよう。

まず、最大の制約要因となっているのは、sendmailを起動していないことである。筆者の発想としては、ローカルのsendmailにメールを投げてみたところで、実用上はそこからさらにISPのメールサーバに転送することになるだろうから、最初からメールクライアントで直接ISPのメールサーバに接続したところで大差ないというのが正直なところである。

しかし、どうせなら送信も「自分のサーバから行いたい」という希望もあるかもしれない。もちろん、ISPのメールサーバがダウンしているなどの状況でもメールの送信が可能なように、ローカルに立てたsendmailから受信者のメールサーバに直接接続してメールを送るような設定を行うことも可能である。ただし、独自ドメイン等を用意しないという前提に立つと、これはあくまでも送信のみの一方通行となるため、メリットがあるかどうかは考え方しだいというところだ。

また、sendmailを起動する場合、procmailを本来のメール振り分け用フィルタとして使いやすくなる、というメリットも考えられる。実は、現状でも簡単なルールを記述することで、メールをフォルダに振り分ける程度のことは実現可能である。

たとえば、ホームディレクトリにJUNKというフォルダを用意して、spam@spamsender.comからのメールはすべてここに放り込む（アドレスは架空のもの）といった設定は、ホームディレクトリに.procmailrcファイルを置き、フィルタの設定を行えば実現できる（**リスト2**を参照）。実用上はこれでも困らないと思うが、どうしてもやや力任せの場当たり的な印象が拭えないという気がする。

最も重要なのは、ここまでの設定では、ほとんどセキュリティに関して考慮していないという点だろう。設定を簡略化することを最優先にしているため、届いたメールは内容によらずともかくprocmailを起動して渡してしまう

▼リスト2　.procmailrcの設定例

```
LOGFILE=$HOME/.pmlog
MAILDIR=$HOME
VERBOSE

# from toshi-w@tt.rim.or.jp
:0 Hw
* ^.*[Ff]rom: spam@spamsender.com
JUNK
```

ごく単純な例だが、指定のメールアドレスspam@spamsender.comがFrom行に指定されていた場合は、そのメールをフォルダJUNKに置く、という動作をする。この設定をホームディレクトリに置いておけば、とりあえず期待どおりに動作するので、単純な振り分けならこれで対応可能だろう。

という辺り、不安を感じる向きもあるかもしれない（かといって、sendmailを起動すればセキュリティレベルが上がるというものでもないが）。

同様の懸念は、外部からのアクセスに関してもいえる。実のところ、これまでの解説では、外出時などにインターネット側からIMAPサーバにアクセスしてメールを読み出す、という使い方は想定していない。また、コラムにも書いたとおり、実はこれを禁止する方策も特に講じてはいない。関係があるのは、hosts.allowによるIMAPサーバへの接続制限くらいだ。逆に、ここを適当に変更すればインターネットから直接IMAPサーバに接続できるように設定することもできるし、ファイアウォールの設定でいったんアクセスを中継するような構成にする手もあるだろう。

ともあれ、LAN内にIMAPサーバを用意してISPのメールサーバからメールを取得して公開するだけでも、LANの使い勝手はずいぶん良くなるものと思う。しかも、これを実現するだけなら、実のところごく単純な設定変更だけで簡単にできる点もありがたいところだ。セキュリティやsendmailの適切な設定など、今回は踏み込まなかった問題も多いが、まずは簡単なところから始めてみるとよいのではないだろうか。

（渡邉利和）

SSHでセキュリティを固めろ!

常時接続のメリットはWebサーバを構築して情報を発信できるだけじゃない、外出先から「ちょっとあの書類を見たいな」なんていうときも、いつでも自分のマシンにアクセスできるようになる。ただ、いつでもアクセスできるというのは他人も同じ。

セキュリティを維持しながら安全にアクセスするにはどうすればよいのだろうか。その答えがSSHだ。

通信のセキュリティの確保

標準の状態では、telnetやFTPを利用している際にネットワーク上の通信は基本的には暗号化されていない。もちろん、パスワードは要求される。しかし、このパスワード自体は暗号化されない。さらに、通信中のデータも暗号化されていないため、通信経路の途中でパケットを捕まえれば、通信内容を盗み見ることもできなくはないのだ。

画面1　OpenSSHのWebサイト
日本語ページも用意されているので、ぜひ目を通しておこう。

そこで開発されたのが、パスワードを含むすべての通信を暗号化できるSSH（Secure Shell）プロトコルだ。SSHを使えば、安心してリモート操作、ファイルのコピーなどの作業を行うことができる。ただし、現在SSHはSSH社の製品となったため、ライセンス規定が厳しくなり、無償利用することが難しくなっている（SSHという名称自体もSSH社の登録商標）。

今回紹介するOpenSSHは、SSHプロトコルを利用したフリーのプログラム群で、現在の商用版SSHがフリーだった頃のバージョンをもとに、OpenBSDプロジェクトが作成したものだ。

■プロトコルバージョンの違い

SSHには、SSH1とSSH2の2種類のプロトコルが存在する。この2つに関しては基本的に互換性はない。そのため、どちらのプロトコルを利用するのかを意識して使う必要がある。

2つのバージョンの大きな違いは、公開鍵の交換方式にある。もともとSSH1プロトコルではRSA方式を使用していたが、RSAで使われている技術の特許問題をクリアするために、新たにDSA方式を採用したSSH2が開発されたという経緯がある。

なお、今回使用するOpenSSHは、SSH1とSSH2の双方を使用することができるが、プログラムによっては片方のプロトコルしか利用できないものもある。たとえば、後述のWindows用SSHクライアントTTSSHなどは、SSH1プロトコルのみをサポートしている。

SSHを使うための前準備

SSHを使うために必要な作業は、以下の2つだ。

①ローカルマシン上で個人用の秘密鍵と公開鍵を作る
②リモートマシン上の.sshというディレクトリに公開鍵のデータを保存する

なお、ここでは常時接続の対象となるSSHサーバを動作させるマシンをリモートマシン、実際にユーザーが操作を行う、SSHクライアントを動かすマシンをローカルマシンと呼ぶ。もちろん、現在Linuxマシンが手元に1台しかなくても、リモートマシンとローカルマシンに同じマシンを使って、接続テストを行うことも可能だ。

■インストール

今回は、テスト環境としてRed Hat 7.1を利用した。Red Hat 7.1では、OpenSSHのバージョン2.5.2p2が採用されているが、5月1日に最新バージョ

RPMパッケージ名	Server	Client
openssh-askpass-gnome-2.9p1-1	○	XとGNOMEを使用の場合
openssh-clients-2.9p1-1	○	
openssh-askpass-2.9p1-1	○	Xを使用の場合
openssh-2.9p1-1	○	○
openssh-server-2.9p1-1	○	

表1　SSHの動作に必要なファイル

ンであるOpenSSH 2.9がリリースされている。

セキュリティに関連する製品はできるだけ最新のものを使うべきだろう。特にオープンソースのものが、バグフィックスしたということであれば、なおさらだ。古いものを使っていて、既知のバグにつけ込まれたのではたまらない。最新バージョンをダウンロードして、これを使うことにしよう。

OpenSSHのWebサイト（http://www.openssh.com/）には、RPMパッケージを配布しているFTPサイトのリストが載っている。日本国内のサーバもあるので、適当なサーバからダウンロードしよう（インストールに必要なファイルについては**表1**を参照）。

Red Hat 7.1では、インストールタイプに「サーバ」を選択した場合、OpenSSHはすでにインストールされているので、それぞれのファイルを次のようにアップデートする。

```
# rpm -Uvh openssh-2.9p1-1.i386.rpm
```

■SSHサーバの起動

SSHを使って通信を行うには、リモートマシン上でサーバプログラムが動作していなければならない。RPMパッケージでインストールすると、WebサーバやFTPサーバなどと同様にSSHサーバプログラムを起動するスクリプトが/etc/rc.d/init.d/に作成される。SSHサーバの起動スクリプトはsshdだ。起動は次のコマンドで行う。

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

また、リブート時に起動し忘れてSSHが利用できない、といった事態を避けるために、ブート時に自動起動するよう設定しておこう。

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

■個人鍵の作成

SSHではユーザー認証に3つの鍵を使う。リモートマシンに保存されている個人用の「公開鍵」と、ローカルマシンに保存されている個人用の「秘密鍵」、そしてユーザーの頭の中にある「パスフレーズ」だ。

まず、SSH1プロトコルの個人用の鍵を作ろう。これにはローカルホストでssh-keygenコマンドを使う。ユーザーが覚えておくべきパスフレーズ（通常のログインパスワードと違いスペースを使うこともできる）を聞かれるので、確認を含めて2回入力する（**画面2**）。

```
$ ssh-keygen
```

ユーザーのホームディレクトリに.sshディレクトリが作成され、その中に2つのファイルができているはずだ。identityが秘密鍵、identity.pubが公開鍵となる。

次にSSH2プロトコル用の個人用鍵を作る。ssh-keygenに“-d”オプションをつけて実行すると、id_dsaという秘密鍵と、id_dsa.pubという公開鍵が生成される。

```
$ ssh-keygen -d
```

このとき、SSH1と同じくパスフレーズを設定する。

■公開鍵をリモートマシンに置く

続いて、リモートマシンのホームディレクトリに.sshというディレクトリを作り、SSH1の公開鍵のデータを「authorized_keys」、SSH2の公開鍵のデータを「authorized_keys2」という名前のファイルに保存する。公開鍵は文字どおり人に見られてもかまわないものなので、FTPやメール、フロッピーなど好きな手段で渡してかまわない。また、ホスト別に複数の公開鍵を保存するときは、authorized_keysなどの最後にcatコマンドなどを使って追加し

```
$ ssh-keygen
Generating public/private rsa1 key pair.
Enter file in which to save the key (/home/nor/.ssh/identity):
Created directory '/home/nor/.ssh'.
Enter passphrase (empty for no passphrase): (パスフレーズを入力)
Enter same passphrase again: (パスフレーズを再入力)
Your identification has been saved in /home/nor/.ssh/identity.
Your public key has been saved in /home/nor/.ssh/identity.pub.
The key fingerprint is:
d1:cf:b3:69:5b:62:ec:3b:cc:66:f0:23:11:50:f1:ae nor@local.ascii.co.jp

$
```

画面2　SSH1プロトコルの個人鍵の作成

```
$ ls -al .ssh
合計 12
drwxr-----    2 nor      nor          4096  5月 25 05:37 .
drwx---r-x    6 nor      nor          4096  5月 25 05:36 ..
-rw-------    1 nor      nor           339  5月 25 05:37
authorized_keys
```

画面3　パーミッションの確認

ていけばよい。

注意して欲しいのが、.sshディレクトリやauthorized_keysファイルのパーミッションだ。これらは自分以外は書き込みできないようになっている必要がある（**画面3**）。

```
$ mkdir .ssh
$ cp identity.pub .ssh/authorized_keys
$ cp id_dsa.pub .ssh/authorized_keys2
$ chmod 700 .ssh
$ chmod 600 .ssh/authorized_keys*
```

sshコマンドでログイン

手始めに、sshコマンドにSSH1プロトコルを意味する“ -1 ”オプションをつけて、リモートマシンにログインしてみよう（**画面4**）。

```
$ ssh -1 ホスト名
```

すると、まず“ The authenticity of host……(yes/no) ”という文字列が表示される。これは、「リモートマシンがローカルマシンに登録されていないが、接続してもいいか」と聞いてきているのだ。“ yes ”と答えればローカルマシンに登録され、次回からは、この手続きは必要なくなる。

次に“ Enter passphrase for RSA key…… ”のように、先ほど登録した個人用のパスフレーズを聞いてくるので、入力する。これでリモートマシンへのログインは完了だ。

もし、ここで単に“ nor@remote's password: ”のようにリモートマシンのパスワードを聞かれたら、いままでの作業がうまくいっていないということだ。手順に間違いがなかったか再確認してみよう。

また、何のプロンプトも表示されない場合は、リモートマシン（SSHのサーバ側）の/etc/hosts.allowと/etc/hosts.denyの設定をチェックすること。sshdに接続可能なホストとして、ローカルマシン（SSHのクライアント側）が登録されている必要がある。

SSH2プロトコルでログインする場合は、“ -2 ”オプションを付けて同様の手順を実行すればよい。オプションを付けない場合（つまりデフォルトの設定では）、SSH2プロトコルが使われる。

ファイルの転送

sshコマンドを使えば、リモートマシンにログインして作業することができるようになるので、telnetやrloginを使う必要はなくなる。

次なる課題のファイル転送を行ってくれるのが、scpコマンドだ。名前を見てわかるように、通常のcpコマンドに相当する機能を持っている。使い方もcpコマンドとほとんど同じだ。大きく違うのは、（当然のことだが）ホスト名を指定しなくてはいけない点だ。

ホスト名の指定はコロン（:）を使って行う。以下のコマンドは、パスフレーズを問い合わせ、ローカルマシン側にある“ foo ”というファイルを“ remote ”というリモートマシンの“ data ”ディレクトリにコピーする。

```
$ scp foo remote:~/data
```

リモートマシンからファイルを取ってきたい場合は、コピー元のファイルにホスト名を指定すればよい。

```
$ scp remote:~/data/bar .
```

ワイルドカードはもちろん、ディレクトリを再帰的にまるごとコピーするための“ -r ”オプションや、コピー元のタイムスタンプやパーミッションもそのままコピーする“ -p ”オプションなども使える。

scpコマンド以外にも、ファイル転送用のコマンドとしてsftpコマンドがある。こちらも名前のとおり、SSHプロトコルを使ったftpプログラムで、通常のftpと使い方は同じだ。リモートマシンの中身を見ながらファイル転送を行いたいときには、sftpを使うとよいだろう。

SSHの便利な機能

SSHを使いこなすうえで、知っておくと便利な機能を紹介しておこう。

■パスフレーズの記憶

1日に何回もログインするリモートマシンがある場合や、scpコマンドを使

```
[nor@local] $ ssh -1 levin
The authenticity of host 'remote (xxx.xxx.xxx.xxx)' can't be established.
RSA1 key fingerprint is 3a:92:a4:df:7b:c4:dc:c6:22:ff:e0:7e:15:bb:b9:d7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'remote,xxx.xxx.xx.xx' (RSA1) to the list of
known hosts.
Enter passphrase for RSA key 'nor@local.ascii.co.jp': (パスフレーズを入力)
Last login: Fri May 25 05:38:16 2001 from xxx.xxx.xxx.xxx
[nor@remote] $
```

画面4　sshコマンドによるログイン

ってファイルをコピーする際に、いちいちパスフレーズを入力するのは面倒だ。そんなときは、ssh-agentとssh-addが便利だ。この2つのコマンドで、秘密鍵とパスフレーズをシェルやウィンドウマネージャに記憶させ、sshやscpの実行時にパスフレーズ入力を省略することができる。

シェルの場合を例に説明しよう。まず以下のコマンドを実行する。

```
$ ssh-agent $SHELL
```

次に、実際に秘密鍵を記憶させる。

```
$ ssh-add
```

このとき、.ssh/identityというファイルについて聞かれたらSSH1のパスフレーズを、.ssh/id_dsaについて聞かれたらSSH2のパスフレーズを入力する。これ以降は、SSHを使うときにパスフレーズを入力しなくてもよい。

もちろん、記憶しているのはこのシェルの中のみだ。一度このシェルを抜けたあとは、またパスフレーズの入力が必要になる。

■ディレクトリの同期

Webサーバなどを運用している場合は、ローカルマシンとリモートマシンのディレクトリの内容を同期させたい、という要求は多いだろう。そんなときに使えるコマンドがrsync。このrsyncは、SSHを使った安全な通信を行うことが可能な同期ツールだ。

次のコマンドは、SSHを使って、

Column

SSHの認証の仕組み

SSHの認証方式の話に入る前に、まず公開鍵暗号方式とはどんなものかを説明しておこう。公開鍵と秘密鍵は、常にペアで使われるものだ。公開鍵は暗号化専用の鍵、秘密鍵は復号専用の鍵である。つまり、誰でも公開鍵を使ってデータを暗号化することはできるが、そのデータを復号できるのは秘密鍵を持っている人（もしくはプログラム）だけということになる。

SSHではこの暗号方式を利用し、大まかにいって3つの仕組みによって、通信のセキュリティを確保している。

■ホスト認証

ユーザーはほとんど意識することはないが、実はSSHではリモートマシンが本当に正しいマシン（ホスト）かどうかも認証している。これは、通信経路への細工などによるリモートマシンへの「なりすまし」を防ぐのに有効だ。

通常、SSHサーバのインストール時点で、リモートマシン上にはホスト用の公開鍵と秘密鍵が作成される。そして、ローカルマシンからリモートマシンへ初めてアクセスするときに、この公開鍵がローカルマシンに渡される。2回目の通信からは、この手順は省略される。

次にローカルマシンが公開鍵を使いランダムなデータを暗号化して、リモートマシンに送る。リモートマシンはそのデータを秘密鍵で復号し、ローカルマシンに送り返す。ローカルマシンは、送り返されたデータを元のデータと比較して同じものであれば、正しい秘密鍵を持っているマシン、すなわち正しいリモートマシンと判断するわけだ。

■ユーザー認証

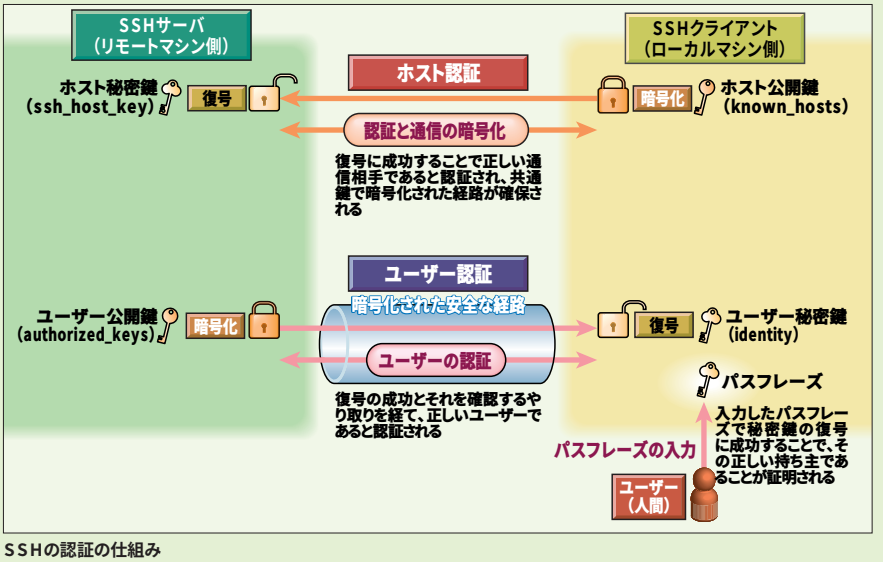
リモートマシンが正しいマシンであるとわかったら、今度はユーザーの認証を行う。

今度は、逆にリモートマシン側が公開鍵を使い、ローカルマシンの秘密鍵が正しいものであるかを調べる。ただし、ローカルマシンの秘密鍵が盗まれたり、ローカルマシン自体が別の人間に使われている可能性もある。そのため、秘密鍵自体もそのままでは使えないように暗号化されており、それを復号して使えるようにするのがパスフレーズなのである。

この方式では、ログインに必要となるパスワード（もしくはパスフレーズ）そのものがネットワーク上を流れることはない。また、パスフレーズが漏洩したとしても、同じローカルマシンを使わなければ、ログインできないのだ。

■通信内容の暗号化

ホストの認証と同時に、通信を暗号化するための共通鍵が作られ、それ以降の通信は共通鍵をもとに暗号化されることになる。ユーザー認証のフェーズもこの暗号化された通信経路で行われる。



SSHの認証の仕組み

“ www ”というリモートマシンの“ /var/www/html ”ディレクトリの内容を、ローカルマシンの“ localhtml ”ディレクトリと同じものにする。

```
rsync -e ssh -a --delete localthml
www:/var/www/html
```

■ポートフォワード

SSHには、sshやscpといった専用プログラム以外のツールでも、通信の内容を暗号化してリモートマシンの任意のポートに接続できるようにする「ポートフォワード」という機能がある。

よく使われるのが、popによるメールの取得だろう。POPプロトコルでは、APOPを使ってパスワードを暗号化することはできるが、メールの内容までは暗号化できない。そこで、ポートフォワード機能を使って、ローカルマシンとPOPサーバマシンのPOPポートとの間をSSHで接続するのだ。

次のコマンドラインを実行すると、ローカルマシンのTCPポート1110番にアクセスする（メールクライアントソフトで接続ポートをローカルマシンの1110番ポートに設定する）ことにより、リモートマシン“ mailserver ”のメールが読めるようになる。

```
$ ssh -2 -N -L 1110:mailserver:110
mailserver
```

	作者	URL
TeraTerm 2.3	寺西高	http://hp.vector.co.jp/authors/VA002416/
TTSSH 1.5.4	Robert O'Callahan	http://www.zip.com.au/~roca/ttssh.html

表2　Tera TermとTTSSHの入手先

Windowsとの通信

Windowsのローカルマシンから、Linuxのリモートマシンに接続したいというのも、よくある状況だろう。会社では泣く泣く（？）Windowsを使っているという人も多いはずだ。

■Tera Term ProとTTSSH

おそらく日本で一番使われているWindows用のターミナルソフトは「Tera Term Pro」だ。それにSSH1プロトコルの拡張を行うのが「TTSSH」。インストールは、まずTera Term Proをインストールし、そのディレクトリにTTSSHのzipアーカイブに含まれるファイルをコピーするだけだ。詳しい手順は省くが、それほど難しい手順ではないので、**表2**のWebサイトや付属のドキュメントを参考にしてインストールしてほしい。

インストールが完了したら、Tera Term Proに秘密鍵を用意しよう。といってもTera Term Proに鍵を作る機能はないので、Linuxマシン上で鍵を作り、秘密鍵をWindowsマシンにコピーし、公開鍵をLinuxマシン上のauthorized_keysに追加する。

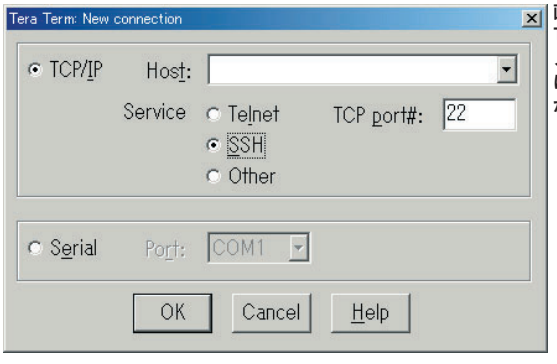
このとき、秘密鍵を異なるマシンに移すことになるので、作業時のセキュリティには十分配慮すること。また、Windows2000／NTでは問題ないが、Windows 9x系にはファイルのセキュリティという概念が存在しないという点にも注意したい。マシンに触れることができれば、誰でも秘密鍵を使うことができてしまうからだ。幸い、TTSSHでは秘密鍵の名前は保存されないので、できるだけ見つかりにくいような場所に保存しておくほうが無難だろう。

Linuxマシン上では、以下の操作を行えばよい。

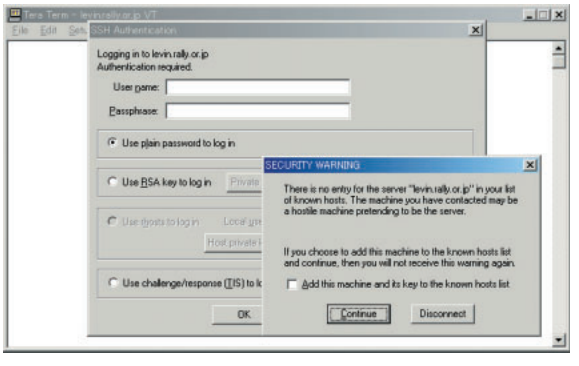
```
$ ssh-keygen -f keyfile
$ cat keyfile.pub >> ~/.ssh/autho
rized_keys
```

ssh-keygenの“ -f ”オプションは指定したファイル名で鍵ファイルを作るためのものだ。あとは、作成されたファイルをWindowsマシンの適当なディレクトリにコピーすればよい。

これで、準備は完了。ttssh.exeを起動すると、SSH接続オプションが加わったTrera Term Proの接続メニュー（**画面5**）が表示される。



画面5　TTSSH拡張したTera Term
ノーマルのTera Term Proにはなかった SSH Service が追加されている。



画面6　初回起動時の警告ダイアログ
初めて接続するホストでは警告ダイアログが表示される。

ホスト名を入力、［SSH］を選択して［OK］ボタンを押すと、いきなり**画面6**の警告ダイアログが現れる。これは、このマシンからのログインが初めてのため、ホスト認証データがないからだ。ここで［Add this machine and its key to the known hosts list］をチェックしてから［continue］しておけば、次回のログインからこの警告ダイアログは現れない。

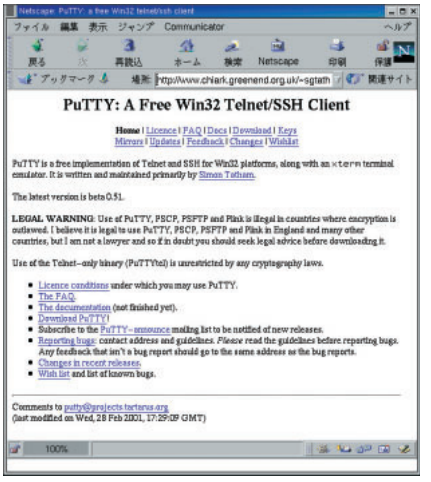
次の［SSH Authentication］ダイアログ（**画面7**）では、［Use RSA key to log in］を選び、［Private key file］ボタンをクリックして、先ほどの秘密鍵のファイルを選ぶ。あとは［User name］（Linuxのユーザー名）と［Passphrase］（パスフレーズ）を入力すれば、ログインできる。

Windowsからのファイルの転送

残念ながら、WindowsからのSSHを使ったファイル転送には、「これで決まり」といったベストチョイスのツールがない。

GUIを使えるものというと、WinSCPがその代表だが、日本語ファイル名の表示ができない（**画面8**）のと、SSHの目玉のひとつである公開鍵暗号方式に対応していないのが難点だ。もちろん、SSHによる通信経路の暗号化には対応しているので、公開鍵暗号は必要ないということであれば、WinSCPを使ってもよいだろう。

GUIでなくてよいのであれば、OpenSSHそのものをCygwinを使ってWindowsに移植したものもあるが、今回はMITライセンスによるフリーソフトの「PuTTY」を紹介しよう。



画面9　PuTTYのWebサイト

■PuTTYのセットアップ

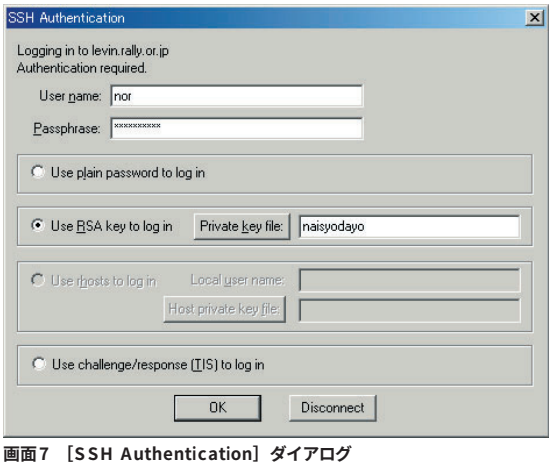
PuTTYはSSHプロトコルを実装したWindowsのフリーソフトだ。sshに相当するPuTTY、scpに相当するPSCP、ssh-keygenに相当するPuTTYgen、ssh-agent／ssh-addと同様の機能を持つPageantなどから構成される。

公開鍵（RSA）暗号方式によるファイル転送に必要なのは、PSCP（pscp.exe）、Pageant（pageant.exe）、PuTTYgen（puttygen.exe）の3つ。

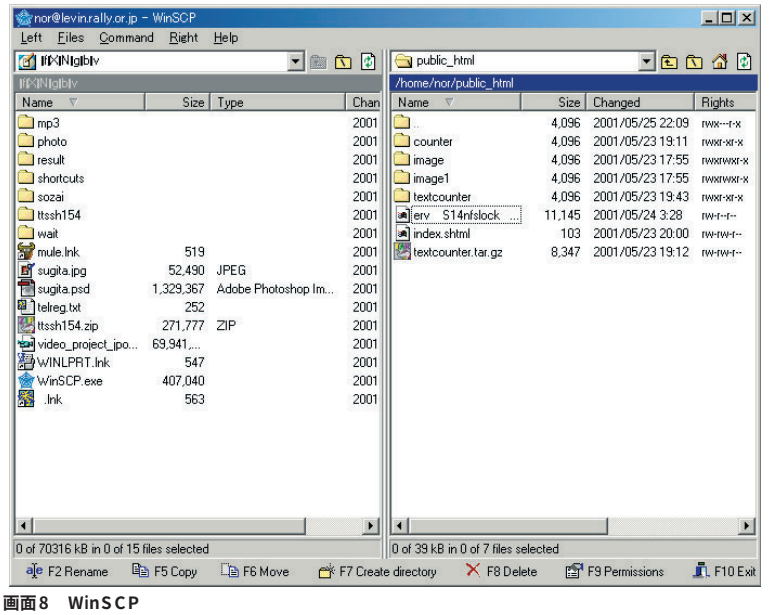
やはり、まずは個人鍵の作成から。TTSSHのときと同じように、Linux上で作ってもよいのだが、ここではWindows上でPuTTYgenを使ってみよう。puttygen.exeを起動すると**画面10**が表示される。

まず［Generate］をクリックする。そして［Key］とある枠の中でマウスを適当に動かそう。このマウスの動きで乱数を発生させているのだ。作成が完了すると、枠の中に、公開鍵が表示される。次に［Key Passphras］と［Confrirm Passphrase］にパスフレーズを入力して［Save］をクリックすれば、秘密鍵がファイルに保存される。

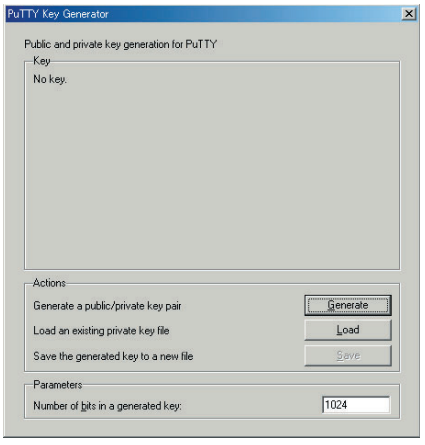
公開鍵は直接ファイルには保存されないのが、ちょっと面倒だが、いった



画面7　［SSH Authentication］ダイアログ
［Use plane password to log in］を選べば秘密鍵を用意しなくても、普通のUNIXパスワードを使ってログインできるが、もちろんお勧めしない。



画面8　WinSCP
ご覧のように日本語は文字化けしてしまい、まったく表示されない。



画面10　PuTTYgenのメインウィンドウ

んクリップボードに保存して、TTSSHでリモートホストにログインし、.ssh/authorized_keysに追加するといいだろう。

最後に認証の準備を行う。Pageant（pageant.exe）を起動すると、タスクトレイにPageantのアイコン（**画面11**）が入る。これを右クリックして［Addkey］を選び、先ほど保存した秘密鍵のファイルを指定する。そしてパスフレーズを入力すればOK。これで暗号認証の準備が整った。

■PSCPでファイルを転送する

PSCPはコマンドラインツールなので、パスの通ったディレクトリ（c:¥Windows、c:\Winntなど）にプログラムファイル（pscp.exe）を置いておくとよいだろう。

画面11　タスクトレイに追加されるPageantのアイコン
Pageant（右端のアイコン）はPuTTYツール群のすべてにおける公開鍵暗号認証をサポートする。

使い方はLinux上のscpコマンドと同じなのだが、Windows9xにはユーザーの概念がないため、どのユーザーでLinuxに接続するかを明示する必要がある。そのためホスト名の前に“ユーザー名@”を付ける（この方法は、Linux上のscpでもユーザー名が異なっている場合に有効だ）。

次の例では、リモートホスト“levin”にユーザー名“nor”でログインし、ホームディレクトリの“foo”というファイルをコピーしている。

```
> pscp nor@levin:foo .
```

実は、PSCPにはSecure iXPlorerというGUIのフロントエンドが存在する。しかし今回試した限りでは、ユーザー認証を行い、リモートホストの内部を見るところまではいくのだが、ファイルの転送がうまく動作しなかった。もっとも、テスト時間を十分に取れなかったので、環境の問題などもあるかもしれない。興味のある方は試してみてほしい。

トンネリングツール「Zebedee」

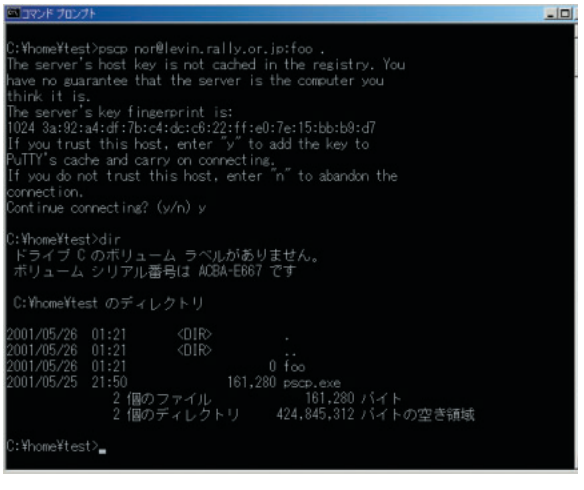
「SSHの便利な機能」のところで、ポートフォワード機能について簡単に説明した。POPによるメールの取得を例に挙げたが、もちろん使えるのはPOPだけでない。ポートフォワードの原理からいえば、SMTPやtelnet、FTPでさえも、セキュリティを確保しつつ、手慣れたソフトウェアで利用することは可能だ。

当然思いつくのが、「Windowsとのファイル転送も、使い慣れたFTPソフトとポートフォワード機能で実現できるのではないか」ということだ。

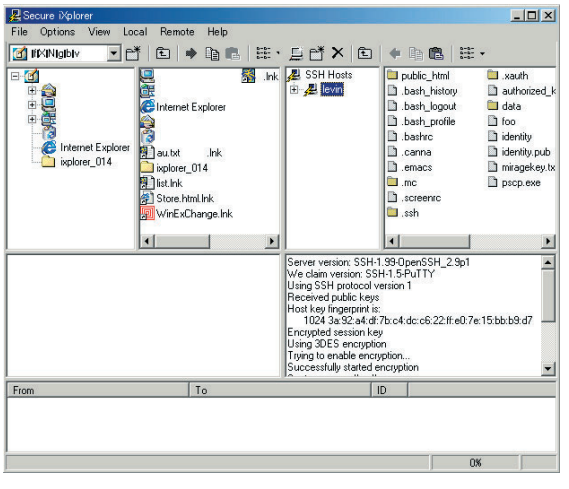
だが、「原理からいえば」と書いたように、実はポートフォワードを利用したFTPは、一筋縄ではいかないことのほうが多い。FTPクライアントの機能やサーバ側の設定に大きく依存するからだ。かといって、代わりの方法がないわけではない。何かをするのに、いくつもの方法から選べるのがLinuxのよいところだ。

■Zebedeeのセットアップ

Zebedeeは、2つのシステムの間に暗



画面12　PSCPを使ったファイル転送
はじめてリモートマシンに接続ログインするときはホスト認証の準備も行う。



画面13　PSCPのGUIフロントエンドSecure iXPlorer

号化された接続用の「トンネル」を作るツールだ。これはSSHのポートフォワード機能と呼び方が違うだけで、実質的にはほぼ同じものである。ただ、Zebedeeは、SSHにはないFTPのポートフォワードを上手に行うための機能を備えている。

ZebedeeのLinux版は現在、バージョン2.2.1になっているが、若干の不具合が発生しているようだ。Project Vineが提供しているVinePlusには、不具合の発生している部分に旧バージョンを利用することにより修正を施したRPMパッケージが登録されている。

今回は、このRPMパッケージを利用させてもらうことにしよう。ダウンロード用のFTPサイトについては、Project VineのWebサイトに掲載されているので参照してほしい。ここでは、代表的なRing ProjectのURLを紹介しておく。

```
ftp://ftp.ring.gr.jp/pub/linux/Vine
/VinePlus/2.1/RPMS/i386/zebedee-
2.2.1-0vl2.i386.rpm
```

Windows版に関してはZebedeeのWebサイトからダウンロードできるものが使用可能だ。インストールに関しては、通常の手順で問題なくできるだろう。

次にZebedeeの起動に移るが、その前にLinux上でFTPサーバがちゃんと動作しているかどうかを確認しておく必要がある。

Red Hat 7.1ではインストールしたままの状態では、FTPサーバは動作していないので、/etc/xinetd.d/wu-ftpdの“disable=yes”を“disable=no”にしておく。そのほかのディストリビューションの場合については、マニュアルなどのドキュメントを参考にして、FTPサーバが動作するよう適宜設定してほしい。

WinSCP	http://winscp.vse.cz/
PuTTY	http://www.chiark.greenend.org.uk/~sgtatham/putty/
Secure iXPlorer	http://www.i-tree.org/ixplorer.htm
Zebedee	http://www.winton.org.uk/zebedee/

表3　紹介したソフトウェアの関連URL

■Zebedeeの起動

まず、Linux上でZebedeeをサーバモードで起動する。

```
# zebedee -s
```

次に、Windows上でリモートマシン名を指定してクライアントモードで起動しよう。

```
> zebedee -d remote
      Listening on local port XXXX
      waiting for client connection
```

“XXXX”の部分にZebedeeが用意したトンネル用のポート番号が表示される。このあとローカルマシンのXXXX番ポートに接続するだけで、暗号化された状態で通信することができる。たとえば、telnetの場合は次のようにする。

```
> telnet localhost XXXX
```

■FTPの利用

ただ、FTPの場合はやはりちょっと面倒だ。ZebedeeにはFTP用のftpgw.tclというプロキシが付属しているので、Zebedeeサーバの起動前に、これを起動しておき、このプロキシを通して接続することになる（コマンドライン中の“remote”はリモートマシン名とする）。

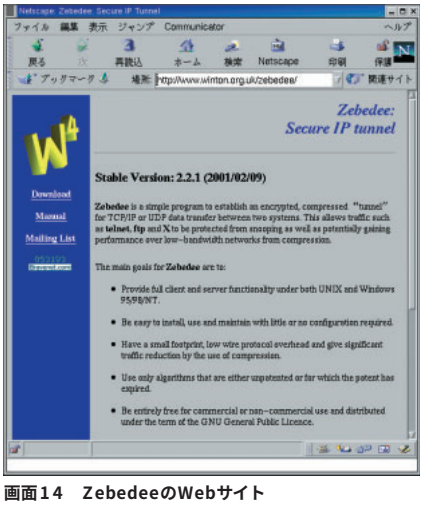
```
# tclsh /usr/bin/ftpgw.tcl -p
10000-10010 2121 remote 21&
# zebedee -s -r 2121,1000-10010
```

Windowsマシン側でも、次のようにZebedeeクライアントを起動する。

```
> zebedee 2121,10000-10010:remote:
2121,10000-10010
```

10000-10010はZebedeeが接続に使うポート、2121がクライアントが接続に使うポート、21は本来のFTPのポートだ。

これで、Windows上からFTPクライアントで暗号化された通信を行うことができるようになったはずだ。接続はローカル側の2121番ポートに対してPASVモードで行うことになる。ファイル転送するだけのために専用のサーバとクライアントを起動するのは、面倒には違いない。ただ、セキュリティと利便性は常に相反するものだ。バランスを考え、自分に合った選択をしてほしい。　（香取精二）



画面1-4　ZebedeeのWebサイト

IPマスカレードとセキュリティの設定

ダイヤルアップ接続サービスや個人向けの常時接続サービスのほとんどは、IPアドレスを1つしか割り当てない。このため、そのままでは複数のPCを同時にインターネットにつなぐことができない。これでは不便なので、ダイヤルアップISDNルータやブロードバンドルータは、プロバイダから割り当てられたIPアドレスを共有して、複数のPCで同時にインターネットを利用する機能を持っている。この機能をNAT（Network Address Translation）、あるいはIPマスカレードと呼ぶ。

LinuxにはIPマスカレード機能があるので、Linuxマシンをインターネットとの間にゲートウェイとして置くことによって、LANに接続した複数のPCで同時にインターネットを利用できるようになる。Linuxマシンにモデムやターミナルアダプタ（TA）を接続して使うこともちろん可能だ。

IPマスカレードの利点は、これだけではない。インターネットに直接接続されているのはLinuxマシンだけなので、ほかのPCはインターネットから隠蔽される。このため、セキュリティの強化を図りやすくなる。

IPマスカレードの設定

IPマスカレードを利用するためには、まずIP転送を有効にしなければならない。これは、Linuxマシンの複数のネットワークインターフェイス（イーサネットカードやシリアルポートなど）の間でパケットを転送する機能だ。IPマスカレードとは、パケットを転送する際に、IPアドレスやポート番号を変換する機能にほかならない。

■IP転送を有効にする

IP転送を有効にするには、ネットワークの設定ツール「Network Configurator」を使うのが簡単だ。rootユーザーになって、

```
# netcfg
```

を実行すると「Network Configurator」が起動する。「Routing」ボタンを押して表示される画面で、「Network Packet Forwarding（IPv4）」にチェックすればIP転送が有効になる（**画面1**）。

「Network Configurator」を使わないで設定するなら、/etc/sysctl.confの内容のうち、

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

と書かれた行の“0”を“1”に書き換えればよい。ところが、Red Hat系の古めのディストリビューションには、/etc/sysctl.confというファイルは存在しない。この場合は、/etc/sysconfig/networkの内容のうち、

```
FORWARD_IPV4=false
```

とある行の“false”を“true”に書き換える。これで、次回の起動時からIP転送が有効になる。

非Red Hat系のディストリビューションでは、

```
# cat 1 > /proc/sys/net/ipv4/ip_forward
```

を実行することでIP転送を有効にできる。/etc/rc.d/rc.localの最後にこのコマンドを書いておけば、起動時に自動的に実行される。

■IPマスカレードを有効にする

カーネルのバージョンが2.2の場合、ipchainsというコマンドを使ってIPマスカレードの設定をする。カーネルのバージョンが2.4の場合は、2.2と同様にipchainsも利用できるし、より高機能なiptablesというコマンドを利用することも可能だ。ただし、これらを同時に使うことはできない。

例として、内部LANで192.168.0.1～192.168.0.255というプライベートアドレスを使い、Linuxマシンのeth0というインターフェイスがLANに、eth1というインターフェイスがインターネットにつながっている環境を考えることにしよう。

ipchainsコマンドでIPマスカレードを有効にするには、rootユーザーになって、**リスト1**のコマンドを実行する。ただし、カーネル2.4を採用しているRed Hat Linux 7.xでは、

```
# modprobe ipchains
```

を実行して、ipchainsコマンドを利用するためのモジュールを組み込んでおく必要がある。

これで、IPマスカレードが有効になったのだが、利用するサービスによっ

ては、個別にモジュールを組み込む必要がある。たとえば、FTPやIRC、RealPlayerを利用するには、それぞれ、

```
# modprobe ip_masq_ftp
# modprobe ip_masq_irc
# modprobe ip_masq_raudio
```

を実行してモジュールを組み込む。モジュールは、/lib/modules/2.2.*/ipv4ディレクトリに置かれている。CuSeeMeなどのモジュールも用意されているので適宜利用してほしい。

カーネル2.4で、iptablesを使う場合は、**リスト2**のコマンドを実行する。

■クライアントの設定

クライアントがLinuxマシンのIPマスカレード機能を利用してインターネットへ接続するためには、クライアントマシンでデフォルトゲートウェイをLinuxマシンにしておく必要がある。それ以外に特別な設定は必要ない。

サーバのセキュリティ

さて、次はLinuxマシンのセキュリティ設定について解説しよう。公開サーバやIPマスカレードを動かすゲートウェイはインターネットに直接つなげるため、さまざまな攻撃を受ける可能性がある。したがって、セキュリティを確保するための作業は必須なのだ。

クラッキングの手口には、OSやサーバプログラムのバグを突くものが多い。この種の攻撃からサーバを守るためには、システムを適切に設定するのはもちろん、常にセキュリティ情報を集めて、セキュリティに関わるバグが修正された最新版のソフトウェアをインストールする必要がある。

では、システムの設定はどうすればよいのだろうか。今回は、設定の簡便さに重点を置き、いくつかのポイントに絞って解説することにしよう。より細かく設定したい方は、Linux JF（Japanese FAQ） Projectの Webページ（http://www.linux.or.jp/JF/）などで関連文書を入手し、研究してほしい。

■不要なサービスを止める

サーバで動かすサービスは必要最小限のものに絞ろう。不要なサービスを止めてしまえば、そのプログラムに弱点があったとしても、そこからクラッカーの侵入を許すことはなくなる。

各種サーバプログラムなどのサービスを自動的に起動するかどうかは、ntsysvコマンドを使うと簡単に設定できる。rootユーザーになって、

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

を実行してntsysvコマンドを起動しよう。--levelオプションは、設定を反映させるランレベルの指定だ。Red Hat系のディストリビューションやTurbolinuxでは、ランレベル3がマルチユーザーモードで、5はX Window System付きのマルチユーザーモードだ。通常、ランレベル4は未使用だが、ランレベル3と同じように扱われることが多い。

ntsysvを起動したら、カーソルキーの上下でサービスを選び、スペースキーで自動的に起動するかどうかを設定する（**画面2**）。カギ括弧内に*があるサービスはシステムの起動とともに動作を開始する。

■パケットフィルタを使う

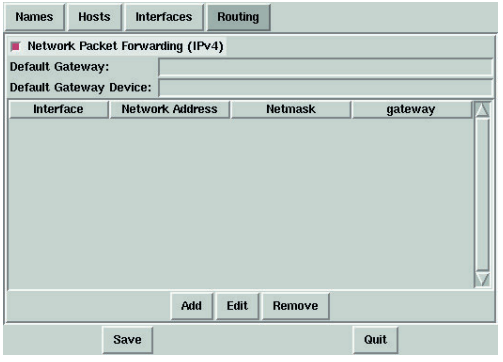
IPマスカレードの設定でも用いたipchainsやiptablesコマンドを使うと、パケットの種類や発信元／送信先のIPアドレスやポート番号をもとにフィルタリングできる。適切なフィルタリングルールを設定するには、コマンドの書式だけでなく、TCP/IPネットワー

▼リスト1　ipchainsでIPマスカレードを有効にする

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

▼リスト2　iptablesでIPマスカレードを有効にする

```
iptables -t nat -A POSTROUTING -o eth1 -s 192.168.0.0/24 -j MASQUERADE
```



画面1　Network ConfiguratorでIP転送を有効にする

画面2　ntsysvコマンドで不要なサービスを止める

クの知識も不可欠なのだが、これらすべてを解説するのは本稿の範囲を越えてしまう。そこで、今回は、Red Hat Linux 7.1に付属するフィルタリングルールの設定ツール「lokkit」を使うことにしよう。「lokkit」は、ipchainsに対応するツールで、簡単なメニューから条件を選ぶと、それに合ったフィルタリングルールが書かれたファイルを出力する。のちほど、ルールファイルの内容も説明するので、ほかのディストリビューションを利用している方でも参考になるはずだ。

「lokkit」を起動するには、rootユーザーになって、次のようにコマンドを実行する。

```
# lokkit
```

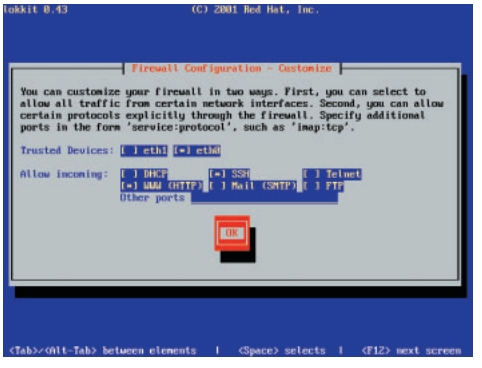
すると、**画面3**のような画面になるので、セキュリティレベルを、「High」、「Medium」、「No firewall」のなかから選ぶ。「High」を選ぶと、TCPの接続要求と、DNSによる名前解決を除くUDPのパケットをすべて排除する。すなわち、このLinuxマシンへの接続は一切できないし、DNSとのやりとり以外はUDPによる通信もすべてできなくなる。非常に安全だが、これではサーバにならない。

「No firewall」を選ぶと、フィルタリングは行われない。これは危険だ。というわけで、お勧めは「Medium」を選び、さらにカスタマイズするという方法になる。

「Medium」を選ぶと、1023番以下のポートへの接続、NFSでのファイル共有、Xサーバへの接続、Xフォントサーバへの接続を禁止する。1023番以下のポートは、root権限がないと利用できないポートで、ほとんどのサーバプログラムが利用する。したがって、このままでは、Webをはじめとするサーバを公開できないことになる。そこで、「Medium」を選んだあと、「Customize」ボタンを押して追加設定をしよう（**画面4**）。

「Trusted Devices：」では、安全な内部LANにつながったネットワークインターフェイスを選ぶ。ここで設定したインターフェイスから出入りするパケットはノーチェックで通すようになるので、インターネットにつながっているインターフェイスを選ばないように気を付けたい。

画面3　lokkitの起動画面



画面4　「Customize」画面で追加設定を行う

▼リスト3　/etc/sysconfig/ipchainsの例（行番号は説明のために付けたもの）

```
 1: :input                                                  入ってくるパケットのデフォルトは許可
 2: :forward ACCEPT                                         転送するパケットのデフォルトは許可
 3: :output ACCEPT                                          出ていくパケットのデフォルトは許可
 4: -A input -s 0/0 -d 0/0 22 -p tcp -y -j ACCEPT           SSHサーバへの接続を許可
 5: -A input -s 0/0 -d 0/0 80 -p tcp -y -j ACCEPT           Webサーバへの接続を許可
 6: -A input -s 0/0 -d 0/0 -i lo -j ACCEPT                  ループバックデバイスに入ってくるパケットを許可
 7: -A input -s 0/0 -d 0/0 -i eth0 -j ACCEPTLAN             LAN側のインターフェイスから入っているパケットを許可
 8: -A input -p tcp -s 0/0 -d 0/0 0:1023 -y -j REJECT       1023番以下のポートへのTCP接続要求パケットを拒否
 9: -A input -p tcp -s 0/0 -d 0/0 2049 -y -j REJECT         NFSサーバへのTCP接続要求パケットを拒否
10: -A input -p udp -s 0/0 -d 0/0 0:1023 -j REJECT          1023番以下のポートへのUDPパケットを拒否
11: -A input -p udp -s 0/0 -d 0/0 2049 -j REJECT            NFSサーバへのUDPパケットを拒否
12: -A input -p tcp -s 0/0 -d 0/0 6000:6009 -y -j REJECT    Xサーバへの TCP接続要求を拒否
13: -A input -p tcp -s 0/0 -d 0/0 7100 -y -j REJECT         XFS（Xフォントサーバ）へのTCP接続要求を拒否
```

「Allow incoming：」では、公開したいサーバを選ぶ。先ほどの画面の例では、インターネットからWebサーバとSSHサーバに接続できるようにしている。また、CATVインターネット接続などで、DHCPによるネットワーク設定情報（IPアドレスの割り当てなど）を受ける場合は、「DHCP」にもチェックを入れる必要がある。

「Other ports」には、「サービス名：使用プロトコル」という書式で公開するサーバを追加できる。たとえば、IMAPサーバを公開するなら、「imap：tcp」と書く。サービス名と使用するプロトコルは、/etc/servicesに書かれているので参考にしよう。

■設定ファイルを読んでみる

さて、こうして設定したフィルタルールは、/etc/sysconfig/ipchainsというファイルに書き込まれる。ntsysvコマンドで「ipchains」というサービスを有効にしておくと、システムの起動時に、この内容を読み込んでフィルタルールが有効になる。

では、/etc/sysconfig/ipchainsの内容を見てみよう。先ほどの例では、**リスト3**のようなファイルが作られた。1～3行目は、デフォルトのポリシーで、入ってくるパケット、転送するパケット、出ていくパケットとも許可する設定だ。4行目からが個別のルール設定となる。パケットの評価はこの行から順に行われていき、条件に合致した行があればそのルールにしたがって処理される。そのため、ルール設定の順番は重要だ。最後まで合致する条件がなかったパケットはデフォルトのポリシーで処理される。

4行目を例に簡単に説明しよう。入ってくるパケット（-A input）で、発信元IPアドレスとポート番号は任意（-s 0/0）で、送信先IPアドレスが任意でポート番号が22（-d 0/0 22）、プロトコルはTCP（-p tcp）の接続要求パケット（-y）を受け入れる（-j ACCEPT）という設定だ。プロトコルがTCPでポート番号22というのはSSHサーバが使うことになっているので、これはSSHサーバへの接続要求ということになる。接続要求パケットは最初の1回しか来ないので、接続が確立したあとのパケットはこの行に合致しない。しかし、残りの行のどれとも合致しないので、デフォルトポリシーにしたがって許可されることになる。

6行目と7行目では、ローカルループバックインターフェイス（lo）と、内部のLANに接続しているインターフェイス（この例ではeth0）からのパケットは無条件に許可している。ローカルループバックはさまざまなプログラムが利用しているので必ず開けておこう。

8行目では1023番以下のポートへのTCP接続要求を拒否している。1023番以下のポートは特権ポートといい、root権限がないと利用できない。多くのサーバプログラムが接続を待つのに使われるポートだ。TCPの接続要求を拒否しているので接続は確立しない。ただし、22番（SSH）と80番（Web）はこれより前の行で許可しているので接続できるのだ。同様に、10行目では1023番以下のポート宛のUDPパケットを拒否している。UDPには接続を確立するという概念がないので、接続要求パケットに合致する-yオプションは付けない。

このほかに、1024番以上のポートを利用するサービスで代表的な、NFS（Network File System）、X Window System、Xフォントサーバへの外部からの接続要求パケットも拒否している（6～7行目があるのでローカルループバックと内部LANからは接続できる）。

■Red Hat Linux 7以外では

Vine Linuxのように、設定ツールのlokkitコマンドが付属しないディストリビューションでも、ntsysvコマンドで「ipchains」サービスが表示されるなら、/etc/sysconfig/ipchainsに**リスト3**のようなルールを書いて利用することができるので試してほしい。

また、実際に実行されるコマンドは**リスト4**のようになるので、「ipchains」サービスが利用できないディストリビューションでは、/etc/rc.d/rc.localファイルにこれを書き込んで利用することが可能だ。

（編集部）

▼リスト4　実行されるipchainsコマンド

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

3 ダイナミックDNSでお手軽マイドメイン

フレッツ・ISDNにはじまり、CATV、ADSLとさまざまな選択肢がここ1、2年で急速に現実のものとなった、インターネット常時接続の世界。おかげでつなぎっぱなしのミラクルワールドを楽しんでいる人の数は、飛躍的に増大した。もちろん、Linuxユーザーのあいだでもそれはしかり。本誌読者の中でも多くの方がLinuxならではの常時接続環境を楽しんでおられるのではないだろうか。

「Linuxならでは」の常時接続といえば、やはり「サーバの公開」。自分の思い通りになるWebサーバを、覚えやすいドメイン名で世界中に公開するのだ。CGIスクリプトを自由に設置できないホスティングサービスとはおさらば。長いURLも要らない「マイドメインでマイサーバ」を公開する。これぞLinuxで常時接続の醍醐味だろう。

常時接続環境は手に入れたけど

とはいえ、多くの環境はこれを実現するにはちょっと力不足だ。マイドメイン・マイサーバを実現するには、

①グローバルIPアドレスを持っている
世界中からあなたの手元のマシンにアクセスしてもらうには、当然、そのマシンに全世界で1つしかないグローバルIPアドレスが必要だ。

②固定IPアドレスを持っている
多くの常時接続サービスでは、接続のたびにIPアドレスが変わる。これでは任意のドメイン名の割り当てはムリだ。

③DNSサーバを持っている
あなたのドメインを管理するために、自由に管理できるDNSサーバが必要だ。

④できれば複数IPアドレスがほしい
公開するサーバ用、DNSサーバ用……。

といった要件を満たす必要がある。冒頭に挙げたようなサービスはこういった条件に合わないか、あるいはそのぶん料金が高くなるかのいずれか。

だが、ビジネスなど本格的な運用目的でなければ、前述の項目を完全に満たさなくても自分だけのサーバを公開できる理想的な（？）方法がある。それが「ダイナミックDNS」だ。

ダイナミックDNSとは？

ダイナミックDNSとは、非固定のIPアドレスしか持たないホストを、特定のドメイン名（FQDN）で常に参照できるようにするサービスの俗称だ。

基本的にはDNSサーバの代行サービスみたいなもので、ドメイン名とサーバのIPアドレスをこのサービスのデータベースに登録しておけば、インターネットのドメイン名問い合わせに応じて「このIPアドレスのマシンにアクセスしてください」と回答してくれる。ちょっと違うのは、IPアドレスがコロコロ変わることを前提にしたサービスだという点である。

ダイヤルアップ環境などで、つなぐたびにIPアドレスが変わってしまうマシンがあったとする。ふつうなら、こんなマシンにはドメイン名を使用してのアクセスはムリだ。だが、ダイナミックDNSでは、接続ごとに「新しくXXX.XXX.XXX.XXXというアドレスで接続しなおしたよ」という「通知」を受け付け、データベースを更新する。そして、ドメイン名について問い合わせがくれば、この新しいIPアドレスを参照するようにと返答してくれる（**図1**）。おかげで、非固定IPアドレスのマシンでも、一意なドメイン名でアクセスが可能になるのだ。

安価な常時接続環境の多くは、従来のダイヤルアップ環境同様、接続ごとに違うIPアドレスを交付する。これは先の4つの条件の②に挙げたように、独自ドメインを実現する際には大きなネックとなる。だが、ダイナミックDNSを利用すればこの問題は解決する。また、DNSサーバとしての役割もダイナミックDNSサービスが果たしてくれるので、③も解決する。複数のサーバを立てる必要もないので、④も解決だ。つまり、ダイナミックDNSを利

用すれば、実に手軽に公開サーバを実現できるというわけだ。

もちろん、問題もいくつかある。

- **グローバルIPアドレスが必須。CATVサービスなどに多いプライベートIPアドレス環境では、残念ながらあきらめるしかない**
- **DNS情報はキャッシュされるものなので、IPアドレスが変わったドメインには、しばらく（短く見て数十分）アクセスできなくなることがある**

こういったポイントを理解したうえで、ダイナミックDNSを活用していこう。

環境構築のポイント

ここでは、ダイナミックDNSで公開サーバを実際に作ってみる。とはいえ、常時接続環境の違いはもちろん、マシンの規模もさまざまだろうから、まずはその部分から。

・常時接続環境は？

とりあえずありきたりなところでフレッツ・ISDNとする。

・マシンは？

デスクトップPC。Pentium III 866MHz、メモリ256Mバイトの自作マシンで実験。常用マシンに、バックグラウンドで公開サーバを走らせるという設定だ。

・通信機器は？

TAをシリアルポート経由でマシンに直付け。アイワTM-AD1282を使用。

・OSは？

Red Hat Linux 7J。もちろんほかのディストリビューションでもOK。

このマシンを、契約済みのフレッツ・ISDNとプロバイダを利用してダイヤルアップインターネット接続できるように設定する。ダイヤルアップ環境の構築は最近のディストリビューションなら不要な気もするが、前ページの解説でもわかるように、接続を確立するたびごとにダイナミックDNSサービスに自分のIPアドレスを登録する「仕掛け」を用意しなければならない。使用するPPPプログラムによって必要な作業は異なるが、ここでは「PPxP」を前提に解説していくこととする。

PPxPの導入

PPxP（http://www.dsl.gr.jp/~manabe/PPxP/）は、多くのディストリビューションで利用されているPPPプログラムだ。Vine LinuxやKondaraでは標準で利用できる。簡単にインストール方法をまとめておこう。

PPxPは「userlink」モジュールと本体のPPxPから成っている。まずはPPxPのホームページでuserlinkのtarボールをダウンロードする（執筆時点では0.99cが最新）。適当なディレクトリにtarボールを置いて**画面1**の手順でインストールする。一部の環境ではヘッダファイルの参照でエラーが出るので、その場合はMakefile中のヘッダファイルのパスを修正するようにしてほしい。

```
$ tar xvfz userlink-0.99c.tar.gz
$ cd userlink-0.99c/
$ ./configure
$ make
# su
# make install
# depmod -a
# modprobe userlink
# lsmod
```

画面1　userlinkのインストール

```
$ tar xvfz ppxp-0.99120923.tar.gz
$ cd ppxp
$ ./configure --sysconfdir=/etc
$ make
# su
# make install
```

画面2　PPxPのインストール

続いて、PPxP本体（同じく0.99120923が最新）のインストールを行う。ダウンロード後、**画面2**の手順を実行する。設定ファイルは/etc/ppxp以下に置かれる。

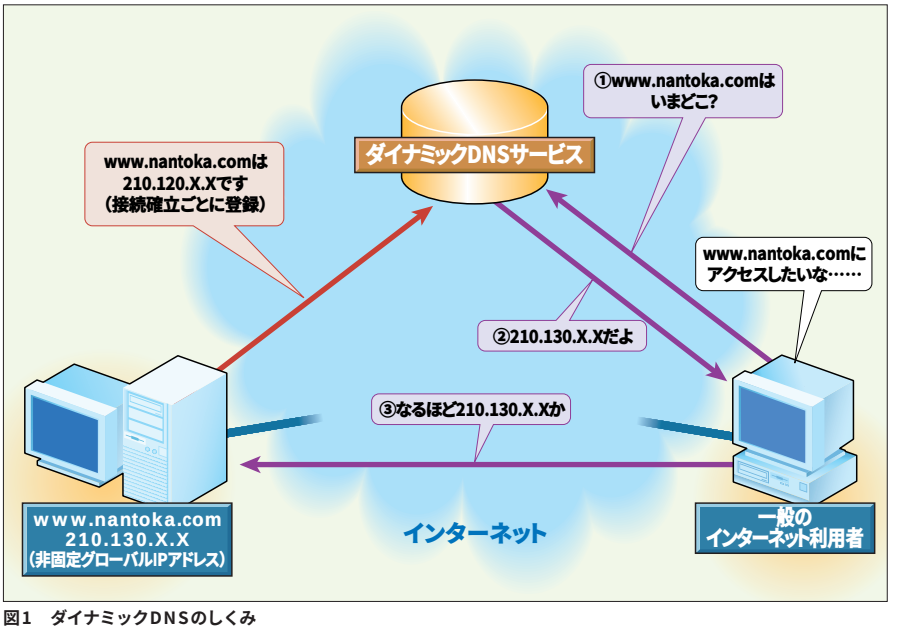


図1　ダイナミックDNSのしくみ
接続ごとに変わる非固定グローバルIPアドレス環境のマシンでも、ダイナミックDNSサービスを使えばインターネットのほかのユーザにサーバとして公開できる。

その後、たとえばTAをCOM1につなげているなら、デバイスを表す「/dev/ttyS0」に利用者への許可を与える必要がある。

```
$ su
# /usr/sbin/vigr（viで/etc/groupを編集）
```

として、uucpグループに常用しているアカウント名を追加する。具体的には、次のように“uucp:”で始まる行の末尾に、カンマに続けてアカウント名を記述すればよい。

```
uucp:x:14:uucp,h_siobar
```

パーミッションの変更は下記のコマンドで行う。

```
# chmod 660 /dev/ttyS0
# chgrp uucp /dev/ttyS0
```

また、ロックファイルが作成される/var/lockにも、uucpグループへの書き込み許可が必要だ。最終的に**画面3**に示すようなパーミッションになっていればよい。作業を終えたら、ログインしなおすことを忘れずに。

最後に、PPxPでフレッツ・ISDN経由によるダイヤルアップ接続ができるよう設定スクリプトを作る。uucpに登録した常用アカウントから、

```
$ ls -l /dev/ttyS0
crw-rw----    1 root     uucp       4,  64 Sep 13  2000 /dev/ttyS0
$ ls -ld /var/lock
drwxrwxr-x    6 root     uucp         4096 May 10 17:52 /var/lock
```

画面3　パーミションの状況

```
$ ppxp
```

として起動し、

```
PPxP version 0.99120923
interface: ul0
ppxp> qdial
```

としてqdialモードに入ると、エントリの作成画面が表示される。**画面4**と［More］でジャンプする**画面5**を参考に、自分の環境用の設定を行おう。終わったら**画面4**の［Save］で保存する。ファイル名は任意だが、“iij4u-isdn”といったわかりやすい名前にしておくといい。

実際に接続できるかどうかは、

```
$ ppxp iij4u-isdn -C connect
```

切断は、

```
$ ppxp iij4u-isdn -C disconnect
```

で確認できる。

Dyndnsで ダイナミックDNSを実践

ダイナミックDNSサービスの中でも有名なのが「Dyndns.org」（http://www.dyndns.org/）だ。ここでは、このDyndnsを例に実際のサーバ公開の手順を見ていこう。

Dyndnsは海外のサービスだ。表記はすべて英語だが、国内でも多くの人に利用されている。その理由は、ダイナミックDNSの基本機能がブラウザ上からカンタンに操作できるよう、きちんと整理して提供されている点、比較的安定している点、そしてなにより無料で利用できるという点だ。現在では多数の利用者がDyndnsを介してドメイン運営を行っており、かなりのコストがかかっているはずだ。もしサービスが気に入って、長期利用することになったら、サイトの案内を読んで運営資金にいくばくかの寄付をするとよいだろう。

ユーザー登録と初期設定を

Dyndnsを利用するには、まずユーザー登録が必要だ。トップページ（**画面6**）から［Members NIC］－［New Account］にアクセス。利用条件が説明されるので、よく読んで［Agree］をクリックする。これでユーザーアカ

画面4　qdialの設定画面

画面4　qdialの設定画面

ウント作成画面に移る（**画面7**）。ここで入力するのは、次の2項目。

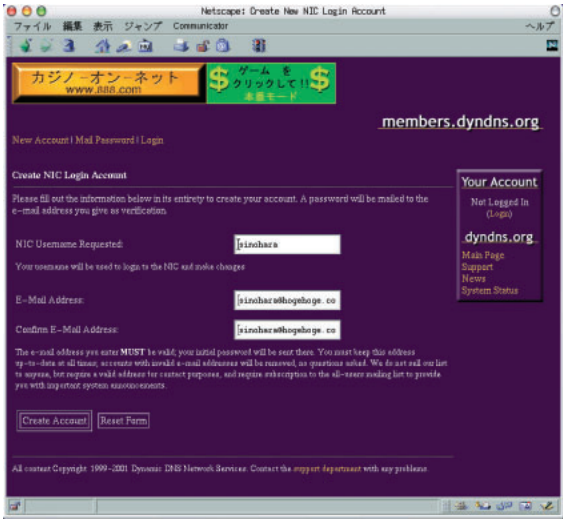
・NIC Username
アカウント名。利用するドメイン名とは関係ないので、適当なものでよい。
・E-Mail Address
通知先メールアドレス。2回入力する。ここに初期パスワードがメールで送られてくるので間違えないこと。

入力を終えたら［Create Account］を押す。すぐに登録通知メールが届くので、さきほどのアカウント名と初期パスワードを使い、Dyndnsのメンバーページ（http://members.dyndns.org/nic/login）にログインする。パスワード変更画面（**画面8**）になるので、新たにパスワードを設定しなおそう。このパスワード変更作業が終われば、ユーザー登録は完了だ。

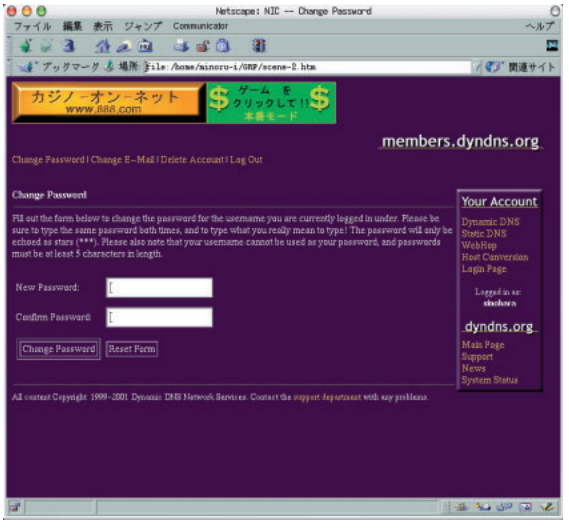
続いて、ドメイン名とIPアドレスの登録作業だ。当初の操作はメンバーページから行う。［Your Account］の［Dynamic DNS］をクリック。この画面では、現在登録済みのドメインの一覧を参照できる（**画面9**）。新規ドメインを作成するには、右側にある［Create New Host］へ。［New Dynamic DNS Host］の画面に移る（**画面10**）。この画面の各入力項目には、それぞれ下記のような意味がある。

・Hostname
新規に作成するドメインの名前。Dyndnsで無料で取得できるのは「dyndns.org」、「homeip.net」、「ath.cx」など9つのドメインのサブドメインだ。好きなものを選んで「hogehoge.dyndns.org」などとする。
・IP Address
そのドメイン名で参照させたいマシンの現在のグローバルIPアドレスを入力。該当マシンからこのページにアクセスしていれば、自動的にそのアドレスが入る。

画面6　Dyndns.org



画面7　ユーザアカウント作成画面



画面8　パスワード変更画面

画面9　登録済みエントリの一覧画面

・Enable Wildcard
「www.hogehoge.dyndns.org」というように「hogehoge.dyndns.org」の前に何かホスト名を追加しても、同じマシンを参照させるかどうか。チェックすると参照させることになる。
・Mail Exchanger
このドメイン宛のメールを処理するメールサーバを指定する。空白の場合は「IP Address」のマシンがメールサーバとして扱われる。とりあえず空白にしておく。
・Backup MX?
IP Addressの項目で指定したマシンがダウンしているときのみ、「Mail Exchanger」に指定したマシンをメールサーバとして扱うか否か。とりあえずチェックなし。

［Add Host］を押すと、ドメイン名が登録されて利用できるようになる。手近なLinuxマシンのから、

```
$ nslookup hogehoge.dyndns.org
  Name: sinohara.dyndns.org
  Address: 210.130.000.000
```

などと名前解決できるようであれば登録成功だ。

現在のIPアドレスを反映するには

とりあえず登録が終わり、hogehoge.dyndns.orgというドメイン名を使ってあなたのマシンにアクセスすることが可能になった。しかし、そのマシンでダイヤルアップの切断・再接続が起これば、IPアドレスが変わってしまうためアクセス不能となる。そこで、接続するたびに最新のIPアドレスをDyndnsに登録する必要がある。

これを手作業で行う場合には、接続後にDyndnsのメンバーページにログインし、**画面9**のホスト名の一覧から該当ドメインを選択して情報の更新を行う必要がある。更新ページ（**画面11**）の各項目の意味は次のとおり。

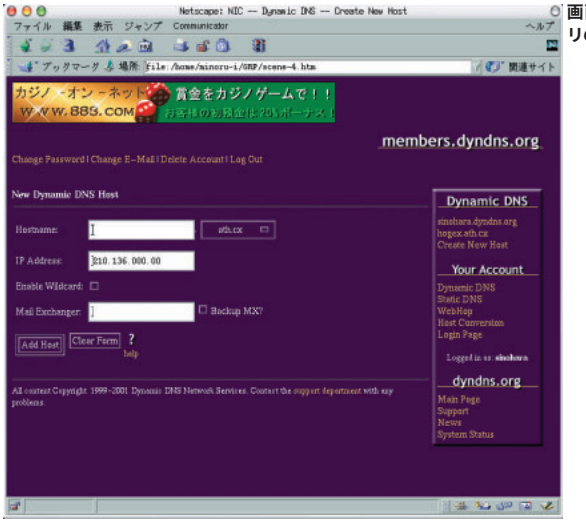
・IP in Database
該当ドメインの、現在Dyndnsデータベースに登録されているIPアドレス。
・IP Address
新しく登録しなおしたいIPアドレス。自動でアクセスに使用しているマシンのIPアドレスが入る。

そのほかは新規登録時と同じだ。IPアドレスがきちんと入力されているのを確認したら［Modify Host］を押す。これでDyndnsデータベースでの登録情報は即座に更新される。なお、インターネット全体にこの更新が反映されるまでには時間がかかるので、すぐにnslookupによる確認を行っても元のIPアドレスが返ってくる可能性があることに注意。

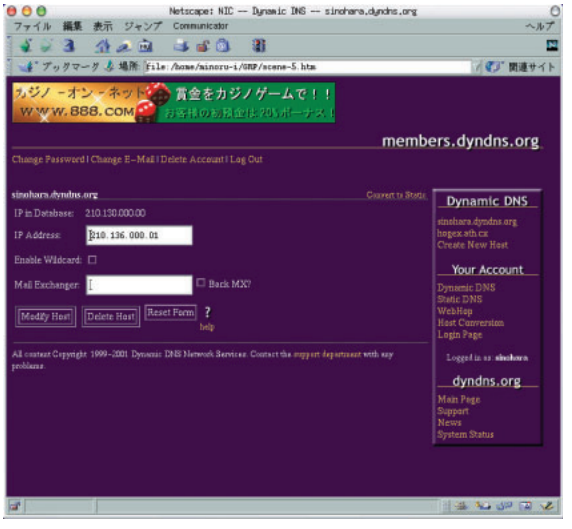
PPxP＋ddupで自動更新しよう

もちろん、こんなことを毎回手作業で行っていては安定して公開サーバを運営するのは不可能だ。DyndnsへのIPアドレス登録を、スクリプトによる自動更新で行う方法を考えてみよう。

この自動化には、Dyndns向けに公開されている自動更新ツール「ddup」が必要だ（http://www.ddup.org/）。ddupは、コマンドラインからドメイン名とIPアドレスを与えるだけで、Dyndnsにアクセスしてドメイン情報を修正してくれるモノ。Windows、OS/2向けなども公開されているが、UNIX用ddup（執筆時点では3.0.1が最新版）をダウンロードしてきてコンパイルする。対話式のインストールスクリプトが付属しているので、**画面12**に従って作業してみよう。なお、PPxP



画面10　新規エントリの作成画面



画面11　ダイナミックDNSエントリの編集画面

の設定を行ったときと同じアカウントでログインし、そのホームディレクトリ下のetcに設定ファイル、binに実行ファイルを置くことを想定した作業例になっている。

ddupは、

```
$ ddup --host ドメイン名 --ip IPアドレス
```

とすることで、Dyndnsへの登録作業を行ってくれる。つまり、このコマンドラインをPPP接続が成立するたびに呼び出せばいいわけだ。今回使用しているPPxPでは、接続用設定スクリプトにちょっと書き加えるだけでカンタンに実現できる。PPxPの設定を行ったアカウントがh_siobar、設定名がiij4u-isdnだとすると、/home/h_siobar/.ppxp/conf/iij4u-isdnがスクリプトの実体ファイルになる。このファイルの末尾に、

```
set IP.UP myipup
```

という行を付け加える。これはIPレベルでの接続が完了したとき、myipupというスクリプトを起動するという意味だ。myipupの実体は「/home/h_siobar/.ppxp/ip/myipup」とし、**リスト1**のような内容とする。

これで呼び出されるのが「/home/h_siobar/.ppxp/rc/ddspawn」というシェルスクリプト（**リスト2**）である。このファイルは必ず実行属性をつけておくこと。実際にDyndnsでの作業を行うのがこのスクリプトで、「ドメイン名」の部分には、Dyndnsで作成したドメイン名を記述する。

なお、各リスト中のディレクトリは使用アカウント名に応じて変更する必要がある。

さて、設定が終わったら正しく動作するか確認を。

```
$ ppxp iij4u-isdn -C connect
```

としてダイヤルアップ接続したあと、Dyndnsのメンバーページにアクセス。該当ドメインの情報をチェックして、データベースに現在のIPアドレスが確かに登録されていれば成功である。これで、ダイヤルアップ接続が確立しているあいだはDyndnsに最新のIPアドレスが反映され、インターネット上のほかの利用者には、任意のドメイン名であなたのサーバにアクセスしてもらえるはずだ。

```
# tar xvfz ddup-3.0.1-unix.tar.gz
# cd ddup-3.0.1/
# ./install
Welcome to configuration for DD-UP.
What directory do you wish to install ddup.conf (CONFIGURATION FILE)
in?
Note, this directory name shouldn't end with a /
[/etc] -> /home/h_siobar/etc          ← 設定ファイルのインストール先
What directory do you wish to install ddup (program binary) in?
Note, this directory name shouldn't end with a /
[/sbin] -> /home/h_siobar/bin         ← 実行ファイルのインストール先
Compiling...this shouldn't take too long
Copying binary to /home/h_siobar/bin
Would you like to make a configuration file: (y/n)
y                                     ← 設定ファイル作成を指定
Creating config file...
Please enter your user name:
sinohara                              ← Dyndnsでのメンバーアカウント名
Please enter your password:
******                                ← Dyndnsでのパスワード
OK..now you need to enter the users who can use ddup:
Seperate the users with a , ie. root,test,this,that
root,h_siobar                         ← ddupの使用許可を与えるアカウント名
```

画面12　ddupのインストール

▼リスト1　myipupの内容

```
source drouteup
/home/h_siobar/.ppxp/rc/ddspawn $(IP.LOCAL)
```

▼リスト2　ddspawnの内容

```
#!/bin/sh
/home/h_siobar/bin/ddup --host ドメイン名 --ip "$1"
```

公開サーバの設定はどうする？

Dyndnsを利用することで、とりあえず希望のドメイン名で公開サーバを立ち上げることはできた。しかし、この状態ではWebなどサーバ側デーモンの準備ができていない。ここでは、いくつかのサーバ設定の勘所を見ていく。

■Webサーバ

ダイナミックDNSできちんと公開サーバを参照できるように設定しておけば、Webサーバの提供については何も難しいことはない。Apacheを起動し

ておけば、インターネット中からふつうにアクセスしてもらえる

たとえばRedHat Linux 7Jでは、

```
$ su
# ntsysv
```

として起動サービスの設定画面に移行し、httpdがきちんと起動するようにチェックを入れておけばよい。なお、標準状態では設定ファイルの一部に不備があるので、/etc/httpd/conf/httpd.conf中の

```
#ServerName localhost
```

とある行の先頭の「#」を削除し、再起動する必要がある。

Apacheは、参照のために使用されたドメイン名がなんであろうと、原則としてDocumentRootに設定されたディレクトリ下のコンテンツを返す。そのため、これ以外の設定は不要だ。RedHatのデフォルトでは/var/www/html/以下にHTMLファイルを配置するだけでよい（Apacheの設定の詳細については64ページからの記事を参照）。

これだけではつまらないので、Dyndnsで複数取得したドメイン名のそれぞれのWebサーバを、1台のマシンでまかなう方法を考えてみる。Apacheに搭載されている「ネームベースのバーチャルホスト」機能を利用するのだ。これを使えば、

- **hoge1.dyndns.orgへのWebアクセスでは、/var/www/hoge1以下を表示する**
- **hoge2.dyndns.orgへのWebアクセスでは、/var/www/hoge2以下を表示する**

といった設定が可能だ。何台もの専用サーバを持つブルジョア気分を味わえて、ちょっといいかも。DyndnsのWildcard設定と組み合わせると、結構おもしろい。また、

- **hoge3.dyndns.orgへのWebアクセスを、プロバイダに設置した個人サイトにジャンプ（リダイレクト）させる**

といった使い方も、もちろんできる。こういった設定を行ったhttpd.confの一部抜粋を**リスト3**に挙げておくので、参考にしてほしい。

■FTPサーバ

FTPサーバも、きちんと起動していれば、

```
$ ftp hogehoge.dyndns.org
```

とするだけでインターネットからふつうにアクセスできる。こちらのほうはApacheのようにカンタンにネームベースのバーチャルホストを実現することはできないので、ややおもしろみに欠けるが、ちょっとしたanonymous FTP用途にはいいかもしれない。

■メールサーバ

ダイナミックDNSを利用して立ち上げたサーバをメールサーバにするのは、あまりお勧めしない。特に、フレッツISDNのような接続が間欠的になりやすい環境では、いざというときに他のドメインからのメールをサーバが受け取れず、送信元にエラーが返ってしまうこともあるからだ。

WebやFTPでは、たまたまつながらなくてもあまり他人に迷惑をかけることはないが、メールは違う。不安定な環境で作成したメールアカウントで、メーリングリストなどのコミュニティに参加するのは、思わぬトラブルの元になりかねない。

どうしてもという人や、安定した通信環境に恵まれているという人は、細心の注意を払ってメールサーバの設定を行おう。メールサーバの場合は、サーバ側で「このドメイン宛のメールは受信する」ときちんと設定しなければ、まともに動作してくれない。qmailの場合、/var/qmail/controlにある

▼リスト4　rcpthosts（qmailの設定ファイル）の例

```
localhost
hogehoge.dyndns.org
```

▼リスト3　httpd.confの例

```
                :
NameVirtualHost hoge1.dyndns.org

<VirtualHost hoge1.dyndns.org>
ServerName hoge1.dyndns.org
DocumentRoot /var/www/hoge1
</VirtualHost>

<VirtualHost hoge1.dyndns.org>
ServerName hoge2.dyndns.org
DocumentRoot /var/www/hoge2
</VirtualHost>

<VirtualHost hoge3.dyndns.org>
ServerName hoge3.dyndns.org
Redirect / http://www.provider.com/user256/
</VirtualHost>
```

“rcpthosts”と、同一マシン上のメールボックスにメールを配信するなら“locals”にドメイン名を記載しておく必要がある。

ちなみに、Dyndnsの「Backup MX」設定をうまく利用すると、自分のサーバがインターネット接続していないときには、ほかのサーバに代わりに受け取っておいてもらうようにもできる。ただし、そのサーバの管理者とお互いに協力しあって設定を行わなければならない。このあたりはメールサーバ関連の資料をあさって調べてみてほしい。

ルータ環境での利用

最近では家庭に1台以上のパソコンがあるのが当たり前になってきている。こういった環境では、ルータを使ってインターネット接続を行うのが一般的だ。また、より高速の常時接続サービスでは、これまたルータを介してネットにつなぐことが多い。このようなルータ環境では、今回のように直接LinuxマシンにつながったTAを介してダイヤルアップする場合のやり方を当てはめることはできない。とはいえ、ルータ環境でもダイナミックDNSを使ってサーバを公開することは可能だ。誌面の制限から詳細は説明できないが、2つのポイントを挙げておこう。

■ルータの「静的マスカレード」機能

ルータの外側（＝インターネット側）から接続要求がきたとき、条件に応じてLAN内の特定マシンに転送する機能を静的マスカレードと呼ぶ（「IPフォワーディング」など、メーカーによって呼称は異なる）。Webサーバを公開するなら、この機能を使って外部からのリクエストを公開用サーバに転送するよう設定する必要がある。

■ルータが取得したIPアドレスの確認

グローバルIPアドレスを取得しているのは、この環境ではルータである。そこで、ルータのIPアドレスを確認して、ddupなどのツールを使いダイナミックDNSサービスに登録するしくみが必要になる。公開用サーバとなるLinuxマシンから作業を行うとして、コマンドラインで動作するツールが欲しいところだが、メーカーから公開されている例はあまりない。telnet経由での作業に対応しているルータなら、PerlのNet::Telnetモジュールを利用すれば、チャットスクリプトを自作できるはず。また、Webベースの設定が可能なルータならば、wgetでルータの設定情報を取得できる。

ちょっと苦労しそうだが、サーチエンジンで検索すれば同様の事例が何件かヒットするだろう。ルータ環境ならセキュリティ上の危険性もやや緩和されるので、試行錯誤しつつ挑戦してみる価値はある。

そのほかのダイナミックDNSサービス

Dyndns以外にも、インターネットではさまざまなダイナミックDNSサービスが存在する。たとえば、Dyndnsは英語のサービスなので言語的な敷居が高かったが、国内にも「dyn.to」というサービスが存在する。当然日本語なので、安心して利用できる（ただし有料）。

また、ダイナミックDNSの使い勝手を増す、さまざまな機能が売りのサービスも存在する。たとえば「オフライン時Web転送」は、ダイナミックDNSの最大の問題である「つながっていなければどうにもならない」という点を解決するもの。登録サーバが利用できない状態にある場合、Webへのアクセスリクエストを任意のURLに転送してくれるのだ。

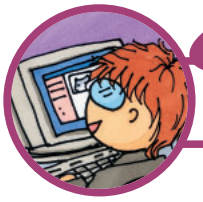
「メール転送」機能は、不安定な接続環境でわざわざメールサーバを構築しなくても、自ドメイン宛のメールを特定のメールアドレスにまとめて転送してくれる。メールアドレスだけプロバイダのモノ、といったカッコ悪い状態も回避できて便利だ。さらに、Dyndnsのように用意されたドメインのサブドメインというかたちではなく、自分で所有するドメイン全体をダイナミックDNSで管理してくれる「フルドメインサポート」を提供するサービスもある。

これらのサービスのごく一部を**表1**にまとめてみた。参考にしつつ、自分に最適なサイトを選んで、「ダイナミックDNSでマイサーバ」を満喫してほしい。

（しのはらひろあき）

名称	URL	メールサーバ登録	オフライン時Web転送	メール転送	フルドメインサポート	国内サービス？	料金
Dyndns	http://www.dyndns.org/	○	−	−	−	海外	無料
No-IP	http://www.no-ip.com/	−	−	−	−	海外	無料
HN.ORG	http://hn.org/	○	○	−	−	海外	無料
DtDNS	http://www.dtdns.com/	○	−	−	○（年間20ドル）	海外	原則無料
dyn.to	http://dyn.to/	○	−	−	−	国内	月額200円
ZiVE.Org	http://www.zive.org/	○	−	○	−	国内	無料

表1　ダイナミックDNSサービスの例



番外編

ちょ〜簡易公開サーバ作成法

非固定のグローバルIPアドレスでも、Webページを公開する程度ならダイナミックDNSサービスを使わなくても実現できる。ここでは、超簡易式のサーバ公開法をお教えしよう。

仕組みは簡単だ。公開したいマイサーバのIPアドレスがわかれば、どこからでも接続できる。したがって、IPアドレスが更新されるたびに、プロバイダに置いたWebページにマイサーバの最新IPアドレスを書いておけばいいわけだ。

もちろん、更新作業は自動化したい。マイサーバでIPアドレスを監視し、変更されたらプロバイダのWebページを書き換える。CGIやSSIを使い、Webページの中にマイサーバへのリンクを埋め込むのがスマートだが、CGIやSSIが使えない場合は、**リスト1**のようなHTMLファイルを自動生成して、FTPで送り込めばいいだろう。

DHCPを利用している場合

DHCPクライアントソフトのpumpやdhcpcdには、IPアドレスが変わったときにプログラムを実行する機能がある。この機能を利用して、ジャンプHTMLファイルの自動更新を行ってみよう。pumpは、Red Hat系のディストリビューションで標準的に使われているDHCPクライアントだが、動作が不安定になることが多いので、ここではdhcpcdを例に説明する。pumpを使いたい方はmanページを参考にチャレンジしてほしい。

dhcpcdもまた、多くのディストリビューションに含まれているDHCPクライアントだ。Red Hat系のディストリビューションでは、dhcpcdをインストールし、pumpをアンインストールすればそのまま動作する。dhcpcdは、割り振られているIPアドレスが変更されると、新しいIPアドレスを引数にして、/etc/dhcpc/dhcpcd-*.exeというファイルを実行する。*には、eth1などのインターフェイス名が入る。

リスト2は、新しいIPアドレスを埋め込んだ強制ジャンプHTMLファイルを作成して、FTPでプロバイダのWebサーバに転送するシェルスクリプトの例だ。

シェル変数UPFILEに作成するHTMLファイルのパスを設定し、echoコマンドをリダイレクトしてHTMLファイルを作成。「$1」の部分は、マイサーバの新しいIPアドレスに展開される。ファイルができたら、ftpコマンドとヒアドキュメントを使ってプロバイダのWebサイト（この例では、ftp.ascii-linux.com）に転送する。このシェルスクリプトには、パスワードが平文で書かれるので、rootユーザー以外には読めないようにパーミッションを設定しよう（もちろん、実行属性も必要だ）。

こうしておけば、プロバイダのWebサイトにある「jump.html」にアクセスすると、即座にマイサーバにジャンプする。このファイルにリンクしてもいい。

ルータやPPPoEで接続している場合

ISDNルータやADSLルータで接続している、あるいはTAやADSLモデムとLinuxマシンの間にルータを挟んでいるネットワーク構成では、違う手順が必要になる。この場合、何らかの方法

▼リスト1　IPアドレスXXX.XXX.XXX.XXXのWebサーバに飛ばすHTMLファイル

```
<HTML>
<meta http-equiv='refresh' content="0; URL=http://XXX.XXX.XXX.XXX/">
</HTML>
```

▼リスト2　シェルスクリプト/etc/dhcpc/dhcpcd-*.exeの例

```
#!/bin/sh
UPFILE="/tmp/jump.html"                    作成する強制ジャンプHTMLファイルのパス

echo "<HTML>" > ${UPFILE}                  強制ジャンプHTMLファイルを作成する
echo "<meta http-equiv='refresh' content=\"0; URL=http://$1/\">" >> ${UPFILE}
echo "</HTML>" >> ${UPFILE}

ftp -n ftp.ascii-linux.com << EOF_FTP      ヒアドキュメントでftpのコマンドを実行
user p-chan Na1sH0D3sU                     ユーザー名とパスワード
ascii                                      転送モードはアスキー
prompt                                     非対話モードにする
put ${UPFILE} jump.html                    作成したHTMLファイルを転送する
quit                                       終了
EOF_FTP
```

でプロバイダから割り当てられたIPアドレスを知ることが必要だ。最近のほとんどのルータはWebブラウザで設定を行うようになっているが、もし現在のIPアドレスを表示する画面があるならば、そこからIPアドレスを取得することが可能だ。例としてeAccessのADSLルータの例を挙げてみよう。

wgetでルータからIPを取得

wgetはWebサイトやFTPサイトから自動的にファイルをダウンロードしてくれるツールで、ほとんどのディストリビューションに標準でインストールされているはずだ。このwgetを使えば、ルータのIPアドレス表示画面をまるごとテキストファイルとして取得することができる。ルータの設定画面はパスワード制限がかかっているだろうが、それがBasic認証という認証方式ならば（たいていはそうだろう）、それもwgetが代わりに入力してくれる。パスワード認証が通れば、本物の画面がダウンロードされる。

wgetの設定は、ホームディレクトリに置く.wgetrcというファイルで行う（**リスト3**）。**リスト3**の<username>と<password>をルータのユーザー名とパスワードに置き換えれば自動的に送信される。次の行の「recursive = off」は、そのページのリンクをたどらないという指定だ。このファイルにはパスワードをそのまま書き込むので、必ずchmod 600 .wgetrcとして自分以外見られないようにしておくこと。

次に、wgetを起動するシェルスクリプト（**リスト4**）を用意する。この例では、スクリプトをroot権限で動かすようにしてある。スクリプトや各種ファイルは/etc/adslというディレクトリを用意して、そこで実行するので、**リスト4**の4行目で/etc/adslに移動する。5行目では、メッセージを出さずに(-q)、「http://」以下のWebページを取得し、.routerというファイルに書き込む（-O .router）というパラメータを与えてwgetを起動している。このURLはeAccessのMegaBitGear TE4111CというルータのIPアドレス表示画面のURLなので、この部分は各自のルータに合わせて変える必要がある。

これでIP情報表示画面を取得できた。そこからIPアドレスを取得するのが最後の行だ。wgetでダウンロードした.routerファイルを、awkのスクリプトに渡している。スクリプトを書いたら、

```
# chmomd 744 getip
```

として必ず実行属性を与えるのを忘れないこと。

IPの切り出しとサイトへの送信

リスト5はそのawkスクリプトだ。IPアドレスはいつも決まったフォーマットでWeb画面に表示されるはずだ。このルータ場合、「ADSL IP</TD><TD NOWARP>:」のあとにIPアドレスがくるので、それを利用して4〜6行目でIPアドレスだけを切り出している。ここの部分はダウンロードしたファイルをよく見て各ルータに合わせた指定にする必要がある。正規表現の意

▼リスト3　.wgetrc

```
http_user = <username>
http_passwd = <password>
recursive = off
```

▼リスト4　getip

```
01 #!/bin/sh
02 # getip
03
04 cd /etc/adsl
05 if ! wget -q -O .router http://192.168.1.1/cgibin/main.cgi?cc_webname=STATUS
06 then
07    exit
08 fi
09 awk -f putip.awk .router
```

▼リスト5　putip.awk

```
01 #!/bin/awk
02 # putip.awk
03 {
04     if ($0 ~ /ADSL IP<\/TD><TD NOWARP>:/) {
05         $0 = substr($0, index($0, ":")+1);
06         CurrentIP = substr($0, 1, index($0, "<")-1);
07         if (CurrentIP !~ /^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$/) { exit }
08         getline FormerIP < ".ipaddress";
09         if (CurrentIP != FormerIP) {
10             print CurrentIP > ".ipaddress";
11                 url = "http://<CGIディレクトリ>/getip.cgi?"CurrentIP;
12             if (system("/usr/bin/wget -q -O /dev/null "url) != 0) { exit }
13         }
14         exit;
15     }
16 }
```

味や、awkの関数についてはここで触れることはできないので、関連書籍などを参考にしてほしい。また、今回はたまたまawkを使ったが、もちろんPerlやRubyでもまったく同じことが可能だ。さらに、PerlやRubyを使えばwgetの代わりまでやらせることができる。興味ある方は挑戦してみてほしい。

さて、7行目では切り出したIPアドレスをチェックしている。ちょうど接続をしようとしている最中などには、IPアドレスの代わりに「接続中」などの表示が入るからだ。

これでIPアドレスを.ipaddressというファイルに書き出すことができた。これ以降はIPアドレスを自分のホームページに送る作業に入る。この例では、CGIを使えるプロバイダを想定しているが、そうでない場合は前の例のようにFTPを使えばいい。

サイトへ送信

IPアドレスをプロバイダのサイトに送るのには、CGIを利用することにする。具体的には、CGIのURLにIPアドレスをパラメータとして付加する。ちょうどフォームに文字を入力して「送信」ボタンを押したのと同じ動作だ。もっとも、人間がボタンを押すことはできないので、これにもwgetを使う。**リスト5**の11行目は、サイトに置いたCGIのURLを指定している。<CGIディレクトリ>の部分は各プロバイダの指定する自分用のCGIディレクトリに置き換えること。そして次の行でURLにIPアドレスを付加して（URLとパラメータの区切りは「？」）送信する。この場合、ページを取得する必要はないので、Webサーバから受け取ったデータは破棄する（「-O /dev/null」）。

もし、ダイナミックDNSのサービスを使っているなら、11行目と12行目を書き換えて、ダイナミックDNSサービスの書き換えツールなどをここに書いておけば、自動でDNSの設定が変更できる。また、メールサーバが動いているなら、書き換わったアドレスをメールで送るということも可能だ。

サイト側のCGI

最後にwgetから送られてくるIPアドレスを受け取るPerlのCGI（**リスト6**）を用意する。PerlのパスやCGIの置き場所などはプロバイダによって違うので、1行目と6行目は適宜書き換えてほしい。このCGIは、ユーザーのホームページエリアのIPディレクトリの下にIPアドレスが書かれたip.htmlというHTMLファイルを作る。ユーザーはこのページにアクセスすることで、現在のIPアドレスを知ることができるし、IPアドレスの部分をクリックすれば、マイサーバにジャンプする。

もうちょっと凝って、受け取ったIPアドレスは別ファイルに書き込み、Web上のパスワード認証が通ったときだけそれを読み込んでダイナミックにページを表示する、というふうにするのもいいだろう。もっぱらメールサーバやSSHサーバとして使うので、できればIPアドレスは隠しておきたいなどというときには有効だ。その際、IPアドレスを書き込むファイルは先頭文字が「.」のファイル名にしておくと、Webブラウザで読まれることがないのでさらに安心だ。

最後に**リスト4**のシェルスクリプトをcronを使って定期的に実行するように設定すれば終わりだ。

PPPユーザーはifconfig

ルータを使わず、PPxPやPPPoEツールでISDNやADSLに接続している場合は、DHCPの機能もwgetも使えないので、ifconfigコマンドの出力をリダイレクトして、IPアドレスを切り出すというのがいいだろう。手順は先ほどの例が参考になるはずだ。

（編集部）

▼リスト6　getip.cgi

```
01 #!/usr/local/bin/perl
02 # getip.cgi
03 print "Content-Type: text/html\n\n";
04 $ip = $ENV{QUERY_STRING};
05 if ($ip ne "") {
06     open(FILE, ">$ENV{'HOME'}/html/IP/ip.html");
07     print FILE '<html>', "\n";
08     print FILE '<head>', "\n";
09     print FILE '<title>IP Address</title>', "\n";
10     print FILE '</head>', "\n";
11     print FILE '<body bgcolor="#FFFFFF">', "\n";
12     print FILE '<a href="http://';
13     print FILE $ip;
14     print FILE '/">';
15     print FILE $ip;
16     print FILE '</a>', "\n";
17     print FILE '</body>', "\n";
18     print FILE '</html>', "\n";
19     close;
20 }
```

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4回　WebライブカメラとMPEGエンコーダの準備

のっけからお詫びで申し訳ないが、今回も内容が予告とは違うものになってしまった。今度こそMPEG-1形式での録画に挑戦する予定だったのだが、LinuxのMPEGエンコーダを取り巻く状況はなかなか複雑で、筆者の力量不足もあって、本誌の締め切りまでに確実な設定方法を見つけ出すことができなかった。そこで、再び予定を変更して、TVキャプチャカードとビデオカメラを使った「Webライブカメラ」の作成方法を紹介する。それだけでは申し訳ないので、LinuxにおけるMPEGエンコーダの現況についても説明する。

文：竹田善太郎
Text : Zentaro Takeda

Webライブカメラの作り方

「Webライブカメラ」とは、単に「ライブカメラ」と呼ばれることも多いが、ビデオカメラやデジタルカメラで撮影した画像を、リアルタイムにWebページ上に表示させるシステムのことだ。各地の風景や自分のペットの様子などをインターネット上に公開したり、事務所や留守宅の監視用に設置したりすることが多い。

Webライブカメラを実現するには、**図1**のようなハードウェアとソフトウェアが必要になる。WindowsやMacintoshマシンでWebライブカメラを構築する場合、比較的安価なUSBカメラが利用できる場合が多く、ソフトウェアについてもフリーで使えるものが流通している。

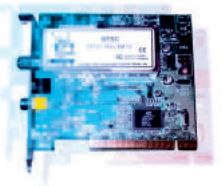
Linuxで同じことをしようとする場合、現状ではUSBカメラの利用は簡単ではないので。ビデオキャプチャカードとビデオカメラが必要になる。ソフトウェアとしては、静止画のキャプチャを行うソフトウェアと、キャプチャした画像をWebサーバに転送するソフトウェアが必要になる。

この連載の最初に説明したように、Linuxでビデオキャプチャカード（TVキャプチャカード）によるビデオ（テレビ）画像の表示を行う「xawtv」というソフトウェアは、動画だけでなく、静止画をキャプチャする機能も持っている。さらに、静止画キャプチャとWebサーバ（FTPサーバ）への転送を自動的に行う、「webcam」という便利なコマンドが付属しているのだ。webcamコマンドは、xawtvをコンパイル／インストールすると同時にインストールされるので、個別にインストールする作業は必要ない。このコマンドを使えば、だれにでも簡単にWebライブカメラシステムを構築できるのだ。

●webcamコマンドの概要

webcamコマンドは、xawtvで利用可能なキャプチャカード（すなわち、bttvドライバで利用可能なキャプチャカード）から静止画（JPEG形式）をキャプチャして、その画像データを指定したWebサーバにFTPプロトコルで転送する（**図2**）。キャプチャと転送は、指定した間隔で自動的に行われる。

画像のキャプチャは、TV放送、ビデオ入力のいずれで行うこともできる。このため、たとえばRFコンバータなどでチューナーのアンテナに接続したり、UHFの微弱電波を使って映像を送信するタイプのTVカメラでも、Webライブカメラの入力源として使うことができる。今回の実験では、ハンディタイプのビデオカムコーダを入力源として使ったが、ビデ



オ出力端子を備えたデジタルカメラなども利用できる。

webcamコマンドは、コマンドそのものにタイマーの機能が含まれており、コマンドが実行されているあいだ、あらかじめ指定した間隔ごとに画像を送信し続けるようになっている（**図3**）。このため、crontabなどに登録して定期的にwebcamコマンドを起動する必要はない。ただし、この場合は、Webライブカメラを更新する場合、webcamコマンドを実行するマシンにログインして、明示的にwebcamコマンドを実行するか、あるいは、/etc/rc.d/rc.localなどから、webcamコマンドをバックグラウンドで実行するように指定しなければならない。

このような使い方に抵抗がある場合は、webcamコマンドを起動したときに1回だけ画像を送信して、すぐにコマンドを終了させるようにオプションで設定できる。そのように設

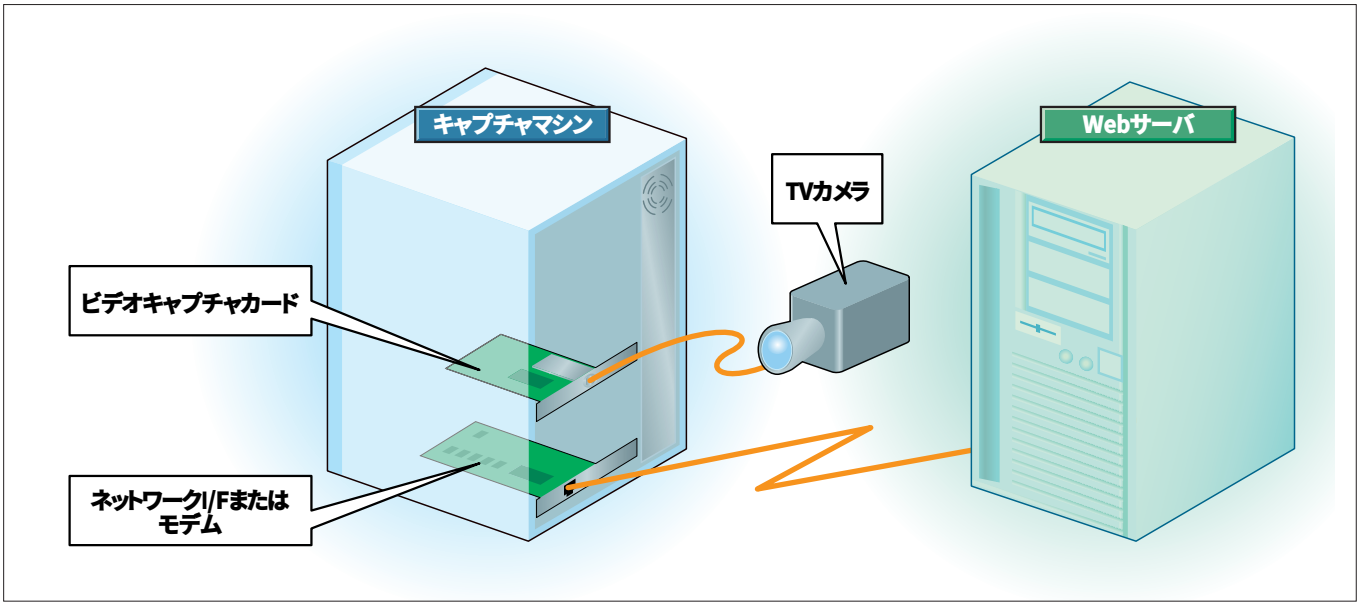


図1　Webライブカメラを構築するのに必要なハードウェア

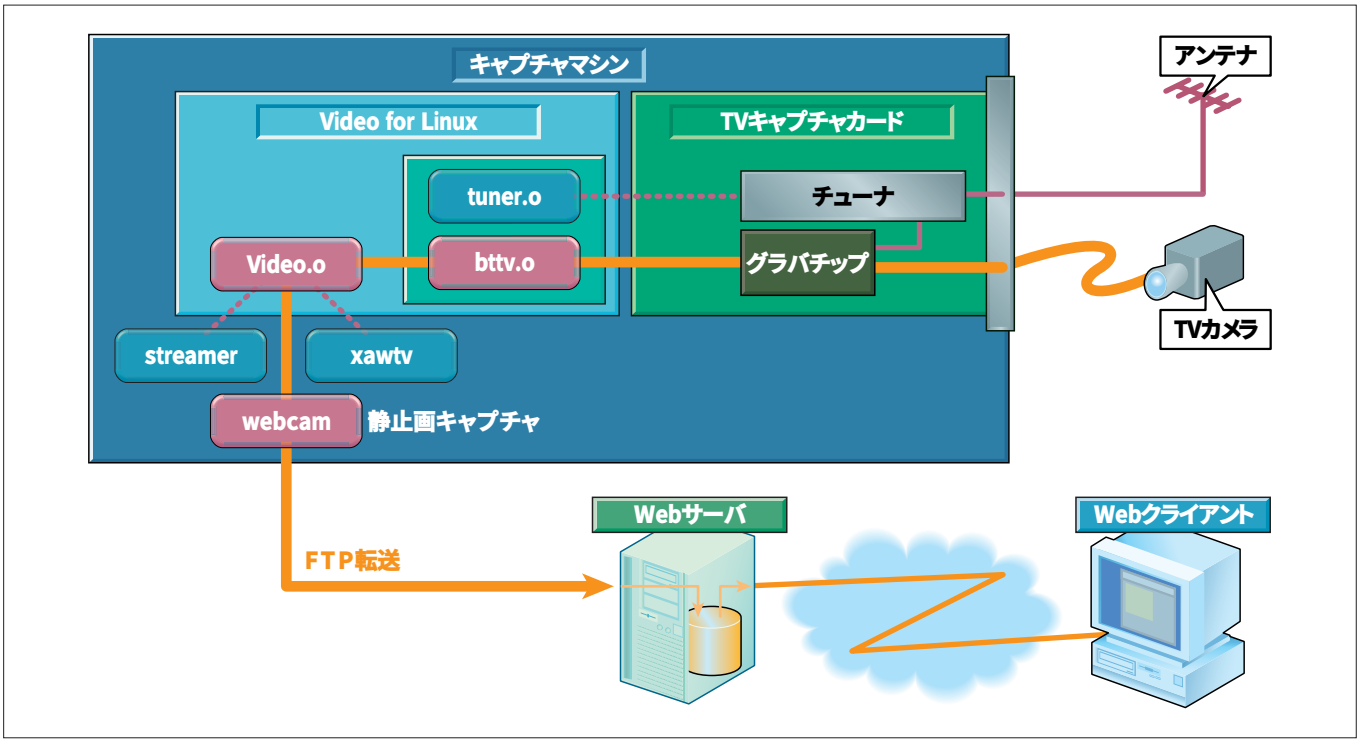


図2　webcamコマンドのしくみ

定してから、crontabで定期的にwebcamを起動するように設定すれば、無人での運転にも支障はない（**図4**）。この方法については、後ほど述べる。

転送先のWebサーバは、ftpコマンドでログインできさえすれば、どのようなものでもかまわない。たとえば、ダイヤルアップ接続でインターネットを利用しているような場合でも、（自動ダイヤルアップの設定さえできていれば）定期的に画像を更新することができる。ファイアウォールやダイヤルアップルータを介してインターネットに接続している場合であっても、FTPさえできればWebカメラは実現可能である。

●.webcamrcの記述

webcamコマンドの動作は、ユーザーのホームディレクトリ上に置いた.webcamファイルに記述する。あるいは、任意のテキストファイルにオプションを記述して、webcamコマンドを実行するときに、コマンドライン引数としてそのファ

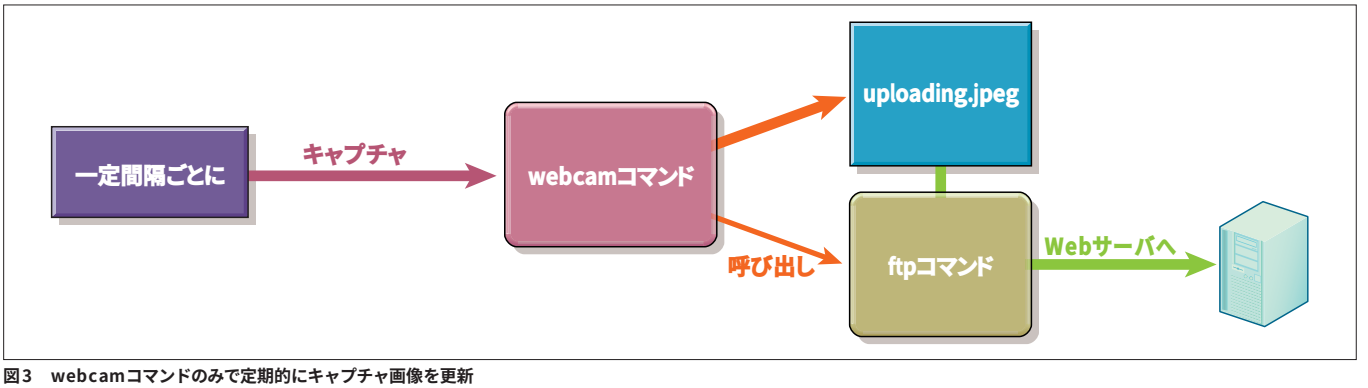


図3　webcamコマンドのみで定期的にキャプチャ画像を更新

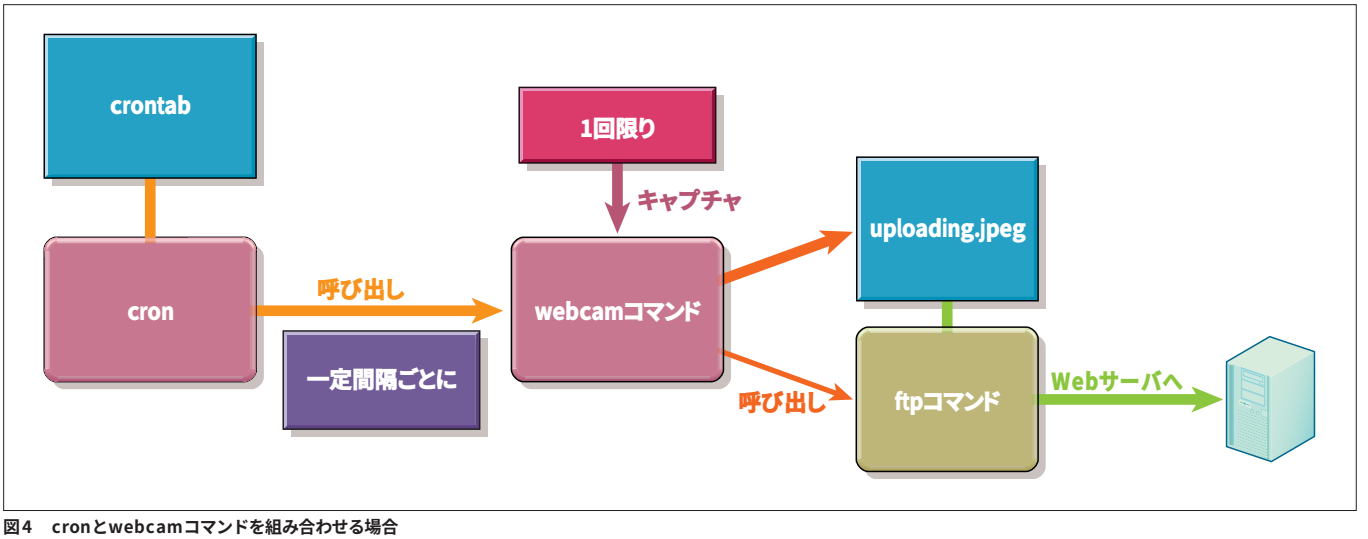
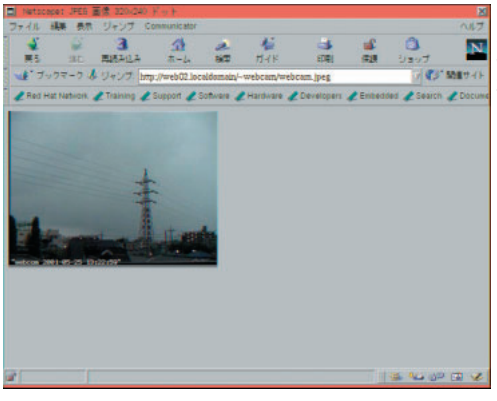
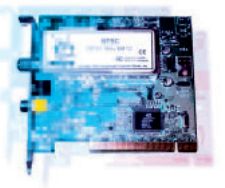


図4　cronとwebcamコマンドを組み合わせる場合



画面1　Webサーバに転送された画像を確認する
webcamコマンドが定期的に転送するタイミングを見て「再読み込み」ボタンなどをクリックして、画像が更新されるかどうかを確認する。

画面2　webcamコマンドの実行中
webcamコマンドをコマンドラインから実行すると、画像をキャプチャして転送するたびに、その時点の日時が表示される（標準エラー出力）。



イルの名前を与えてやってもよい。

なお、.webcamファイルには、FTPサーバへのログイン用ユーザー名やパスワードを記述することになるので、他人に読まれないようにアクセス権を設定する必要がある。

```
$ chmod 600 ~/.webcamrc
```

.webcamrcファイルの記述例をリスト1に掲載する。このリストの内容を見れば、だいたい意味はわかると思うが、この例では、3分ごとに「web02.localdomain」というホスト（Webサーバ）に「webcam」というユーザー名で「public_html」ディレクトリに「webcam.jpeg」というファイル名で画像を転送している。キャプチャするのはビデオ入力端子（composit1）に接続したカメラの映像だ。

.webcamrcファイルは、［ftp］セクションと［grab］セクションの2つからなるが、各セクションごとの主要な設定値について、リスト2にまとめておいたので参考にしてほしい。

.webcamrcファイルを記述してしまえば、あとはwebcamコマンドを実行するだけで、テレビカメラの映像が定期的にWebサーバに転送されるようになる。適当なWebブラウザ

▼リスト1　.webcamrcファイルの例

```
[ftp]
host = web02.localdomain
user = webcam
pass = xxxxxxxx
dir  = public_html/
file = webcam.jpeg
tmp  = uploading.jpeg
passive = 1
debug = 0
auto = 0
local= 0

[grab]
device = /dev/video
text = "webcam %Y-%m-%d %H:%M:%S"
width = 320
height = 240
delay = 180
input = 1
norm = 1
rotate = 0
top = 0
left = 0
bottom = -1
right = -1
quality = 75
trigger = 0
once = 0
```

▼リスト2　.webcamrcファイルに設定できるオプション（主要なもの）

●[ftp]セクション
このセクションでは、キャプチャした画像を転送する先のFTPサーバ（すなわち、公開するWebページのWebサーバ）や、転送時するファイルの名前などを設定する

- host = [ホスト名]
  転送先のFTPサーバのホスト名を記述する
- user = [ユーザー名]
  転送先FTPサーバにログインするときのユーザー名を記述する
- pass = [パスワード]
  転送先FTPサーバにログインするときに使うパスワードを記述する。パスワードを直接記述するので、このファイルの中身は他人には読めないように設定しておく必要がある
- dir = [転送先ディレクトリ名]
  転送先FTPサーバ上で、Webライブカメラの画像ファイルを配置するディレクトリを指定する
- file = [転送先ファイル名]
  転送する画像ファイルのファイル名を指定する
- tmp = [一時ファイル名]
  転送する前にローカルマシンに画像を一時的に保存するときのファイル名を指定する

●[grab]セクション
このセクションでは、画像をキャプチャする方法について設定する

- device = [キャプチャデバイスのデバイスファイル名]
  通常は、/dev/videoを指定する

- text = [画像にスーパーインポーズする文字列]
  キャプチャした画像に埋め込む文字列を指定する。次のフォーマットで日時などを埋め込むことができる
  %Y = 西暦
  %m = 月
  %d = 日
  %H = 時
  %M = 分
  %S = 秒
  たとえば、「MyWebCamera 2001/5/23 23:40:00」というような文字列を埋め込みたい場合は、

  ```
  text = "MyWebCamera %Y/%m/%d %H:%M:%s"
  ```

  というように指定すればよい。

- width = [幅]
- height = [高さ]
  キャプチャする画像の大きさをピクセル単位で指定する

- delay = [撮影間隔（秒）]
  画像をキャプチャしてFTPサーバに転送する間隔を秒単位で指定する

- input = TV | composite1
  キャプチャ先の信号（チューナー、ビデオ入力など）を指定する

- norm = 0 | 1 | 2
  画像信号の形式を設定。NTSCの場合は「1」を指定すればよい

- quality = [画像品質]
  キャプチャするJPEG画像の品質を1〜100の数値で指定する。数値が大きいほど品質は高くなり、画像ファイルのサイズも大きくなる

- once = 0 | 1
  画像を継続的にキャプチャするのか、1回だけキャプチャするのかを指定する。0を指定すると継続キャプチャ、1を指定すると1回だけのキャプチャになる

で、転送した画像のURL（この例の場合は、http://web02.localdomain/~webcam/webcam.jpeg）を参照して、画像が転送されていることを確認してみよう（**画面1**）。webcamコマンドを実行すると、画像の転送を実行するたびに、転送を実行した時刻が画面上（標準エラー出力）に表示される（**画面2**）。バックグラウンドでwebcamコマンドを実行する場合には、次のようにして、ログファイルに記録させるようにすればよいだろう。

```
$ webcam 2> /tmp/webcam.log &
```

●無人運転をしたい場合の設定

リスト1の.webcamrcファイルでは、webcamコマンドを起動しているあいだ、3分おきに画像を更新するようになっている。前述したように、この場合はWebライブカメラを使うには、Linuxマシンにログインして、コマンドラインからwebcamコマンドを実行するか、あるいは、/etc/rc.d/rc.localファイルなどに、

```
(su tvmaster; webcam 2>> /tmp/webcam.log &)
```

というような行を追加する。

cron機能を使って一定間隔で実行させたいような場合には、.webcamrcファイルの「once=」の値を「1」に設定する。こうすると、「delay=」にどのような値を設定していても、1回キャプチャを行った時点でwebcamコマンドは終了するので、cronから定期実行するのに都合がよい。あとは、「crontab -l」を実行して、crontabに次のような行を追加すればよい。

```
0-59/3 * * * * webcam
```

●Webサーバとキャプチャマシンが同一の場合

画像キャプチャを実行するマシンがWebサーバを兼ねているような場合もあるだろう。このような場合でも、FTPによる転送は可能（自分自身にFTPでログインしてファイル転送する）なのだが、あまりスマートではない。

このような場合には、.webcamrcファイルの「local=」の値に0以外の値（たとえば、local=1）を設定すれば、ファイルはFTPで転送される代わりに、「dir=」で指定されたディレクトリに「file=」で指定されたファイル名でコピーされる。

Webサイト側の準備

これだけでも、簡単なWebライブカメラが実現できてしまうわけだが、このままでは、画像を更新するのに、いちいちブラウザの「更新」ボタンをクリックしなければならず、少々やっかいだ。このため、ライブカメラの画像を表示して、表示を定期的に更新するための「枠」となるWebページを作っておこう。

●自動リロードするHTMLページ

リスト3のようなHTMLファイルを、Webサーバ上のWeb公開用ディレクトリ（たとえば、~/public_html/）に作成する。このHTMLファイルは、最小限のタイトルなどを配置してから、webcamコマンドが転送してくるJPEGファイルを表示するだけのものである。また、3分間隔でページ全体をリロードするようになっているので、画像ファイルが更新されるのにほぼ同期して、表示内容も更新されるようになっている。

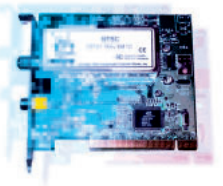
もちろん、Webページの作成に慣れているのであれば、フレームなどを活用して見やすくしたり、ページ全体を再ロードする代わりに、JavaScriptなどを使って画像の部分のみを更新するようなことは、簡単にできるだろう。各自の興味に応じて、挑戦してもらいたい。

▼リスト3　Webサーバ上の「~/public_html/webcam.html」

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN" "http://www.w3.org/TR/REC-
html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>MyWebcam</TITLE>
<META http-equiv="refresh"
content="180;URL=http://web02.localdomain/~webcam/webcam
.html">
</HEAD>
<BODY>
<H1>今日のさいたま</H1>
<IMG src="http://web02.localdomain/~webcam/webcam.jpeg"
alt="live view">
</BODY>
</HTML>
```

「防犯カメラ」として使う方法

Webライブカメラの用途のひとつとして、風景などを他人に公開するのを目的とするのではなく、留守中の自宅などを



「監視」したり、不審人物などの侵入を「記録」するのを目的とする、いわゆる「防犯カメラ」を目的に使う場合もある。このような場合は、撮影した画像をあとから参照できるようになっていないと意味がない。いままで紹介したwebcamの設定では、新たに映像をキャプチャするたびに、画像ファイルが上書きされてしまうので、このような目的には使えない。

残念ながら、webcamコマンド自体には、画像ファイルのファイル名を変えて、複数のファイルをログとして残すような機能はない。このため、webcamコマンドによるキャプチャの実行後に、ファイル名を変更して上書きされないようにするスクリプトを追加する必要がある。

このような場合、webcamコマンド自身のタイマー機能を使っていると、キャプチャしたファイルの名前を変更するタイミングの設定が難しいので、ここでは、cronによる自動実行でwebcamコマンドを使う。また、キャプチャしたファイルはFTPで転送せずに、ローカルマシンのディスク中に残すことにする。このような場合のwebcamの設定ファイル（.webcamrcではなく、spycamという名前のテキストファイルにする）は、**リスト4**のようになる。

ファイル名を変更するシェルスクリプト（spyfilemv）は**リスト5**の通りである。このスクリプトをwebcamコマンドの直後に実行すれば、/tmp/webcam/webcam.jpegファイルは、日付と時刻を含んだ「webcam20010523104500.jpeg」のような形式のファイルにmvされる。

以上のようなファイルを用意したうえで、crontabに次のような行を追加すればよい（3分おきに撮影する場合）。

```
0-59/3  *  *  *  *  (webcam  /home/tvmaster/spycam;
/home/tvmaster/spyfilemv)
```

次回に向けて MPEGエンコーダの現況

さて、次回こそはMPEG-1形式の録画をご紹介できることを祈りつつ、その助走としてLinuxにおけるMPEGの状況を概説しておこう。

MPEGとは、主に動画データの圧縮形式の標準化を行っている団体のことで、MPEG-1、MPEG-2、MPEG-4などの圧縮規格が現在利用されている。MPEG-1はVideo CDなどで、MPEG-2はDVDやデジタル衛星放送などで利用されている。また、MPEG-4は、インターネットや次世代携帯電話での動画配信などで使われ始めている。

MPEGでどのような原理で動画データの圧縮を行っているかについて、ここでは詳しく述べる余地はない。簡単にいってしまえば、連続するフレーム（静止画像）の比較をして、前のフレームから変化している部分だけを抽出して、それ以外のデータは省略するなどの方法で、データの量を減らしている。

パソコンで動画を扱う場合、最近のパソコンの性能からいえば、MPEG-1、MPEG-2、MPEG-4のどの形式でも問題なく扱える。たとえば、TV放送をハードディスクレコーディングする場合ならMPEG-1、DVカメラの映像を編集するのならMPEG-2、ネットワーク経由で公開する映像データを作るなら、MPEG-4といったぐあいに、うまく使い分ければよいのだ。

MPEG形式のデータを作成したり再生したりするには、「エンコーダ」や「デコーダ」（これらをまとめて「コーデック」と呼ぶこともある）と呼ばれるソフトウェア（あるいはそのような機能を備えた専用のハードウェア）が必要になる。

▼リスト4　spycamファイル

```
[ftp]
dir  = /tmp/webcam
file = webcam.jpeg
tmp  = uploading.jpeg
passive = 1
debug = 0
auto = 0
local= 1

[grab]
device = /dev/video
text = "webcam %Y-%m-%d %H:%M:%S"
width = 320
height = 240
delay = 30
input = 1
norm = 1
rotate = 0
top = 0
left = 0
bottom = -1
right = -1
quality = 75
trigger = 0
once = 0
```

▼リスト5　spyfilemvスクリプト

```
#!/bin/bash
mv /tmp/webcam/webcam.jpeg `date +"/tmp/webcam/webcam%Y%m
%d%H%M%S.jpeg"`
```

市販されているビデオ編集ソフトやTVキャプチャ製品には、これらのMPEGコーデックがハードウェアあるいはソフトウェアの形で含まれている場合が多い。

パソコンをWindowsで使っている場合には、このような製品を購入して使えば、すぐにでもMPEGデータを作成したり再生したりできるのだが、Linuxの場合は、残念ながら製品として購入できるものは存在しないに等しい。

また、MPEGの中の音声圧縮技術である「MP3」でも問題になっているが、MPEGの圧縮技術は「標準規格」でありながら、特許権で保護されている。このため、コーデックの開発や配布には、特別の契約や高額のライセンス料が必要になる場合がある。

このような事情から、また、動画を扱うという複雑さから、MPEGコーデックの開発は、資金力のある企業が中心で、自社の製品に組み込んだ形でしかコーデックを公開していない場合も多い。

このような困難な状況にありながら、Linux上で自由に使えるMPEGコーデックの開発を行っている、個人やボランティアベースのプロジェクトがいくつか存在している。その代表的なものをいくつか紹介しておこう。

●Berkeley MPEG Tools（mpeg_encode）（画面3）

- **http://bmrc.berkeley.edu/frame/research/mpeg/**

米国カリフォルニア州立大学バークレイ校で進められている、MPEG関連の研究の成果として、MPEG-1のコーデックおよびMPEG-2のデコーダ（プレーヤ）などのツールが公開されている。もともとは、SunやHPなどのUNIXワークステーション用に作成されたコードだが、Linuxでもコンパイルして実行できる。

規格通りのMPEGファイルを作成できるが、画像部分と音声部分を別々にエンコードしてから、それらを合成して1つのファイルにする必要があるなど、使い方がかなり面倒である。また、画像のエンコーディングについても、AVI形式などでキャプチャしたデータを、いったんバラバラの静止画像ファイルに分解してからエンコードを行わなければならないので、リアルタイムのエンコーディングは不可能で、処理の負荷もかなり大きいという欠点がある。

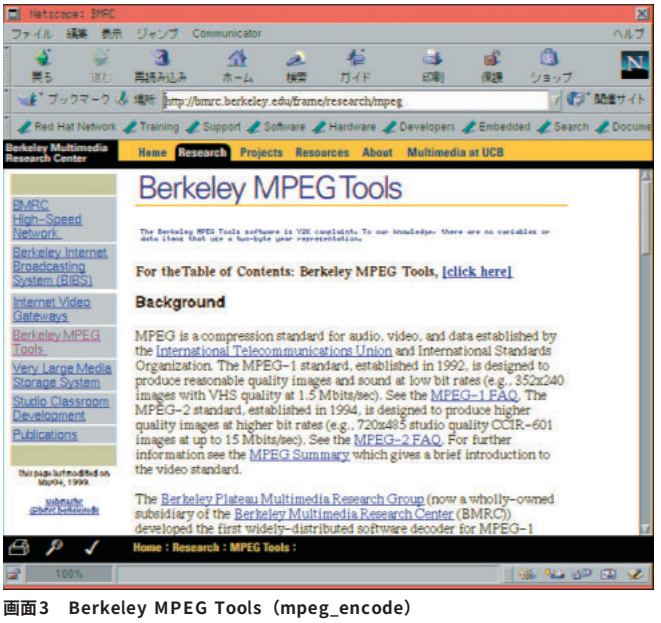
●MJPEGTOOLS（画面4）

- **http://mjpeg.sourceforge.net/**

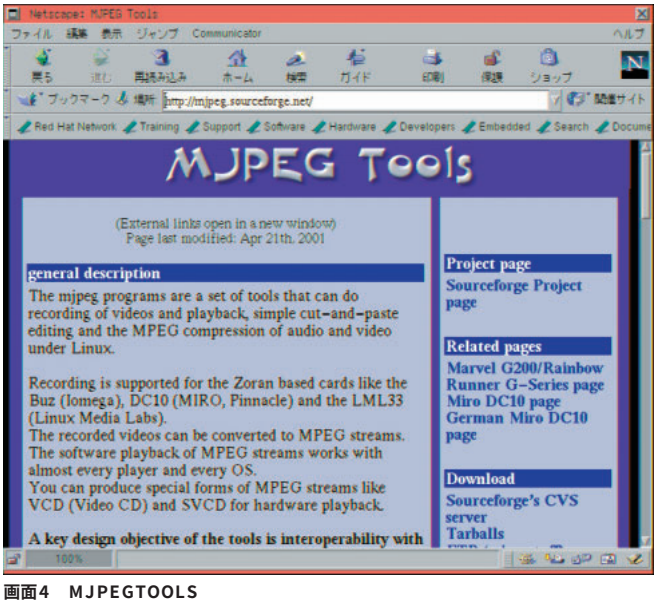
Linux上で、MPEG形式やMotion JPEG形式のキャプチャ、再生、編集を行えるツールを作成するプロジェクト。現在、一部のカードに対応するキャプチャツールが公開されているが、bttvとの互換性はないようなので、本記事の目的には利用できない。ただし、このプロジェクトは現在でも活発に活動しているので、今後の展開に注目したい。

●mp1e

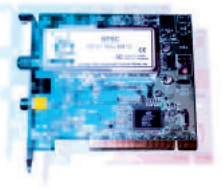
Michael H. Schimek氏が作成した、MPEG-1形式のリアルタイムエンコーダ。CPUパワーをそれほど必要とせずに、リアルタイムに音声と画像のエンコードができる。バージョン1.7までは、MPEGの規格の一部しか実装していなかったた



画面3　Berkeley MPEG Tools（mpeg_encode）



画面4　MJPEGTOOLS



め、作成されるMPEGファイルを再生できない場合があったり、ファイルサイズが大きくなったりすることがあったが、バージョン1.8.1ではこれらの欠点は解消されている。Video 4 Linuxに対応したキャプチャカードを利用できる。

ただし、作者がこのツールを公開していたWebページは現在存在していないため、開発が継続されているかどうかは不明である。また、ソースファイルからのコンパイル作業はかなり難しく、ドキュメントも少ないのが難点である。

現状では、リアルタイムでMPEG-1形式のキャプチャができるツールは、このmp1e以外には見あたらない。そこで、本連載でもこのツールを利用することにしているが、コンパイルやインストールには若干のコツがいる。また、うまくインストールできても、キャプチャした音声がとぎれたり、長時間のキャプチャを行うとマシンがハングアップするなど、不安定な部分も多い。

これらを解決する方法については、あらためて解説することにして、今回はmp1eのインストールのコツについてだけ触れる。

mp1e-1.7のインストール

mp1eの旧バージョン「1.7.1」は、前述したように、MPEGの仕様の一部しか実装していない。しかし、最新バージョンの「1.8.1」は、Video 4 Linux 2（Video 4 Linuxの拡張版）がインストールされた環境でないと、動作しない場合がある（Video 4 Linuxでも動作するのだが、bttvのドライバのバージョンによっては動作しないことがある）。このため、利用しているLinuxのディストリビューションや環境によって、旧バージョンの1.7でないと都合の悪い場合もある。

残念ながら、現在mp1eのバージョン1.7.1を公式にダウンロードできる場所はない。mp1eは、METALAB（旧称Sun SITE）のFTPサイト（http://metalab.unc.edu/）、あるいはそのミラーサイトからダウンロードできるのだが、現在、mp1eのソースファイルはすべてバージョン1.8.1のものに置き換えられている。バージョン1.7.1のmp1eを使いたい場合には、Web上の検索サイトなどで探すしかないだろう。

●gccコンパイラ用パッチの適用

mp1e-1.7.1（mp1e-1.7.1.tar.gz）は、そのままでは現在のLinuxディストリビューションに付属するgccではコンパイルできない。このため、gccでコンパイルできるようにするためのパッチ（mp1e-1.7.1-fefe.diff）を、http://www.fefe.de/mp1e/から入手する必要がある。

mp1eのソースファイルとパッチファイルを入手したら、次のような手順でソースファイルにパッチを適用し、コンパイルを実行する。

```
$ ls
mp1e-1.7.1.tar.gz    mp1e-1.7.1-fefe.diff
$ tar xzf mp1e-1.7.1.tar.gz
$ cd mp1e-1.7.1
$ patch -p1 < ../mp1e-1.7.1-fefe.diff
$ ./configure
$ make
```

コンパイルが成功すれば、カレントディレクトリに「mp1e」という実行ファイルが作成されるので、

```
$ ./mp1e > test.mpeg
```

を実行してみる。ビデオキャプチャカードのドライバ類の設定が問題なければ、キャプチャが実行されるはずだ。キャプチャを終了するには、Ctrl-XあるいはCtrl-Cを入力すればよい。

mp1eコマンドの詳しい使い方については、改めて紹介するが、とりあえずは、

```
$ ./mp1e --help
```

を実行すると、オプションの一覧が表示されるので、それを参考にしてほしい。

mp1e-1.8.1のインストール

最新バージョンの「1.8.1」は、http://www.ibiblio.org/pub/Linux/apps/sound/convert/などからダウンロードできる。1.8.1でも、基本的なコンパイルの手順は旧バージョンと変わらない（gcc用のパッチを当てる作業は必要ない）。しかし、現行のいくつかのLinuxディストリビューション（カーネル2.2系、カーネル2.4系の両方）で試してみたところ、そのままではうまくコンパイルできない場合が多かった。次のような手順で、「libaudiofile」ライブラリのバージョンアップを行い、場合によっては「Video 4 Linux 2」を行えばコンパイルと実行ができるようになるはずだ。

●libaudiofileのバージョンアップ

mp1e-1.8.1のソースファイル（mp1e-1.8.1.tar.bz2）をダウンロードして展開して、「./configure」、「make」を実行したとき、「afSetVirtualSampleFormat関数が見つからない」というような意味のエラーメッセージが表示される場合は、libaudiofileのバージョンアップをする必要がある。

使用中のディストリビューションによって、このライブラリのバージョンアップ方法は異なるが、Red Hat LinuxやDebianなど、パッケージによるバージョンアップが可能なディストリビューションの場合なら、バージョン0.2.1以上のlibaudiofileのパッケージを入手して、インストールすればよい。

パッケージが見あたらない場合でも、tarファイルを入手すればコンパイルとインストールは問題なくできるはずだ。

libaudiofileのバージョンを上げれば、mp1e-1.8.1のコンパイルとインストールは、何の問題もなくできるはずである。

●Video 4 Linux 2のインストール

mp1e-1.8.1のコンパイルが成功しても、いざ実行してみると、「V4Lのドライバが機能していない。V4L2を使え」という意味のエラーメッセージが表示されて、キャプチャが不可能な場合がある。このような場合は、Video 4 Linux（V4L）のドライバをバージョンアップするか、あるいはVideo 4 Linux 2（V4L2）をインストールしなければならない。

V4L（およびV4L2）のドライバは、カーネルモジュールになっているため、最低でも、カーネルの再構築ができる状態になっていなければならない。どちらの場合も、バージョンアップと再構築の方法はほとんど同じなので、ここでは、V4L2にバージョンアップする方法だけを、ごく簡単に説明する。自分でカーネルモジュールのコンパイルをする自信のある人は挑戦してみてほしい。より詳しい手順については、改めて解説することにしたい。

まず、V4L2のソースファイルを入手する（http://bttv-v4l2.sourceforge.net/）。

V4L2のドライバは、V4L2のビデオデバイスドライバ部分と、キャプチャカードドライバ（bttv2など）の2つの部分に大きく分かれている。ソースファイルも、この部分ごとに別々に配布されている。

ビデオデバイスドライバ部分は「videodev20010302.tgz」というような名前のファイルで配布されている。このファイルを入手したら、カーネルのソースコードのディレクトリ（通常は/usr/src/linux）上で展開する（古いソースコードが上書きされるので、必要ならコピーをとっておく）。あとは、カーネルモジュールの再コンパイルを行えばよい。

```
# cd /usr/src/linux
# make oldconfig
# make depend
# make modules
# make modules_install
```

次に、キャプチャカードドライバ部分のコンパイルを行う。bttvドライバのV4L2版は、「driver-20000804.tgz」というようなファイル名で配布されているので、これを入手して、適当な作業用ディレクトリ上で展開する。

展開したディレクトリ（driverという名前のサブディレクトリ）の中のMakefileを適宜編集して、「make」コマンドを実行すれば、ドライバのコンパイルが完了するはずだ。ここで、suコマンドでrootユーザーになってから、「make install」を実行すれば、作成されたモジュールが/lib/modules/以下の適切なディレクトリにコピーされるはずだが、見当違いのディレクトリにコピーされることもあるので、そのような場合は、

```
# cp *.o /lib/modules/[使用中のカーネルのバージョン]/misc
```

のように、手動でコピーする必要がある。

ドライバのコンパイルとインストールが終わったら、/etc/modules.confを編集して、V4L関連の設定部分について、「bttv」という文字列をすべて「bttv2」に書き換える。あとは、

```
# depmod -a
# rmmod bttv.o
# modprobe bttv2
```

を実行すれば、V4L2が利用できるようになるはずだ。

※ここで紹介したリスト、mp1e関連のファイルは、Linux magazineのホームページ、
http://www.ascii.co.jp/linuxmag/
からダウンロードできます。

箱の中のペンギンたち

文：みわよしこ
Text : Yoshiko Miwa miwachan@pp.iij4u.or.jp

第6回　技術という無形物の価値

組み込みシステムの開発とは、技術を製品に作り込むことであると言っても過言ではありません。組み込みシステム開発は、これまでいくつもの開発者の事例で紹介してきたように、技術力とアイデアの発想力勝負です。

しかしながら、技術やアイデアは製品に実装されて初めて目に見える形になるものです。言い方を変えれば、製品にまだ実装されていない技術やアイデアは目に見えないということでもあります。

今回は、組み込みシステムにとって極めて重要な「技術」「アイデア」の価値について考えてみます。

技術やアイデアの価値とは？

まず、PCという目に見えるハードウェアの価格から、「価値」が何によって決定されるかを見てみましょう。「価値」と「価格」は同義ではありませんが、価格は価値と概ね比例関係にあります。

PCの場合、価格を決定する大きな要因はもちろん生産コストです。生産コストには、部品の購入や保管に関する費用、生産設備の導入・維持に関する費用、そして人件費が含まれます。

価格を決定するもうひとつの重要な要因は出荷時期です。最新CPUが発売されれば、1世代・2世代前のCPUの価格が下がり、それを利用したPCの価格も下がることは「常識」でしょう。

もちろん価格を決定する要因は、ほかにも多数あります。たとえばWindows 95が登場し、PCが一般家庭に急激に普及した時期、PCの価格は高めに設定されていました。それでも需要があったからです。逆に現在のように不況が長期化し、一般家庭の経済力にそれほどの余裕がない時期には、PCの価格は安く設定される傾向にあります。もちろん販売価格を安く設定するためには、コストを抑える必要があります。各メーカーは部品を安価に調達し、人件費を抑える努力を最大限に払っています。

デスクトップPCは既存のパーツを適切に組み合わせ、組み立てるだけで完成してしまいます。ここに技術が入り込む要素はほとんどありません。技術はCPUやボード類をはじめとする、各パーツの開発にもっぱら関わってきます。

ノートPCは多くの場合、外形の制約に合わせてボードの設計を行います。適切なチップを既存のものから選択し、それらをボード上に適切に配置し、さまざまな制約をかわしながらボード上に配線を作成します。また、動作中に発生する熱が特定箇所に集中することがないようにボードを設計するには高度な技術が必要です。

ノートPCがデスクトップPCより若干割高な理由のひとつは技術料であると言ってもおかしくありません。

技術やアイデアは「知的財産」になる

技術やアイデアの発明者は、それを特許や実用新案にすることによって自分のオリジナルなものとして権利を主張し、保護を受けることができます。

人間の知的創造物には発明、考案、意匠（デザイン）、著作物など多様なものがあります。それぞれ特許法、実用新案法、意匠法、著作権法など数多くの法律の適用対象となります。これらを総称して「知的財産」、その権利を「知的財産権」と呼びます。

「知的財産」は「無形財産」のひとつです。ちなみに物や不動産などは「有形財産」です。無形財産についても有形財産と同様に、所有などの権利が認められています。つまり、技術やアイデアを生み出し、発明や考案を行うということは、「無形財産」を生み出すことなのです。そして技術やアイデアは、知的財産権に関する制度で保護されます。

実情の後追いを続ける特許制度

前述のように、技術者は特許制度を利用すれば、自分の技術やアイデアを権利化し、保護してもらうことができます。しかし、特許制度が現在の実情に必ずしも適していないという問題があります。

まずは特許法の条文を見てみましょう。

特許法第一章第二条

1　この法律で「発明」とは、自然法則を利用した技術的思想の創作のうち高度のものをいう。

2　この法律で「特許発明」とは、特許を受けている発明をいう。

3　この法律で発明について「実施」とは、次に掲げる行為をいう。

一　物の発明にあつては、その物を生産し、使用し、譲渡し、貸し渡し、若しくは輸入し、又はその譲渡若しくは貸渡しの申出（譲渡又は貸渡しのための展示を含む。以下同じ。）をする行為

二　方法の発明にあつては、その方法を使用する行為

三　物を生産する方法の発明にあつては、前号に掲げるもののほか、その方法により生産した物を使用し、譲渡し、貸し渡し、若しくは輸入し、又はその譲渡若しくは貸渡しの申出をする行為

「物」という言葉が頻出することに注意してください。

「自然法則を利用した技術的思想の創作のうち高度のもの」の価値を評価することが容易でないため、かつて特許はほとんど、具体的な物に限定して出願され、審査されていました。

筆者は以前、ある企業でシミュレーションシステムの研究開発に従事していましたが、入社した当時（'90年）、ソフト

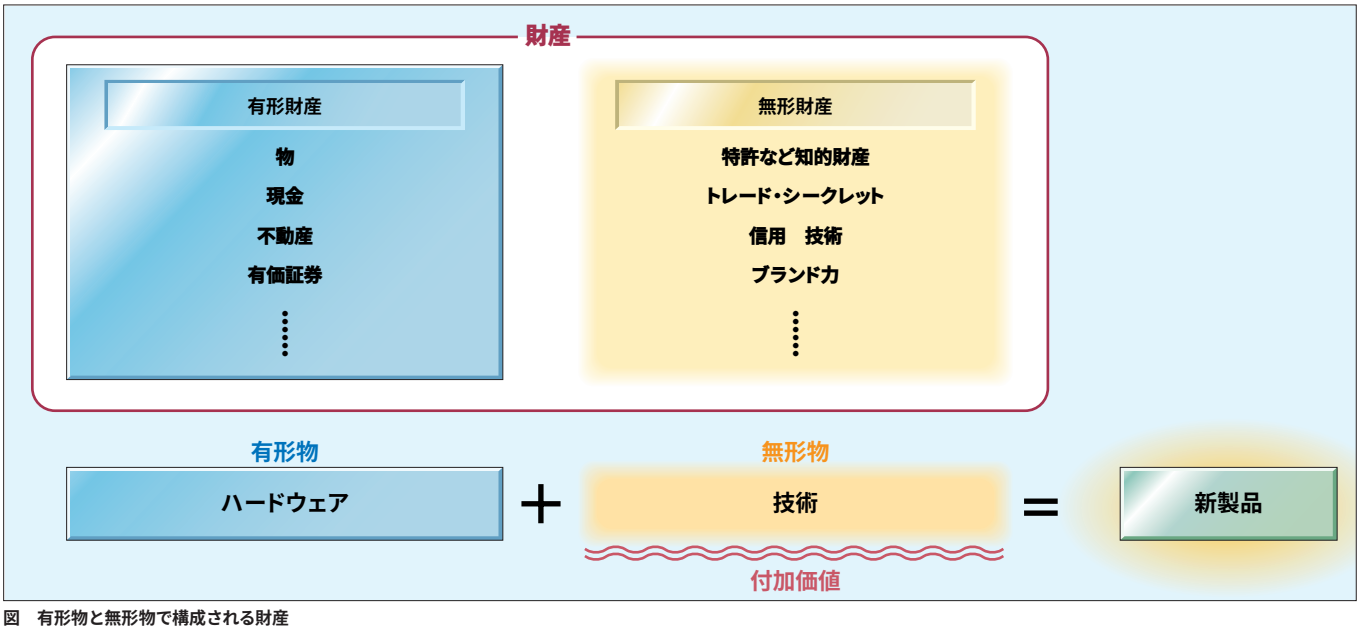


図　有形物と無形物で構成される財産
知的財産、信用、技術、ブランド力など、形のないものも法的に認められた財産。技術者は「既存のハードウェア」という有形物に技術という無形物で付加価値を高め、新製品を生み出している。

ウェアのアルゴリズムは特許として成立させることができませんでした。このため、アルゴリズムを特許化したい場合には、バスなど、何か具体的な物と無理やりに関連付けるしかありませんでした。ソフトウェア特許が認められ、アルゴリズムをそのまま特許化することが可能になったのは数年後のことでした。

最近話題となっている「ビジネスモデル特許」も同様の経緯をたどって、特許として成立させることが可能になりました。ビジネスモデルを知的財産として権利化するという動きが活発になり、それを特許法が数年遅れで後追いしたわけです。

こういった歴史を見るにつけ、物と直接結びつかない技術やアイデアを守るために特許制度が完全であるとは、とても考えられません。「技術立国」を誇るはずの日本で、国として施行している特許制度が技術やアイデアを守るために十分でないというのは悲しむべきことです。

物と結びつかない技術の価値は評価しにくい

物でなく、技術やアイデアそのものを評価するという雰囲気は、もともと日本には濃くありませんでした。先に述べたように、特許制度が「物」に偏る傾向を持っているということは、そのことと無縁でないでしょう。

ともあれ、このような現状でも技術者は自分の技術やアイデアを活かし、収益の上がるビジネスをしなくてはなりません。技術やアイデアそのものの価値を正当に評価し、価格に反映する仕組みがない以上、生産物の価格や人件費の形で、技術やアイデアの対価を得るということになります。

今回は、ネットエージェントの製品「パケットブラックホール」で、その一例を見てみましょう。

ネットワークを監視する「パケットブラックホール」

サーバへの侵入は日常茶飯事になっていますが、通常、侵入者は「足跡」を残さないように万全の注意を払います。ネットエージェントの社長である杉浦隆幸氏は、なんとかして侵入者の足跡を残すことはできないかと考えました。それを可能にしたのが「パケットブラックホール」です。

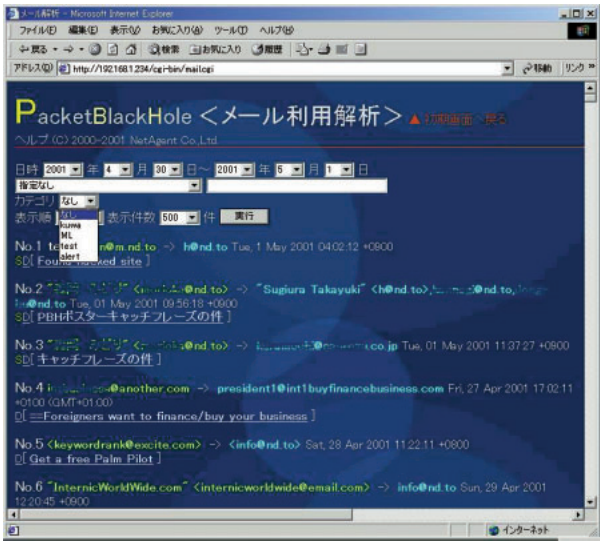
パケットブラックホールには、タワータイプとラックマウントタイプの2種類がありますが、どちらも一見、通常のPCと何ら変わるところはありません。しかしこれをインターネットとLANの間に設置すると、通過するパケットをすべてハードディスクに記録します。

パケットの記録自体は、たとえばLinuxでは「tcpdump」コマンドで容易に行えますが、パケットブラックホールの特徴は、「外部からはその存在がわからない」ということです。この機能は杉浦氏自身が開発しました。

なお、パケットブラックホールはLinuxを組み込んでいますが、ユーザーはまったくLinuxを意識することはありません。操作はすべてブラウザから行うことができ、インターネットでWeb検索エンジンを使ったことのある人なら操作できるようになっています。

操作メニューは非常にわかりやすく作られているため、セキュリティに関する知識がまったくなくとも、必要な操作を行い、情報を取り出すことが可能です。ちなみに海外でのビジネス展開も考慮し、操作メニューは日本語、英語、フラ

写真1　「パケットブラックホール」製品外形



画面1　メールの送受信のチェック画面　要注意メールアドレスからの送受信をまとめて表示することもできる。

ンス語、ドイツ語、中国語（2種類）に対応しています。

パケットブラックホールを生み出したネットワーク技術

杉浦氏は学生時代、アルバイトでプロバイダの立ち上げに関わり、立ち上がったあとの管理を全部任せられるという経験を通じて、「いやでもネットワークに詳しくなった」ということです。

プロバイダを管理していると、セキュリティの問題を避けて通ることはできません。そこで杉浦氏は「セキュリティの情報を集めるサイトを作りたい」と考えました。それも一般論ではなく、セキュリティに関する企業の個別事例を集めたいと考えたのです。そのために杉浦氏は、個人事業所として「ネットエージェント」を立ち上げ、サーバ構築やシステム構築の案件をこなしながら情報を収集し、技術を蓄積し、セキュリティに関しては世界トップレベルの技術を誇る存在になってゆきました。

しかし、サーバ構築やシステム構築にはサービス業の要素が強く、具体的な物を作って販売するのに比べると価値を低く見積もられがちです。また、法人でないことで信用に欠けると評価され、案件が取れないこともあったそうです。そういったことから、杉浦氏は自分の技術を具体的な製品に実装して販売し、また個人事業所を法人化することを考えるようになりました。

ネットエージェントが株式会社になったのは昨年6月、パケットブラックホールの最初のモデルが発売されたのは2001年2月のことですが、これだけの短期間で開発できたのは、それまでの数年間、杉浦氏が蓄積した技術やノウハウがあって、はじめて可能になったと言えるでしょう。

なお、杉浦氏はパケットブラックホールのコアとなる部分の技術を特許として出願しています。自分の技術を自分で守るのは、これからの技術者にとっては「常識」と言えるでしょう。

パケットブラックホールの技術の価値

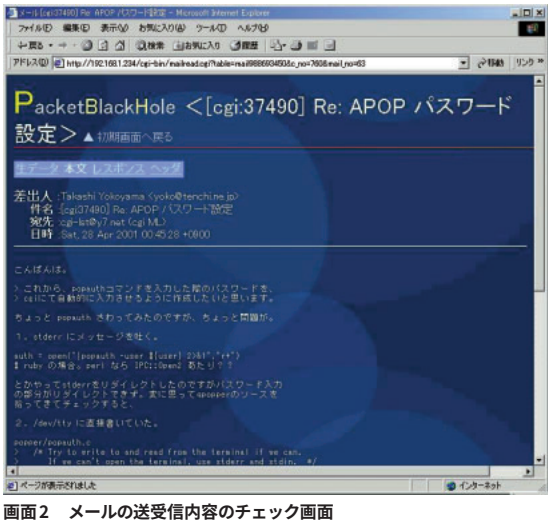
前述したとおり、パケットブラックホールはハードウェア的にはごくごく普通のPCです。信頼性の高い高級なパーツを利用したとしても、せいぜい30万円程度のものでしょう。

一方、機能面、代替製品がないという面から見ると、パケットブラックホールは1台300万円でも十分に需要を見込めるマシンです。もし1台300万円で売ることができれば、技術の価格が270万円ということになります。

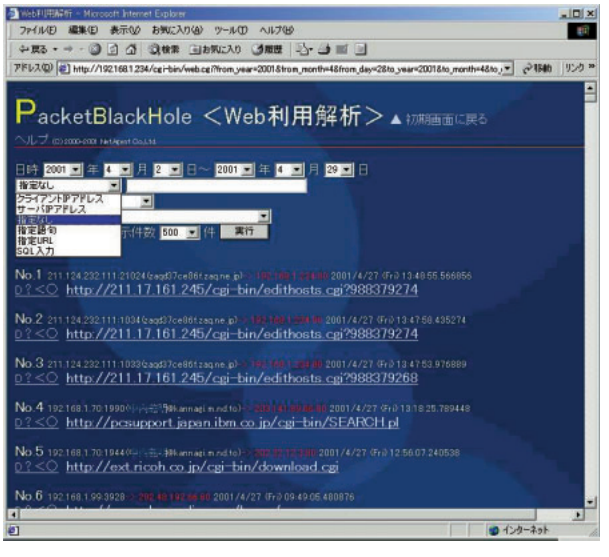
この5月25日、初期モデルから大幅な機能強化を行った「パケットブラックホール2」が発売されました。実際のパケットブラックホール2の売価はリセラーによって異なりますが、100万円を少し超える程度です。筆者からみても妥当な価格だと感じられますが、技術によって70万円以上の付加価値が産み出されるわけです。

一人の技術者が、技術という無形物の価値を有形の製品の価格に置き換えたという面から、非常に興味深い事例であり、製品であると筆者は思います。

技術力の低下がしばしば指摘される現在の日本ですが、組み込みLinux製品を中心に、このような事例がもっと増えれば、若い技術者のモチベーションも上がってゆくというものではないでしょうか。



画面2　メールの送受信内容のチェック画面
要注意メールアドレスから送受信したメールをチェックしている画面。



画面3　Web利用解析画面
社員が勤務時間中に掲示板やアダルトサイトを利用しているかどうかのチェックもできる。このため、パケットブラックホールは外部からの侵入チェックより、社員のメールやWebの利用状況のチェックに利用されているケースが多いという。

ネットエージェント代表取締役杉浦隆幸氏に聞く

Interview

26歳の若さで、世界的に見ても最高レベルのセキュリティサービスを提供するネットエージェントを設立し、ユニークな組み込みLinux製品「パケットブラックホール」を開発した杉浦隆幸氏。どのようにして現在の杉浦氏が形作られたのかは、大いに興味深いところです。今回は、杉浦氏のライフストーリーや会社設立に至る経緯を中心にお話を伺いました。

――はじめまして。ネットエージェントのホームページ（http://www.netagent.nd.to/）で拝見していたこれまでのご経歴から30代の方かな？　と勝手に想像していたのですが、ずいぶんお若いですね。

杉浦：はい、現在26歳です。

――その年齢でこれだけの技術を蓄積してこられたというのは凄いですね。

杉浦：コンピュータは12歳くらいのころから触ってきましたので。

――コンピュータに接したきっかけは、どういったことだったのでしょうか？

杉浦：親がMSX II*を買ってくれたんです。

――MSXというと、ゲームのために開発されたマシンという記憶があるんですが。

杉浦：そうです。ただ当時、ゲームをやるためには自分でBASICやマシン語のプログラムを打ち込まくてはならなかったんです。

――当時のパソコン雑誌には、ゲームのプログラムリストが必ず掲載されていましたね。それをただ打ち込んでいるだけでも、基本的なプログラムの作り方はかなり習得できますよね。

杉浦：ええ。自分でプログラムを書けないと面白いことは何もできない時代でしたから、すぐにプログラムの作り方を覚えましたね。

写真　ネットエージェント代表取締役　杉浦隆幸氏（すぎうらたかゆき）
1975年生まれ。東京理科大学中退。地方プロバイダでプロバイダ事業の立ち上げからシステム設計・運用・管理・プロモーションに至るすべての業務を手がける。1998年、個人事業所ネットエージェントを設立。セキュリティ事業のほか、Linuxディストリビューション開発、多くのWebサイトのシステム開発に関わる。2000年6月、ネットエージェント株式会社を設立し、現在は代表取締役。

――当時の典型的なパソコン少年だったわけですね。ネットワーク環境も初期のころから利用していたのですか？

杉浦：いえ、本格的に利用するようになったのは大学生の時です。'96年ごろ、立ち上げ前のプロバイダでアルバイトしまして、立ち上がったあとは業務を全部任せられたんです。それで、いやでもネットワークに詳しくなりました。

――OSはLinuxですか？

杉浦：Solaris 2.5.1です（笑）。立ち上げが終わって、運用管理もほとんど自動化できるようにしたら、ふだんはやることが何もないじゃないですか。それで、合間に本を読んだり、いろいろ実験してみたりして、技術や知識を蓄えていきました。そのうちに仕事が面白くなってきて、大学は中退したんですが。

――大学では何を専攻されていたんですか？

杉浦：材料工学です。

――情報工学関連ではないところが意外ですね。

杉浦：コンピュータ関連の学科なんて、行ってもしょうがないじゃないですか。

――確かにそうですね。大学の情報関連学科には、わざわざ大学でやらなくても……という面も多々見受けられますね（笑）。ところで、プロバイダのバイトからセキュリティに特化した株式会社の設立までは、どういう道をたどられたのでしょう？

杉浦：プロバイダの運営をやっているうちに、セキュリティの問題の重要さに気づいたんですが、世の中にセキュリティの情報がないんです。一般論的なことだったらあるんですけど、個別の具体的な事例に関する情報はほとんどなかったんですね。それで、セキュリティ情報を集めるサイトを作りたいなと思いました。そのためにまず、個人で「ネットエージェント」を始めたんです。

――最初は個人事業所だったわけですね？

杉浦：そうです。サイトを作って事例を集めて、それを本や雑誌に書いたり、サーバ構築やシステム構築をして生活していました。そのうちに収入も増えてきたので、何人か取り込んで組織的にやろうと思うようになったんです。

──それで2000年6月に株式会社を設立されたんですね。

杉浦：ええ。サーバ構築やシステム構築はサービス業なので対価を安くされることが多くて悔しかったのと、個人事業所だと信用の面から案件が取りにくいということがあって……。

──株式会社を設立するには最低1000万円の資本金が必要ですが、それはどうやって調達されましたか？

杉浦：ずっとサーバ構築やシステム構築の仕事をやっていて、幸いに良い評価をいただいていて、自然にできたコネクションがありました。「投資したい」と言ってくださる方が13人か14人いらして、それで出資いただき、株式会社を作ることができたんです。

──やはり、一朝一夕にできることではありませんね。

杉浦：それはそうでしょう（笑）。

──起業する前と後で、変化はありましたか。

杉浦：そうですね。株式会社を設立することでコネクションが広がり、強くなりました。

──パケットブラックホールも、製造はナカガワメタルさんですね。そういったコネクションでしょうか？

杉浦：この製品は、開発はネットエージェント、組み立て・プロモーション・営業はナカガワメタルさんという分業体制でやっています。ネットエージェントは会社として非常に若いので、ナカガワメタルさんのような老舗のコネクションを利用しないと製品の販売が難しいということもあって……。

──現在の社員数は何人ですか？

杉浦：自分のほかに2人です。パケットブラックホールのユーザーインターフェイスのCGI部分は、5カ国語（日、英、仏、独、中、台湾）対応ということで外注を利用していますが。

──その人数で営業や販売までやっていたら、技術開発はできませんね。会社を設立されて、そのほかに変化はありましたか？

杉浦：自分自身が非常に忙しくなりました（笑）。

──「嬉しい悲鳴」という感じですね（笑）。今後の展開はどのようにお考えですか？

杉浦：開発したいセキュリティ関連製品が、ほかに3つくらいあるんです。それを形にして、販売していきたいです。それと関連して、セキュリティの案件を取るという形で広げていきたいと考えています。

──無理がない発展方向という感じを受けます。

杉浦：ええ。それから海外進出も考えています。パケットブラックホールのユーザーインターフェイスは、そのために最初から多国語対応しているんです。といってもCGIを5カ国語と一地域に対応させているだけなんですけど……。

──どの国で利用しても、ブラウザで対応できますね。

杉浦：そうなんです。もともとは「Webで検索エンジンを利用したことがある人なら利用できるように」ということで、ブラウザから簡単に操作できるようにしたんですが、結果としてそのおかげで多言語対応が楽になりました。

──今後の展開が楽しみですね。海外での発表予定は決まっていますか？

杉浦：今年11月、アメリカのCOMDEXに出展する予定です。アメリカで評価されることで、日本でもさらに販路が広がる……という方向性も狙っています。

──最後に、これからのLinux業界に望むことを聞かせてください。

杉浦：着実にビジネスを広げてほしいと思います。そして各社が、「オープンソースでフリーなんだからタダだ」と思わせないビジネス、高いビジネスをできるようになってほしいと思いますね。

──Linux自体はフリーなんだけど、技術を売って高い付加価値を付けるということですね。

杉浦：そうです。日本は無形なものに対しての意識が低いので、そのあたりをLinuxビジネスでうまく啓蒙できるといいな、と思いますね。

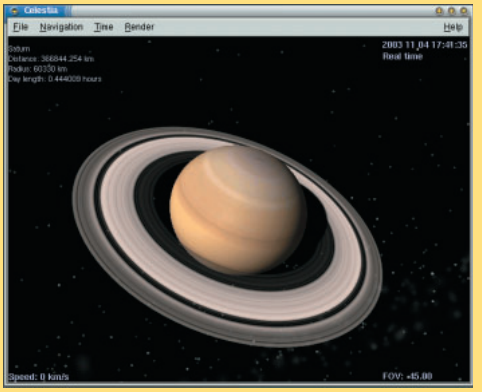
──私自身も技術者でしたので、そのあたりは実感として感じます。今日はありがとうございました。

* 1983年にマイクロソフトとアスキーが提唱した8ビットホームコンピュータ規格。

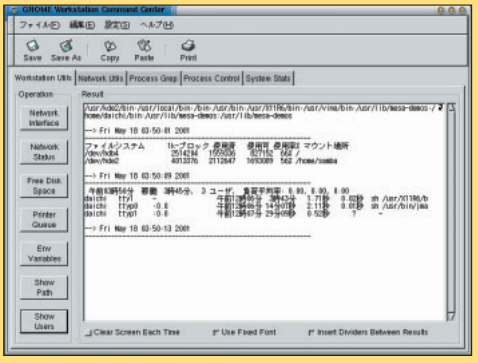
写真2　ネットエージェント入り口近くの風景
ずらりと揃った「ハッカージャパン」が目を引く。

Free Application Showcase

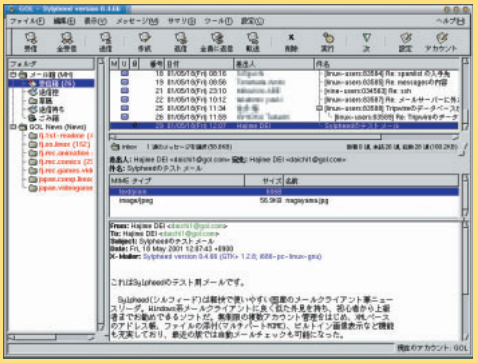
文：出井 一
Text : Hajime Dei



Celestia　P.118



GNOME Workstation Command Center　P.120



Sylpheed　P.115

安定して使いやすいメールクライアント
Sylpheed 115

太陽系の惑星や銀河の星々を訪問しよう
Celestia 118

各種コマンドの実行やプロセス制御をGUIで行う
GNOME Workstation Command Center 120

KDE2用の高速イメージビューア
KuickShow 122

多人数対戦も可能な海戦シミュレーションゲーム
Batalla Naval 124

溜め撃ちが楽しい横スクロールシューティングゲーム
Geki3 126

パネルを回転させてボールを移動するパズルゲーム
Groundhog 127

たくさんのかわいいペンギンが画面を走り回る
XPenguins 128

複数の配列を切り替えられる仮想キーボード
xvkbd 129

※紹介したソフトは、すべて付録CD-ROMに収録されています。

安定して使いやすいメールクライアント

Sylpheed

バージョン：0.4.66　　ライセンス：GPL

http://sylpheed.good-day.net/
http://y-imai.good-day.net/sylpheed/（RPM）

Sylpheed（シルフィード）は軽快で使いやすい国産のメールクライアント兼ニュースリーダソフトだ。Windows系メールクライアントに似た外見を持ち、初心者から上級者までお勧めできるソフトだ。無制限の複数アカウント管理をはじめ、XMLベースのアドレス帳、ファイルの添付（マルチパートMIME）、ビルトイン画像表示など機能も充実しており、最近の版では自動メールチェックも可能になった。

●ビルドとインストール

Sylpheedは、tarボールとRPMパッケージが配布されている。最近のディストリビューションではSylpheedがインストール済みの場合も多いので、「rpm -q sylpheed」としてバージョンを確認してみよう。

RPMはソース／バイナリパッケージがそれぞれ用意されている。Red Hat 6系列のディストリビューションなら、バイナリパッケージをそのまま利用できるはずだ。「rpm -Uvh sylpheed-0.4.66-1.i386.rpm」としてインストールしよう。

ソースパッケージからリビルドするには、「rpm --rebuild sylpheed-0.4.66-1.src.rpm」とする。あとは、/usr/src/redhat/RPMS/i386に作成されたバイナリパッケージをインストールすればいい。

一方、tarボールを利用する場合は、「./configure」→「make」としてビルドし、「su」でスーパーユーザーになってから「make install」でインストールする一般的な手順だ。

●起動から初期設定まで

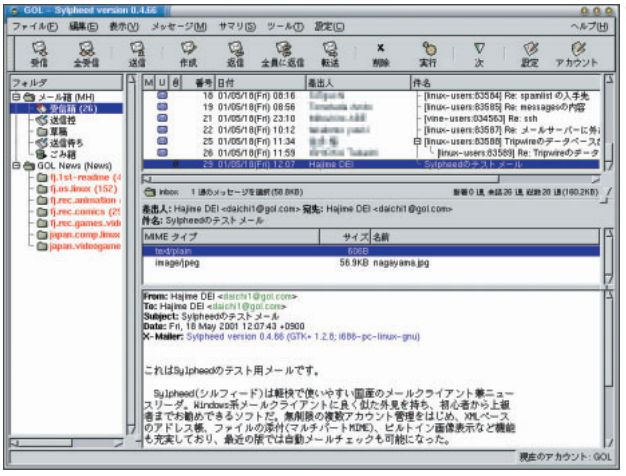
ktermなどから「sylpheed&」として起動するか、GNOMEメニューの[プログラム]－[インターネット]－[Sylpheed]を選択すると、Windows系メールクライアントでおなじみの3ペイン方式のメインウィンドウが開く（**画面1**）。左側にフォルダツリー、右上にサマリビュー、右下にメッセージビューという画面構成だ。

初めてSylpheedを起動した（あるいはバージョン0.4.9以前からバージョンアップした）場合は、最初にメールボックスの位置を指定する必要がある。通常は初期設定（Mail）をそのまま選択すればいい。MH形式のメールボックスがすでに存在する場合は、そのディレクトリ名を指定することで、これまでに蓄えてきたメールをそのままSylpheedで扱える。

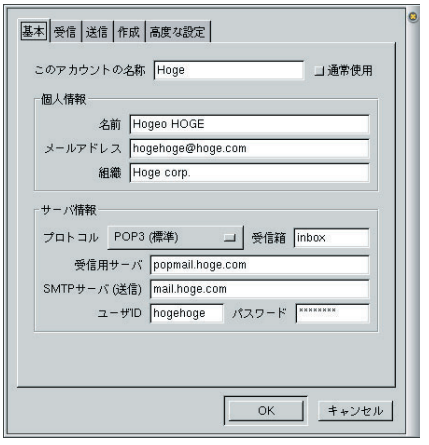
続いて、アカウントの登録を行おう。ツールバーの[アカウント]ボタンを押し、設定ダイアログの[基本]ページでアカウントの名称、名前、メールアドレス、送受信するサーバ名、プロトコルなどを設定する（**画面2**）。他の設定は後で行えばいい。

登録したアカウントは「アカウントの編集」ダイアログに表示される。このダイアログでは、新規アカウントの追加、既存のアカウントの編集・削除などが可能だ。各ユーザーが、複数のアカウントを無制限に利用できるようになっている。

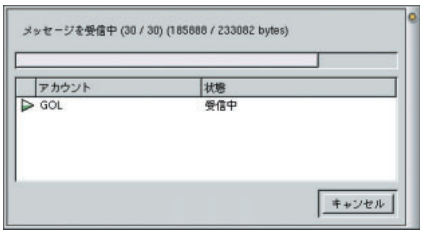
なお、現在のアカウント名はメインウィンドウの右下に表示されており、この部分をクリックして別のアカウン



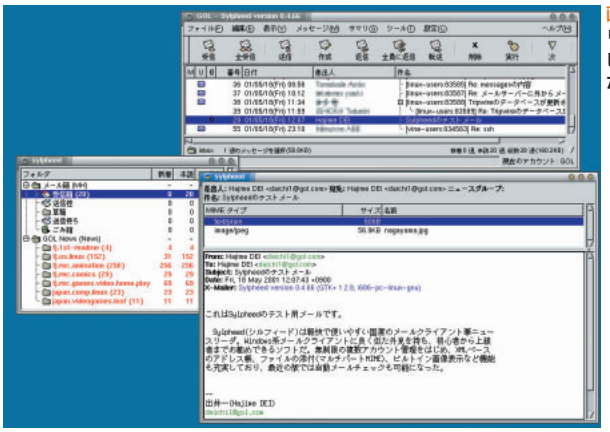
画面1　Outlook Expressなどと同じ3ペイン構成のウィンドウ。



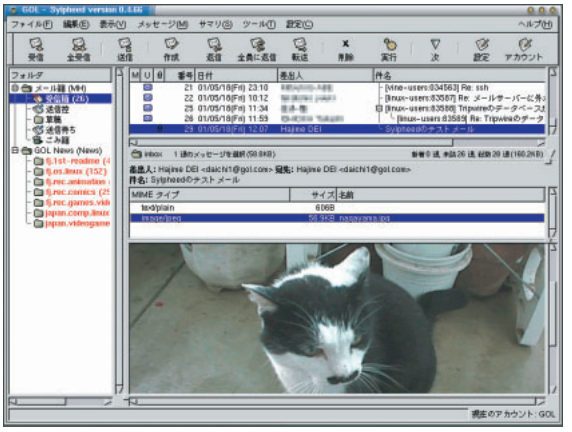
画面2　アカウントの設定はこのダイアログで。複数アカウント対応だ。



画面3　メールサーバからメールを受信中。複数サーバの巡回も可能。



画面4　フォルダツリーやメッセージビューを独立させた状態で使う。



画面5　メールに添付された画像を直接メッセージビューで閲覧できる。

トに切り替える。

●メールの受信と閲覧

ツールバーの[受信]ボタンを押すと、現在のアカウントのメールサーバに接続して新着メールを受信する（**画面3**）。また、登録した複数のアカウントの新着メールを一度に受信する[全受信]ボタンも用意されている。ローカルなメールボックス（mbox）のインポートも可能だ。

受信したメールは[受信箱]フォルダに格納される。このフォルダを選択すると、メールのタイトルや差出人などがサマリビューにスレッド表示され、サマリビューで選択したメールの本文がメッセージビューに表示される。なお、フォルダツリーやメッセージビューを独立したウィンドウとして利用することもできる（**画面4**）。

未読メールを閲覧するには、スペースキーを押せばいい。次の未読メールへの切り替え、画面に収まりきらない本文のスクロール、次のフォルダへの切り替えなどをインテリジェントに処理してくれる。キー割り当ては、Mew／WanderlustといったEmacs系メールクライアントと互換性があるので、これらを使った経験のある人なら違和感なく操作できるだろう。

マルチパートMIMEによる添付ファイルを持つメールは、サマリビューにクリップアイコン付きで表示され、メッセージビューの上部にファイル名やMIMEタイプのリストが表示される。テキストや画像の場合は、クリックするだけでメッセージビュー内で閲覧できる（**画面5**）。このほか、ファイルの保存やMIMEタイプに応じたアプリの起動にも対応している。

●フォルダの作成と振り分け

流量の多いメーリングリストに加入しているなら、振り分け機能を利用してメールをフォルダに分類しよう。フォルダツリーの[受信箱]フォルダなどで右クリックし、[新規フォルダの作成]を選択すると、整理用のサブフォルダ（階層化可能）を作成できる。

振り分けるルールの設定は、[設定]－[振り分けの設定]で開くダイアログで行う（**画面6**）。振り分けに利用するヘッダとその内容、移動先のフォルダを登録しよう。2つの条件をAND/ORで組み合わせたり、フォルダに振り分ける代わりに受信しないように設定することも可能だ。

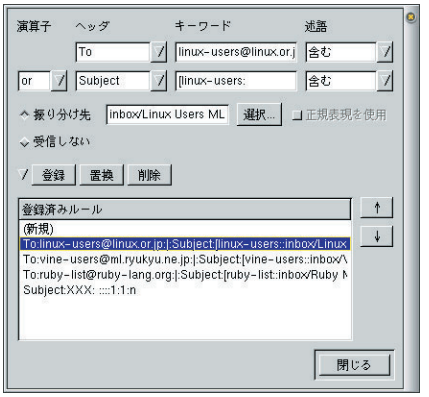
メールを受信すると、設定したルールに基づいて自動的に振り分けが行われる。受信後にルールを追加した場合は、[サマリ]－[メッセージを振り分ける]を選択して、振り分けを手動で指示すればいい。

●メールの作成とアドレス帳

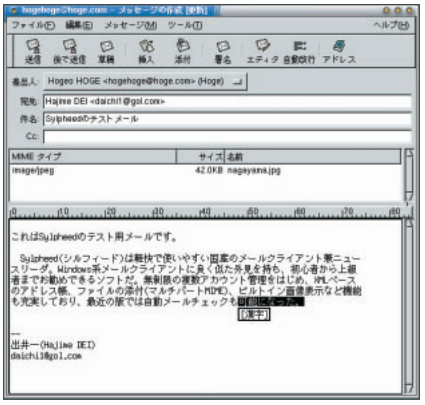
メールを新規作成するには、ツールバーの[作成]ボタンを押す。また、メールの返事を書く[返信]ボタンや、相手とは別の人に転送する[転送]ボタンも用意されている。

いずれにせよ、メッセージ作成ウィンドウが開くので、件名や宛先、本文を入力しよう（**画面7**）。返信などの場合は、件名や宛先はすでに入力済みだ。本文には自動的に署名（シグネチャー）が挿入される。

メールアドレスの入力は、[アドレス]ボタンで表示されるアドレス帳（**画面8**）を利用するのが簡単だ。一覧中からメールアドレス（複数可）を選択して[宛先]や[Cc:]ボタンを押すだけで、作成中のメールの各フィールドにそれ



画面6　受信したメールを自動振り分けするためのルールを設定する。



画面7　メッセージ作成ウィンドウで、送信するメールを書く。

らが挿入される。

なお、アドレス帳の内容はXML形式で記述されており、~/.sylpheed/addressbook.xmlに保存されている。メールアドレスを大量に登録しなければならない場合には、このファイルをテキストエディタやスクリプト言語で直接変更するとよいだろう。

このほか、本文の入力を外部エディターで行ったり、gmcなどのファイルマネージャからテキストファイルをドラッグ&ドロップで本文に挿入することもできる。同様に、添付ファイルについても、リストへのドラッグ&ドロップによる追加が可能だ。

●ニュースリーダ機能を使う

Sylpheedには、ネットニュースを購読するニュースリーダとしての機能も

画面8　XMLベースのアドレス帳。フォルダによる分類や検索も可能だ。

用意されている。メールを読むのと同じ感覚でニュースグループの記事を閲覧可能だ。現在では、ニュースの投稿も可能になっている。

ニュースリーダとして使うには、ニュース専用のアカウントを作成する必要がある。複数のニュース用アカウントを使い分けることも可能だ。[設定]－[新規アカウントの作成]で設定ダイアログを開き、プロトコルとして「ニュース（NNTP）」を選択しよう。あとは、ニュースサーバ名やあなたの名前などを設定すればいい（**画面9**）。登録後は、メインウィンドウのフォルダツリーにニュース用のフォルダが作成される。

続いては、ニュースグループの設定だ。ニュース用のフォルダで右クリックし、メニューから[ニュースグループを購読]を選択して、グループ名（たとえば「fj.os.linux」など）を入力しよう。これを購読したい各グループに対して行えば準備完了だ。

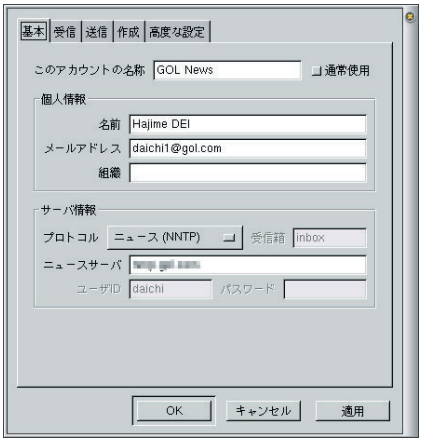
●ニュースの閲覧と投稿

ニュースグループの記事を取得するには、そのグループ名のフォルダをクリックするだけでいい。自動的にニュースサーバに接続して、新しい記事を取得する。

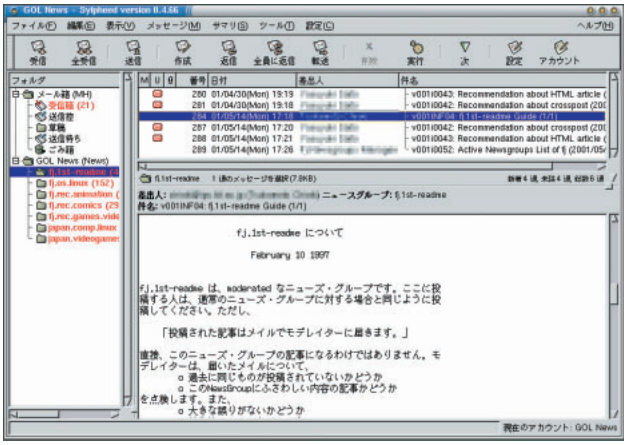
サマリビューには、各記事の件名や投稿者の一覧がスレッド表示され、メッセージビューには選択した記事の本文が表示される（**画面10**）。あとは、メールを読む場合と同じようにして閲覧すればいい。もちろん、スペースキーを押しているだけで未読の記事を読み進められる。

なお、フォルダツリーのペインを拡大すると、ニュースの新着数・未読数・総数が表示される。ニュースグループ名は長くなることが多いので、メニューの[表示]－[フォルダツリーを分離]を選択してフォルダツリーを独立させ、横幅を広げると見やすいだろう。上記のメニューを再度選択すれば、元のペイン表示に戻せる。

ニュースグループに自分の記事を投稿するには、[作成]ボタンや[返信]ボタンを押してメッセージ作成ウィンドウを開く。その後の作成手順はメールの場合とまったく同じだ。ただし、送信先はメールアドレスではなく、ニュースグループ名になる。



画面9　ネットニュース用のアカウントもメールと同様に設定する。



画面10　ネットニュースをメールと同じ感覚で読み書きできる。

太陽系の惑星や銀河の星々を訪問しよう

Celestia

バージョン：1.0.10　　ライセンス：GPL

http://www.shatters.net/celestia/

Celestiaは、宇宙を自由に旅するスペースシミュレータだ。太陽系内の惑星や衛星などが美しいテクスチャ付きで表示され、時間経過の速度を変更して運行のようすを眺められる。さらに、太陽系外まで足を伸ばして、見なれない形に歪んだ星座を眺めたり、惑星を持つ恒星系を訪問したり、銀河系全体を外部から眺めたりすることさえ可能だ。実行には3Dライブラリのなどが必要となる。

●ビルドとインストール

Linuxで動作するCelestiaは、ソース一式やデータ、画像などをすべて含んだフルセットのtarボールのみ配布されている。「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。

なお、このバージョンから、GTK＋とGtkGLArea（http://www.student.oulu.fi/~jlof/gtkglarea/）を利用したメニューを付けられるようになった。この機能を使う場合は、configureの1468行目と1742行目の「==」を「=」に修正後、「./configure --enable-gtk」としてビルドする必要がある。

●起動とデモモード

Celestiaを実行するには、展開先のディレクトリで「celestia&」としなくてはならない。これでは面倒なので、どのディレクトリからでも起動できるスクリプトを用意した（**リスト1**）。2行目の「/usr/src/celestia-1.0.10」は、各自の展開先ディレクトリに合わせて変更してほしい。

起動したらdキーを押して、Celestiaの能力を手軽に試せるデモモードに切り替えよう。地球から始まり（**画面1**）、月や太陽、土星（**画面2**）といった太陽系内の天体を訪問する。さらに、さそり座の赤色巨星アンタレスや銀河系を外部から一望したりした後、太陽系に帰還して終了する。途中でデモを終了するにはEscキーを押せばいい。

なお、ウィンドウサイズは自由に変更できるので、3Dアクセラレータ環境ではウィンドウを大きくして迫力ある画面を満喫しよう。

●マウスとキーボードで宇宙探索

操作にはマウスとキーボード（**表1**）を併用する。クリックで天体の選択、ドラッグで画面のスクロールだ。また、天体を選択した状態では、左上に天体の名前や距離などが表示され、右ボタンのドラッグで天体中心の回転、マウスホイールの回転で天体との距離の変更を行える。

太陽系内の惑星を眺める場合は、画面上で惑星を探すよりも、数字キーで直接選択したほうが早い。1キーが水星、9キーが冥王星に対応する。0キーでは太陽を選択可能だ。

選択した天体に接近するには、gキーを押せばいい。自動的にその天体が画面中央に来るように視点が移動し、視認できる距離まで接近する。微調整はHome／Endキーやマウスホイールで行う。なお、太陽系外の恒星（**画面3**）などを選択した場合は、gキーを何回か押すことで、素早く段階的に接近することができる。

惑星名や惑星軌道、星座名、恒星名などの表示には、それぞれ対応するキーを利用する（GTK＋対応版ではメニューも使える）。たとえは、惑星の軌道を表示するには、oキーを押せばいい

▼リスト1　celestia_start

```
#!/bin/sh
cd /usr/src/celestia-1.0.10
exec celestia
```

(1) ナビゲーション	
h、0	太陽を選択（ホーム）
1〜9	太陽系の惑星を選択
Enter	選択対象をキー入力
c	選択した天体を画面中心に
g	選択した天体に段階的に接近
Home	選択した天体に近づく
End	選択した天体から離れる
f	選択した天体を追尾する
(2) 自由移動	
F1、s	停止（速度0）
F2	秒速1kmに設定
F3	秒速1000kmに設定
F4	秒速100万kmに設定
F5	秒速1AU（天文単位）に設定
F6	秒速1ly（光年）に設定
a	速度を10倍にする
z	速度を1/10にする
q	移動方向を反転する
x	移動方向を画面中心に変更
(3) 時間の経過	
Space	時間経過の停止・再開
l	時間経過を10倍にする
k	時間経過を1/10にする
j	時間経過を反転する
(4) オプション	
u	銀河の描画（トグル）
n	惑星名の描画（トグル）
o	惑星軌道の描画（トグル）
v	HUDテキストの描画（トグル）
i	惑星大気（雲）の描画（トグル）
w	ワイヤーフレーム（トグル）
/	星座線の描画（トグル）
=	星座名の描画（トグル）
b	恒星名の描画（トグル）
[	恒星の表示数を減少させる
]	恒星の表示数を増加させる
,	視野角（FOV）を小さくする
.	視野角（FOV）を大きくする
(5) その他	
d	デモモードに移行
`	フレームレート（FPS）を表示
Ctrl－Q	Celestiaを終了

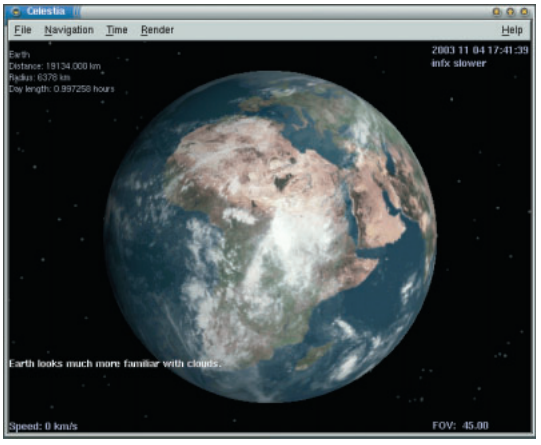
表1　キー操作一覧

（**画面4**）。再度oキーを押すと表示は消える（トグル動作）。また、星座線を表示したまま高速に太陽系外に移動すると、次第に星座の形が歪んでいくのを観察できる（**画面5**）。

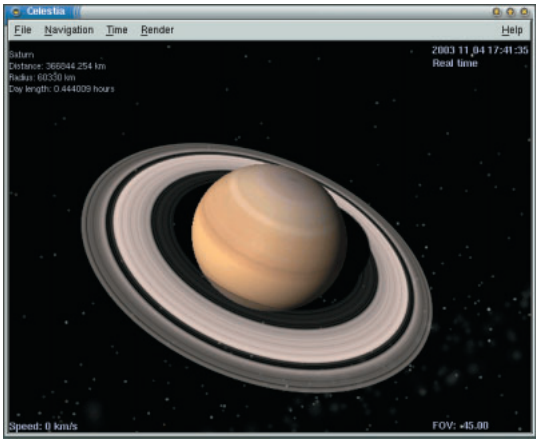
●太陽系外の惑星を眺めよう

Celestiaには、最近の観測で発見された太陽系外の惑星のデータも含まれている。たとえば、アンドロメダ座（Celestiaでは表示されない）のユプシロン星（HD 9826）で発見された3つの惑星を見てみよう。

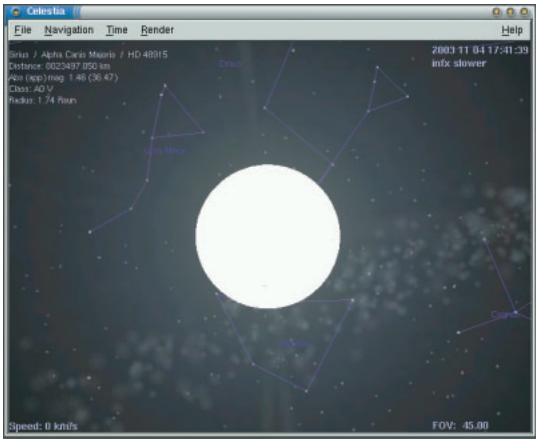
恒星の場合、いちいち画面から探し出すのは大変なので、名前で検索できるようになっている。Enterキーを押して「テキスト入力モード」に移行し、画面下の領域に恒星名（通称またはHD番号）を入力しよう。今回のユプシロン星なら「HD 9826」と入力してEnterキーを押せばいい。gキーを4、5回押して恒星に接近すれば、3つの惑星を持つ恒星系が画面中央に現れるはずだ（**画面6**）。



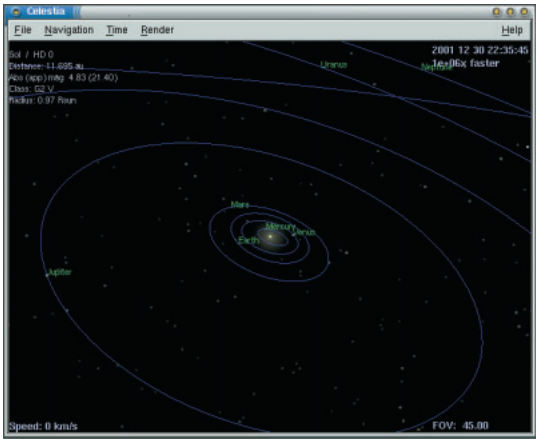
画面1　デモモードの出発点は地球。雲を重ねて表示することも可能だ。



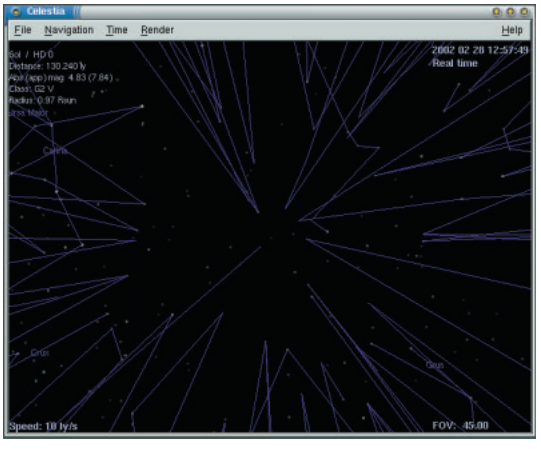
画面2　デモモードで土星を訪問。多層のリングが美しい。



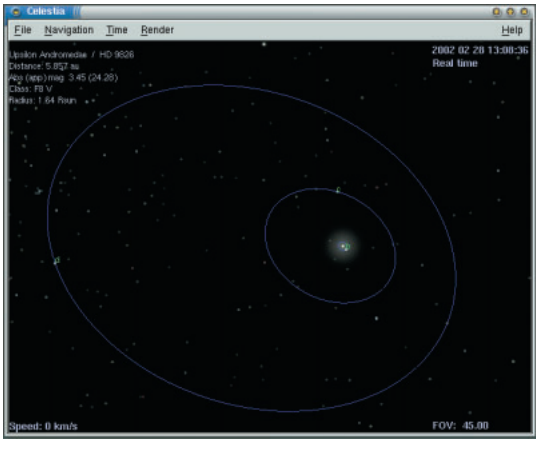
画面3　太陽系から8.5光年先のシリウス。小熊座の形が少し歪んでいる。



画面4　太陽系の惑星の軌道を表示。時間経過を速めて公転を観察しよう。



画面5　秒速10光年以上で飛行すると、星座の形がどんどん歪んでいく。



画面6　アンドロメダ座ユプシロン星とその周囲をめぐる3つの惑星を表示。

各種コマンドの実行やプロセス制御をGUIで行う

GNOME Workstation Command Center

バージョン：0.9.4　　ライセンス：GPL

http://gwcc.sourceforge.net/

GNOME Workstation Command Center（以下GWCCと表記）は、ネットワーク系（pingなど）やワークステーション系（lpqなど）の各種コマンドをボタン操作で実行し、結果をウィンドウ上で確認できるソフトだ。topコマンド風のプロセス一覧表示やプロセス制御、特定のプロセスの検索などを行ったり、各種のシステム情報をグラフなどでわかりやすく表示することもできる。実行にはGNOME／GTK＋が必要だ。

●ビルドとインストール

GWCCは、tarボールとRPMソース・バイナリ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。なお、RPMのバイナリパッケージはRed Hat 7用なので、Red Hat 6系ディストリビューションで利用するにはrpm 3.0.5以降が必要だ。

ただし、そのままでは、GWCCから起動するnslookupのコマンドラインに問題があり、正常に起動できない。また、固定幅フォントを使うと日本語が文字化けする不具合もある。

こうした不具合を修正するパッチ（gwcc-mb.patch）を作成したので、tarボールと組み合わせて使うとよいだろう。まず、tarボールの展開先で「gunzip -c gwcc-mb.patch.gz | patch -p1」としてソースを修正する。あとは、「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールするという一般的な手順だ。

●起動と初期設定

ktermなどで「gwcc&」とするか、GNOMEメニューの[プログラム]－[システム]－[GNOME Workstation Cmd Cntr]を選択すると、GWCCのウィンドウが開く（**画面1**）。機能別にタブでページ分けされた画面構成だ。

はじめて起動した場合は、GWCCから起動されるpingとnslookupのオプションパラメータが正しく設定されていないため、[設定]－[設定]で開く設定ダイアログの[Network Utils]ページ（**画面2**）を開いて、以下のようにパラメータを変更する必要がある。

まず、[ping Options]の[-c]を3、[-w]を10に変更する。また、[-i]は2に設定して[適用]ボタンを押してから、1に戻しておこう。続いて、[nslookup Options]の[Number of retries]を4、[Timeout]を5に設定し、[OK]ボタンでダイアログを閉じれば準備完了だ。次回からはこの設定が引き継がれるので変更の必要はない。

●ワークステーション系コマンド

最初に表示される[Workstaion Utils]ページでは、左側に並んだボタンをクリックするだけで、ifconfig、netstat、df、lpq、env、echo $PATH、wコマンドを実行できる。

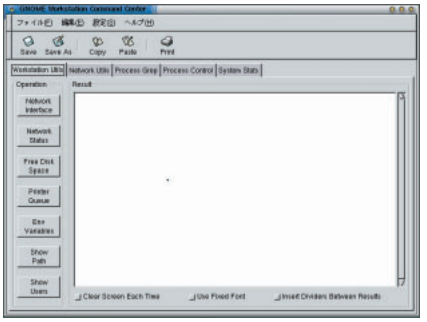
たとえば、[Free Disk Space]ボタンを押すとdfが実行され、マウントしているパーティションの総容量や空き容量などが右側のスクリーンに一覧表示される。また、[Show Users]ボタンではwが実行され、ログイン中のユーザーの情報が一覧表示される、といった具合だ（**画面3**）。

なお、[Network Interface]ボタンによるifconfigの実行にはroot権限が必要なので、一般ユーザーでは何も表示されない。表示結果などについては、manコマンドを使って各コマンドのマニュアルを参照されたい。

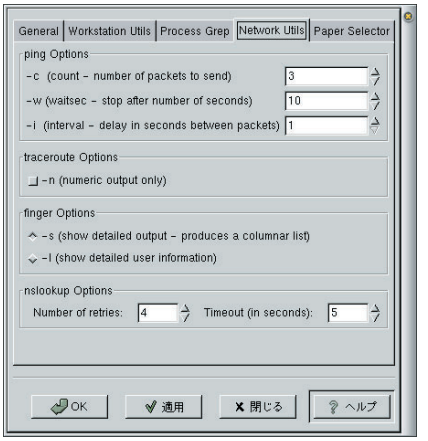
下に並んだ3つのチェックボックスは、結果の表示方法を制御するためのものだ。左から順に、毎回スクリーンをクリアする、固定幅フォントで出力を揃える、コマンドの出力間に境界線を挿入することを可能にする。

●ネットワーク系コマンドを使う

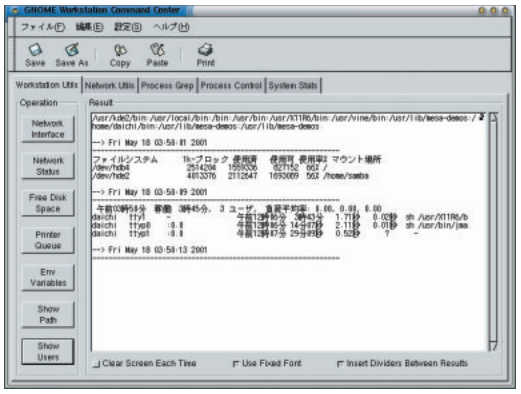
[Network Utils]ページでは、ping、whois、nslookup、traceroute、fingerといったネットワーク系のコマンドを実行できる（tracerouteの実行にはroot権限が必要）。



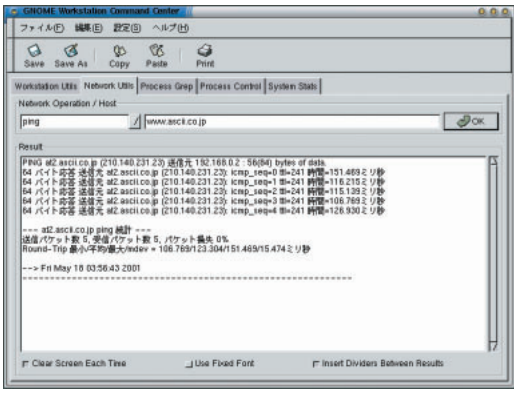
画面1　GWCCでは、機能ごとに上部のタブで画面を切り替える。



画面2　ネットワーク系コマンドのパラメータを正しく設定しておこう。



画面3　ディスク容量一覧（df）とログイン中のユーザー一覧（w）を表示。



画面4　pingやnslookupなどのネットワーク系コマンドもGUIで実行可能だ。

これらのコマンドは、対象となるドメイン名やユーザー名などを指定する必要があるので、ワークステーション系コマンドとは画面構成が異なる。まずは左側のリストからコマンドを選択し、続いて右側のテキストボックスに引数を入力する。

たとえばpingの場合は、リストから「ping」を選択し、対象となるホストのドメイン名（またはIPアドレス）をテキストボックスに入力して[OK]ボタンを押せばいい。実行結果は下のスクリーンに表示される（**画面4**）。

もし、pingやnslookupの結果が正しく表示されない場合は、「起動と初期設定」の節で説明したようにパラメータが設定されているか設定ダイアログで確認してみよう。

●プロセス関係の処理もGUIで

[Process Control]ページでは、実行中の全プロセスのプロセスIDやユーザー（所有者）、コマンド名などのリストが表示される（**画面5**）。インデックス（「PID」など）をクリックして、並び替えも可能だ。

また、クリックで選択したプロセスに対して、シグナル送信による強制終了を行える。操作は簡単で、送信するシグナル（SIGHUP／SIGTERM／SIGKILLの3種類）を下部のリストから選択し、[OK]ボタンを押すだけでいい。

また、[Process Grep]ページでは、指定した文字列を含むプロセス情報を検索して一覧表示できる。つまり、「ps | grep 文字列」などとするのと同じ処理を行うわけだ。

初期設定ではローカルなプロセスのみが対象となるので、設定ダイアログの[Process Grep]ページで設定を変更しよう。具体的には、[Grep Options]の設定を、「-e」や「-efH」に変更すればいい。

●システム情報の表示

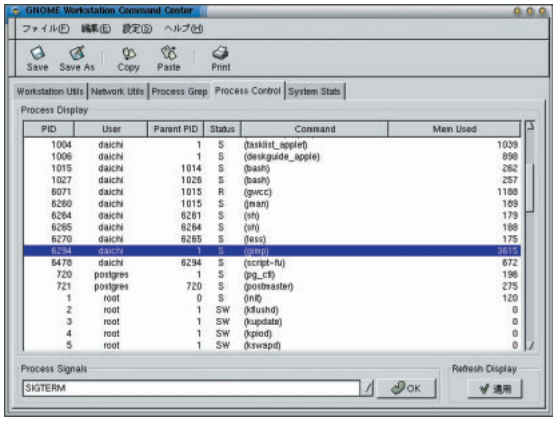
最も右側にある[System Stats]ページでは、さまざまなシステム情報をわかりやすく整理して表示する（**画面6**）。

左上の[Network Info]にはマシンのIPアドレスやSamba／Apache／Sendmailの状態が表示される。

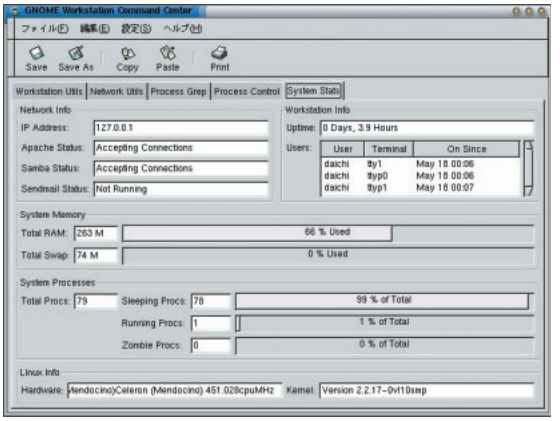
また右上の[Workstation Info]にはUptime（連続稼動時間）とログイン中のユーザー一覧が表示される。

[System Memory]と[System Process]では、グラフが効果的に使われている。前者は物理メモリとスワップの使用率、後者はスリープ／実行／ゾンビの各状態にあるプロセスの比率を棒グラフで表わしている。

一番下の[Linux Info]には、マシンのCPUと周波数、カーネルのバージョンが表示されている。



画面5　topコマンド風の一覧からプロセスを選択して、強制終了できる。



画面6　さまざまなシステム情報をまとめて表示する[System Stats]ページ。

KDE2用の高速イメージビューア

KuickShow

バージョン：0.8.1　　ライセンス：GPL

http://master.kde.org/~pfeiffer/kuickshow/

KuickShowは、KDE2上で動作するイメージビューアだ。JPEG／PNG／TIFF／BMP／GIFなど各種画像形式に対応する。画像の一覧が表示されたファイルブラウザから1クリックで実際の画像を表示でき、次の画像をあらかじめ先読みして高速に表示するなど、大量の画像を効率よく閲覧できるのが特長だ。フルスクリーン表示やスライドショー、画像の明るさやガンマ値の調整も行える。実行にはKDE2.1以降が必要だ。

●ビルドとインストール

KuickShowは、tarボールとRPMソース・バイナリパッケージが配布されている。ただし、バイナリパッケージはSuSe用で、インストール先（/opt/kde2）が国産ディストリビューションとは異なる。以下の手順でソースパッケージをインストールし、SPECファイルを修正後にバイナリパッケージを作成したほうがよいだろう。

まず、「su -」としてrootになった状態で、「rpm -i kuickshow-0.8.1-1.src.rpm」としてソースパッケージをインストールする。/usr/src/redhat/SPECSにインストールされたkuickshow.specを適当なエディタで編集し、4行目の「/opt/kde2」を「/usr/kde2」に変更して保存しよう。

続いて、「rpm -bb kuickshow.spec」とすると、修正したSPECファイルに基づくバイナリパッケージが、/usr/src/redhat/RPMS/i386に作成される。これを、「rpm -Uvh kuickshow-0.8.1-1.i386.rpm」としてインストールすればいい。

一方、tarボールを利用する場合は、「./configure」→「make」でビルドした後、「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。

●ブラウザウィンドウの操作

ktermなどのコマンドラインで「kuickshow&」とするか、KDEメニューの[グラフィックス]－[KuickShow」を選択すると、KuickShowのブラウザウィンドウが開く（**画面1**）。

ブラウザウィンドウには、KuickShowを起動したディレクトリ内の画像ファイルとサブディレクトリの一覧が表示される。サムネイル（縮小画像）表示はサポートされていないが、一覧表示と詳細表示を切り替え可能だ。

サブディレクトリへの移動は、ディレクトリ名をクリックするだけでいい。ツールバーのボタンを利用して、ひとつの上のディレクトリや、以前表示していたディレクトリに簡単にも戻ることができる。

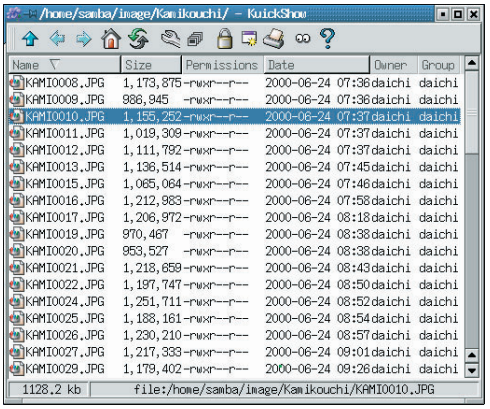
画像の表示もファイル名をクリックするだけだ。独立したイメージウィンドウが開いて、実際の画像が表示される（**画面2**）。このとき、画面サイズより大きな画像は自動的に縮小される。また、一定サイズ以下の画像を自動的に拡大する設定も可能だ。

通常は、画像ファイルをクリックした数だけイメージウィンドウが開いて、複数の画像を同時に閲覧できる。一方、ツールバー右から4番目の[Open only one window]ボタンを押した状態では、イメージウィンドウはひとつに保たれ、画像ファイルをクリックするたびに表示内容が切り替わる。

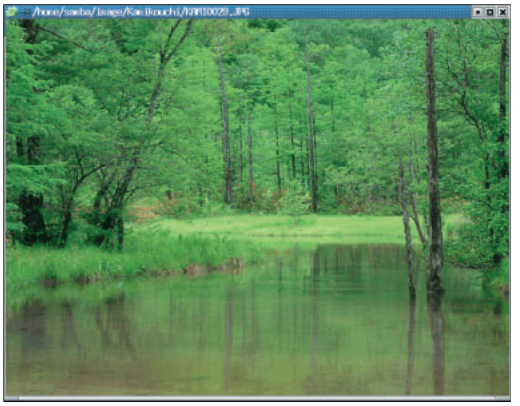
●イメージウィンドウでの操作

イメージウィンドウでは、右クリックメニューやショートカットキー（**表1**）により、前後の画像への切り替えや画像の拡大／縮小、回転や反転、明るさ／コントラスト／ガンマ値の変更、さらには印刷や別名での保存といった操作を行える。

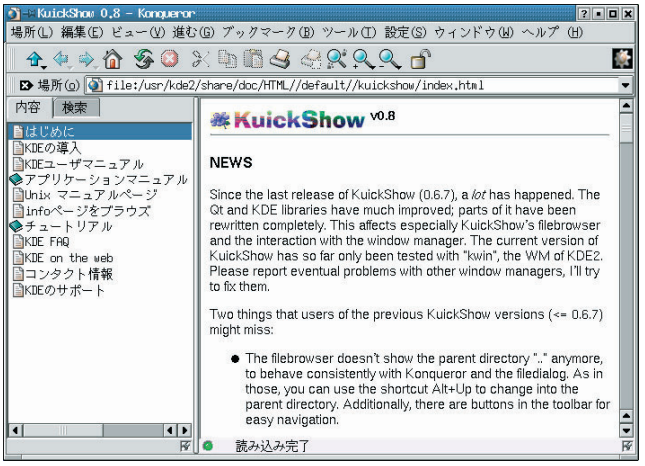
特に、ショートカットキーを覚えると、マウスより素早い操作が可能にな



画面1　ブラウザウィンドウには画像ファイルの一覧が表示される。



画面2　画像は独立したイメージウィンドウ（複数可）に表示される。



画面3　Konquererに表示されるヘルプでキー割り当てなどを確認しよう。

PageUp	前の画像に切替
PageDown	後の画像に切替
Home	先頭の画像に切替
End	末尾の画像に切替
＋	画像を拡大
－	画像を縮小
o	元のサイズで表示
m	最大化して表示
＊	左右反転
／	上下反転
7	左90度回転
9	右90度回転
8	180度回転
カーソル	上下左右にスクロール
Enter	フルスクリーンモード（トグル）
Space	ブラウザウィンドウの隠蔽（トグル）
Esc、q	イメージウィンドウを閉じる
b	明るさを増す
B	明るさを減らす
c	コントラストを増す
C	コントラストを減らす
g	ガンマ値を増す
G	ガンマ値を減らす
Delete	画像を削除（Shiftキーで無確認）
F1	ヘルプを表示
Ctrl－Q	KuickShowを終了

表1　イメージウィンドウのショートカットキー一覧

る。たとえば、前後の画像に切り替えるには、PageUp／Downキーを押せばいい。KuickShowでは、次に表示する画像をあらかじめ先読みしているため、瞬時に切り替わる。

このほか、Enterキーを押すとフルスクリーン表示とウィンドウ表示を交互に切り替えられるし、Spaceキーを押すとブラウザウィンドウを隠すことができる。キー操作についての詳細は、ツールバー右端の[Help]ボタンやF1キーで表示されるヘルプ（**画面3**）を参照されたい。

また、ツールバー中央の[Slideshow]ボタンを押すと、現在のディレクトリの全画像が一定時間（初期値は3秒）ごとに切り替わる「スライドショー」が始まる。現時点では、表示対象の画像ファイルをあらかじめ選択しておくことはできない。

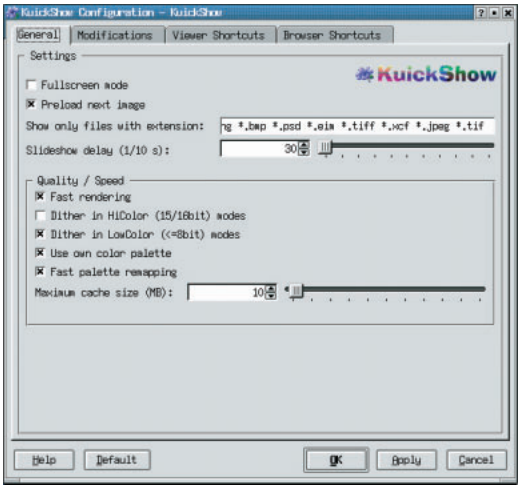
●キー割り当てなどをカスタマイズ

KuickShowでは、ブラウザウィンドウに表示する画像ファイルを、ファイル名のパターン（たとえば「＊.jpg」など）だけで判断している。

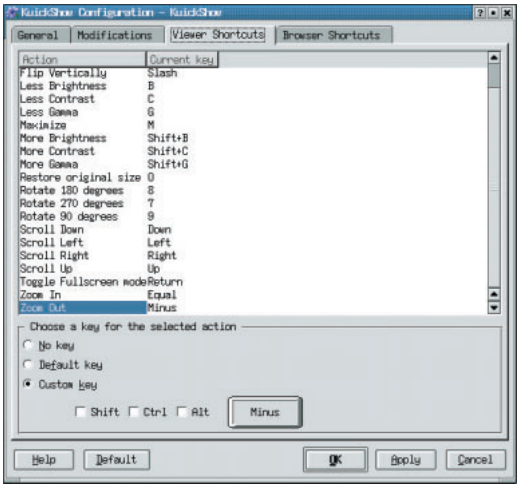
このため、対応している画像形式であっても、登録されているパターンと一致しないファイル名だとブラウザウィンドウには表示されない。拡張子「jpeg」や「tif」といった画像ファイルがこれに相当する。

こうした場合は、ツールバーの[Configration]ボタンで設定ダイアログを開き、[General]ページ（**画面4**）の[Show only files with extension]に、ファイル名パターン「＊.jpeg」や「＊.tif」を追加登録すればいい。このほか、スライドショーの表示間隔も、[Slideshow delay]で変更可能だ。

また、[Viewer Shortcuts]ページ（**画面5**）や[Browser Shortcuts]ページでは、イメージウィンドウやブラウザウィンドウで使われるショートカットキーを、各自の好みにあわせて変更できる。



画面4　一覧表示の対象となる画像ファイル名のパターンなどを設定。



画面5　キー割り当ては設定ダイアログで柔軟にカスタマイズできる。

多人数対戦も可能な海戦シミュレーションゲーム

Batalla Naval

バージョン：1.0.2　　ライセンス：Artistic

http://batnav.sourceforge.net/batnav-en.html

Batalla Navalは、スペイン生まれのマルチプレイヤー海戦シミュレーションゲームだ。昔のボードゲーム「バトルシップ」のように、ボード上に配置した敵船の位置を探りつつ、交互にミサイルを発射して破壊する。最大8名までの同時対戦に対応しており、ロボット（AI）を相手に対戦したり、付属の独自サーバやGGZ Gaming Zoneを利用したネットワーク対戦も可能だ。動作にはGNOME／GTK＋が必要となる。

●ビルドとインストール

Batalla Navalは、サーバ、クライアント、ロボットの各プログラムがひとつのtarボールにまとめられた状態で配布されている。

Batalla Navalをより楽しんでいただくために、メニューやメッセージを日本語化するカタログ（ja.po）などを追加するパッチ（gbatnav-ja.patch）を用意した。tarボール展開先のディレクトリで、「gunzip -c gbatnav-ja.patch.gz ¦ patch -p1」とすると、パッチによりソースが修正される。今回紹介する画面はすべてこのパッチを当てた状態のものだ。

あとは、「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。gbnserver（サーバ）とgbnclient（クライアント）、gbnrobot（ロボット）が/usr/local/binにインストールされる。

●サーバとクライアントの起動

最初に、ネットワーク接続されたマシンのひとつで、Batalla Navalのサーバを起動する。一般ユーザーのまま、ktermなどから「gbnserver&」とすればいい。サーバのウィンドウには、最大8名までのプレイヤーのリストが表示される（**画面1**）。

続いて、ゲームに参加するプレイヤーのマシンで、Batalla Navalのクライアントを「gbnclient&」として起動する（**画面2**）。接続用ダイアログが自動的に開くので、さきほどサーバを起動したホスト名（同一マシンの場合は「localhost」）や、プレイヤー名などを設定し、[OK]ボタンを押して接続しよう（**画面3**）。

インターネット経由でサーバに接続することも可能だが、あらかじめ接続先のドメイン名（またはIPアドレス）とポート番号（初期値は1995）を知っておく必要がある。

このほか、汎用ゲームサーバ・クライアントシステムの「GGZ Gaming Zone」（http://ggz.sourceforge.net/）を利用して、Batalla Navalをプレイすることも可能だ。この場合は、GGZのクライアントソフトを使ってGGZサーバに接続し、ゲーム一覧からBatalla Navalを選択する。

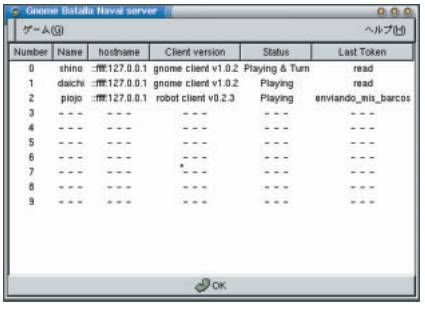
ただし、正式公開されているバージョン0.0.3より新しいGGZ（0.0.4以降）をCVSで入手する必要がある。また、GGZサーバ側でもBatalla Naval用の設定ファイルを追加しなければならないので、今回は省略する。詳細は同梱のREADME.GGZを参照されたい。

●船をボード上に配置する

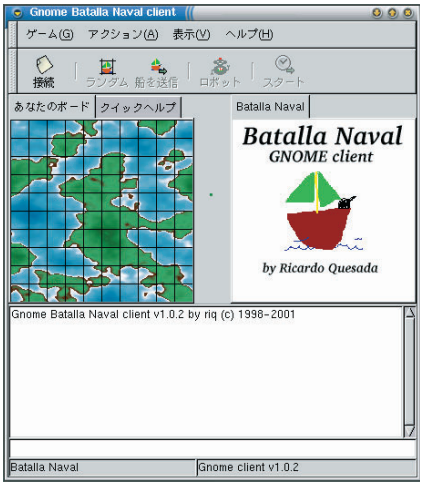
ゲームに参加するプレイヤーは、自分の船をボード上に配置する必要がある。船は全部で10隻で、

4マスを占める船…1隻
3マスを占める船…2隻
2マスを占める船…3隻
1マスを占める船…4隻

という対称的な構成だ。これらを、お



画面1　ひとつのサーバには、プレイヤーが8名まで参加できる。



画面2　プレイヤーごとに起動するクライアントのウィンドウ。

画面3　サーバを動かしているマシンのホスト名などを設定する。

互いに隣接しないように配置する（**画面4**）。ボード上の陸地は単なる模様なので気にしなくていい。

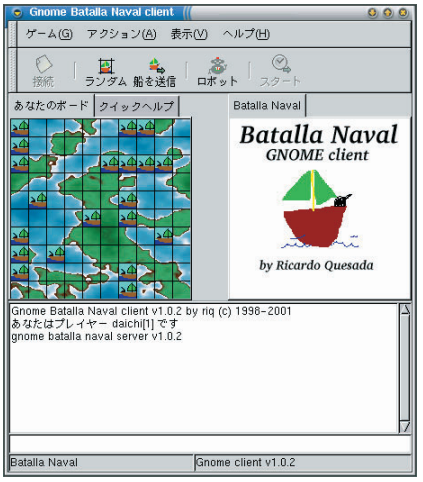
いちいちマウスで配置するのが面倒なら、ツールバーの[ランダム]ボタンを押すたびに適当な配置が行われる。気に入った配置になったら、[船を送信]ボタンを押して、準備が整ったことをサーバに伝えよう。

また、人間の対戦相手が見つからない場合は、[ロボット]ボタンを押して、対戦用のロボット（AI）を起動しておく（**画面5**）。サーバのリストにロボットが追加されたことを確認しよう。複数のロボットを相手に対戦することも可能だ。

[スタート]ボタンを押すと、準備が完了した他のプレイヤーやロボットとのプレイが始まる。相手が一人だけの場合は1対1の対戦、二人以上いる場合は全員と同時に戦いを繰り広げるマルチプレイヤー対戦となる。

●相手の船の位置を読め

ウィンドウ右のエリアには、全プレイヤーのボードが表示されている。ボードはプレイヤーごとに独立しているので注意されたい。



画面4　10隻の船を互いに隣接しないように配置する。自動配置も可能。

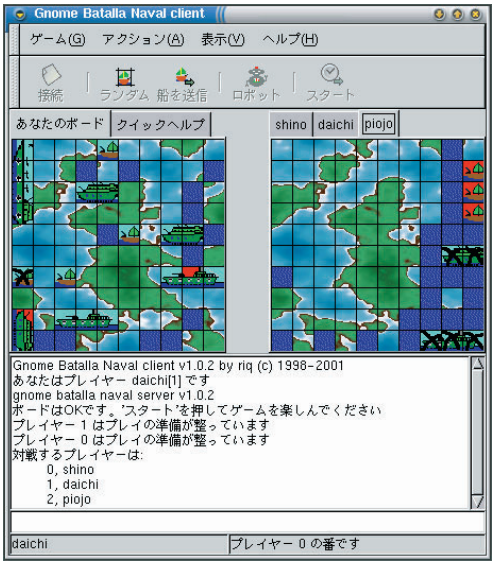
画面5　コンピュータが操作するロボットとの対戦も可能だ。

ゲームはターン制で進行し、自分のターンに一回ずつ、自分以外の任意のプレイヤーのボード中の1マスをクリックして、ミサイルの発射を指示する（**画面6**）。敵が複数いる場合は、上部のタブでボードを切り替えよう。

クリックしたマスに相手が船を配置していると、1マスだけの小さな船ならば即座に破壊され、×印が表示される。一方、2マス以上を占める大きな船の場合は、赤い被弾マークが表示され、残りのマスがすべて被弾した時点で破壊される。被弾マークが表示されたら、その上下左右のいずれかのマスに必ず残りの船体が存在するので、連続して攻撃を加えよう。

なお、被弾マークのついたマスの周囲斜め4マスや、破壊された船の周囲すべてのマスは自動的に開いて海が表示される。他の船と隣接できないルールなので、それらのマスには船体が存在しないことが確定するからだ。この性質を利用して、大型船を素早く発見して破壊するのがコツだ。

自分の船がすべて破壊されたプレイヤーはゲームから抜け、最後に残ったプレイヤーが勝者となる（**画面7**）。なお、ロボットのプレイヤーは自分が負けたり、プレイが終了すると自動的に待機状態となるため、すぐに次のプレイを開始できる。



画面6　相手の船の位置を読んでミサイルを発射しよう。

画面7　最後まで残ったプレイヤー一人が勝利者となる。

溜め撃ちが楽しい横スクロールシューティングゲーム

Geki3

バージョン：0.3.6　　ライセンス：GPL

http://www2.mwnet.or.jp/~fc3srx7/

Geki3は、国産の横スクロールタイプのシューティングゲームだ。先月紹介した「Geki2」と同系統のゲームだが、溜め撃ちによる強力なレーザー攻撃や自機のダメージ蓄積制の導入など、プレイ中のテイストはかなり異なるものになっている。全部で4つのステージが用意されており、各ステージの最後には特徴あるボスキャラが待ちうけている。実行には同じ作者のKXLライブラリが必要だ。

●ビルドから起動まで

Geki3とKXLライブラリは、どちらもtarボールのみ配布されている。先月号でKXLを導入していない人は、KXLライブラリを「./configure」→「make」としてビルドし、「su」でrootになってから「make install」でインストールしよう。Geki3のインストールもまったく同じ手順だ。

ktermなどのコマンドラインで「geki3&」とすると、ウィンドウが開いてタイトル画面が表示される。zキーを押すとプレイ開始だ。

●基本的な遊び方

自機（ペンギン）を操作して画面の右から迫り来る敵を倒そう（**画面1**）。カーソルキーが上下左右への移動、zキーが通常弾の発射だ。zキーを押しつづけることで連射できる。

溜め撃ちによるレーザー攻撃を行うには、zキーを離してしばらく待つ必要がある。画面上部の青いバーがフル充電されれば準備完了だ。zキーを押すと、通常弾とは比べ物にならない破壊力のレーザーが発射される。レーザー攻撃は、青いバーがすべて消えるまで有効だ。

ザコ相手にレーザーを使う場合は、zキーを押しっぱなしにせず、断続的に使うといい。発射後のレーザーの縦位置は自機と連動するため、短いレーザーでも複数の敵を攻撃できる。

●アイテムで回復とパワーアップ

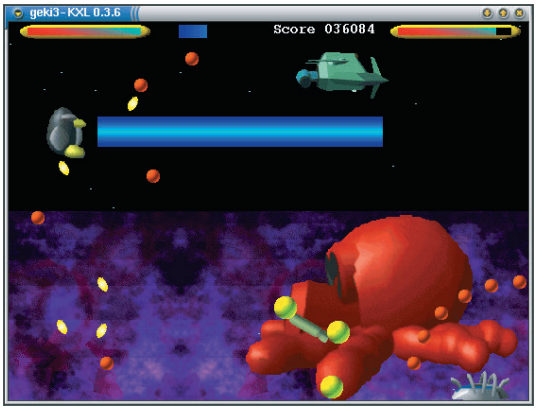
自機のほうは、敵の弾や敵自身に接触するとダメージを受け、左上のバーがしだいに短くなる。完全に消えるとゲームオーバーなので、特定の敵を倒すと現れるアイテム（白）を取ってダメージの回復を図ろう。

また、アイテム（青）では通常弾のパワーアップ、アイテム（赤）ではミサイルの追加とパワーアップが行われる。最大限パワーアップすると、レーザーを使わなくても強力な攻撃が可能になる（**画面2**）。

各ステージの最後には、「お約束」の巨大なボスキャラが待ちうけており（**画面3**）、これを倒せばステージクリアだ。全部で4ステージが用意されており、すべてクリアすると最初のステージに戻る。

画面1
隙の無い通常弾と強力なレーザー攻撃をうまく使い分けよう。

画面2
通常弾やミサイルもアイテムを集めることでパワーアップする。



画面3
各ステージの最後には、巨大なボスキャラが待ちうけている。

パネルを回転させてボールを移動するパズルゲーム

Groundhog

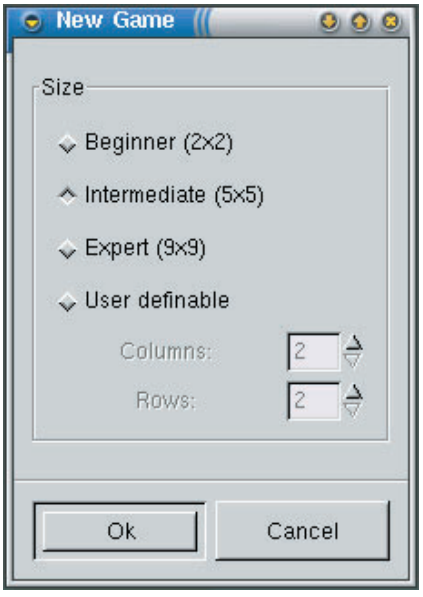
バージョン：1.3　　ライセンス：GPL

http://home-2.consunet.nl/~cb007736/groundhog.html

Groundhogは、2本のチューブが描かれた複数のパネルを回転させて、すべてのボールを同じ色のポケットに格納する思考パズルゲームだ。ルールは単純だが、ボールやパネルの数が増すにつれて経路が複雑になり、同時に複数のボールを思い通りに動かすには、かなり頭を使わなくてはならない。クリアまでの時間の短い上位10名がハイスコアに名前を残せる。動作にはGTK＋が必要だ。

●ビルドから起動まで

Groundhogは、ソース一式をtar＋gzipしたtarボールのみ配布されている。「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。



画面1　最初にパネルの数を設定する。まずは小さなものから挑戦だ。

ktermなどのコマンドラインで「groundhog&」として起動すると、5×5の配置に並べられたパネルと、その周囲に4色のカップとボールが並んだウィンドウが開く。

ゲームを始めるには、ツールバー左端の[Start new game]ボタンを押して、パネルの数を選択する（**画面1**）。はじめのうちは、初期設定の「Intermediate」（5×5）か、さらに小さな「Beginner」（2×2）で、ボールの動きに馴れるといい。

なお、「User definable」を選択した場合は、行と列のパネル数を独立して指定できる。「2×4」など正方形でない構成も設定可能だ。

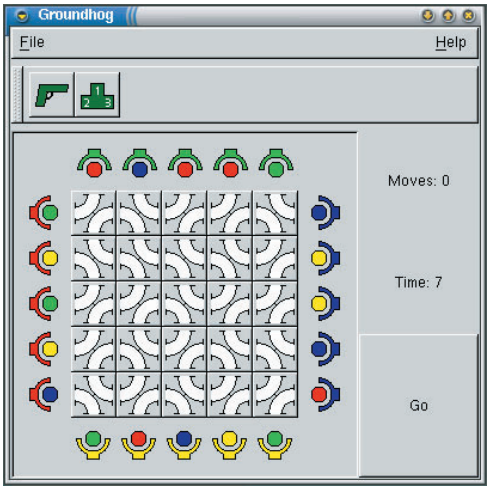
●パネルを回転させて経路を作る

開始直後は、ほとんどのポケットの色と、その中に格納されているボールの色が一致していない状態になっている（**画面2**）。すべてのポケットとボールの色が一致した状態に移行することがゲームの目的だ。

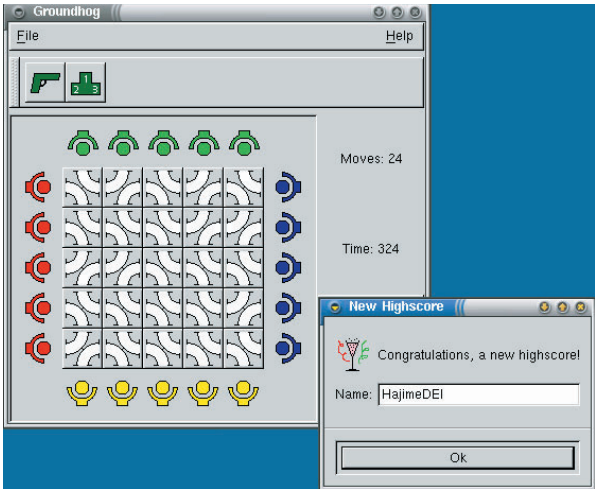
そのために、縦横に並んだパネルをクリックして回転させる。どのパネルにも、90度に曲がったチューブが2本描かれており、周囲のパネルのチューブと組み合わせてボールの経路が形成される。

複数のパネルを回転させることで、違う色のポケットに格納されているボールを、同じ色のポケットまで移動させる経路を作るわけだ。右下の大きな[Go]ボタンを押すと、実際にボールの移動が行われ、それぞれの経路の両端にあるポケットのボールが交換された状態になる。

一度の移動で色が揃わなかったら、再度パネルを回転させて別の経路を生成する。せっかく同色のポケットに格納したボールが、別のボールと交換されてしまうこともあるので気をつけよう。うまい具合にすべてのボールとポケットの色を一致させるとゲームクリアとなる（**画面3**）。



画面2　パネルを回転させてボールを同じ色のポケットまで運ぼう。



画面3　この状態になればゲームはクリア。時間が短いほど上位になる。

たくさんのかわいいペンギンが画面を走り回る

XPenguins

バージョン：2.0　　ライセンス：GPL

http://xpenguins.seul.org/

XPenguinsは、ペンギンのTux君がデスクトップを駆け回るソフトだ。複数のTux君がスケートボードで滑ったり、風船にぶら下がりながら降りて来たりとさまざまな芸を見せてくれる。テーマによる画像パターンの切り替えも可能で、Tux君を含め3種類のテーマが付属する。このほか、使われる画像パターンは古いものの、GNOMEパネルで実行や一時停止を制御できるアプレットも用意されている。

●ビルドから実行まで

XPenguinsおよびXPenguins Appletは、どちらもtarボールとRPMソース／バイナリ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ボールからのインストールは、configureスクリプトを利用する一般的な手順だ。

ktermなどから「xpenguins&」として起動すると、画面上部から8匹のTux君が降りてきて、ウィンドウやパネル上を歩き始める（**画面1**）。スケートボードで疾走したり、しばし読書にふけったり、スーパーペンギンに変身して空を飛んだり…と眺めているだけで飽きない。

ウィンドウのレイアウトによっては、Tux君の移動が阻害されることがあるので、ウィンドウをずらして通り道を作ってあげよう。

こうした標準テーマ（Penguins）以外に、以前のバージョン（Classic Penguins）や、亀（Turtles）のテーマが用意されている。テーマを切り替えるには、起動時に-tオプションでテーマ名を指定する。たとえば、「xpenguins -t Turtles&」とすると、ペンギンの代わりに亀が表示される（**画面2**）。

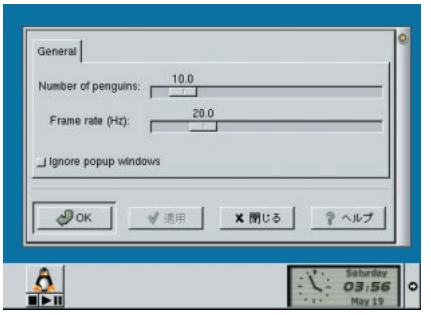
表示数を変更する-nオプションや、残酷な画像を表示しない-bオプションなども用意されている。なお、バックグラウンド実行したXPenguinsを終了するには、ktermなどのターミナルソフトから、「killall xpenguins」とすればいい。

●GNOMEパネル用のアプレットも

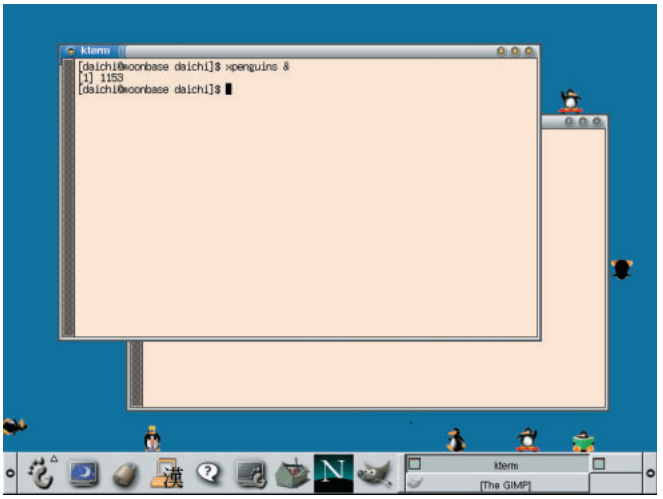
XPenguins Appletは、GNOMEパネル上で右クリックして、メニューから[アプレット]－[遊び]－[XPenguins]を選択するとパネルに常駐する。

操作は簡単で、右下の[再生・一時停止]ボタンを押すとTux君が登場し、左下の[停止]ボタンで消える。Tux君の数やフレームレートなどの設定は、上部のボタンで開くダイアログで変更可能だ（**画面3**）。

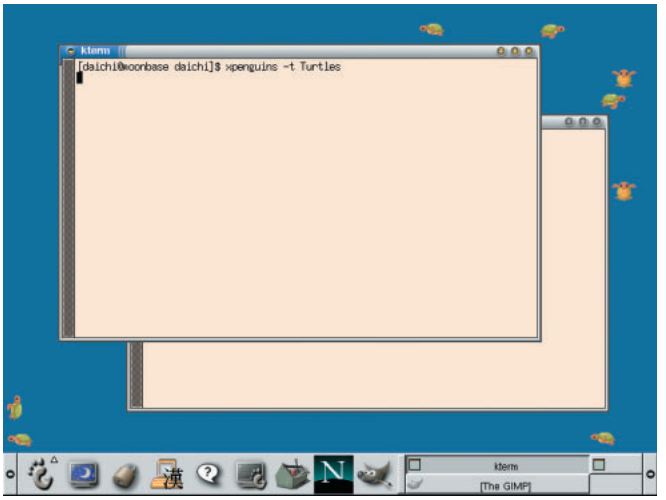
なお、XPenguins AppletはXPenguinsを起動するフロントエンドではなく、画像パターンや表示ルーチンが内蔵された独立したソフトだ。画像パターンは「Classic Penguins」テーマに相当する古いもので、テーマの切り替えには対応していない。



画面3　GNOMEパネルから表示を制御できるアプレット版もある。



画面1　LinuxのマスコットTux君があなたのデスクトップを駆け回る。



画面2　「Turtles」テーマでは、ペンギンではなく亀が表示される。

複数の配列を切り替えられる仮想キーボード

xvkbd

バージョン：1.4　　ライセンス：GPL

http://member.nifty.ne.jp/tsato/xvkbd/index-j.html

xvkbdは、キー入力をマウスで代行する仮想キーボードだ。画面上に表示されたキーボードのボタンをクリックすることで、他のソフトにそのキーの入力を伝える。ファンクションキーなどに文字列を登録して一括送信することも可能だ。標準ではUS配列のフルキーボードが使われるが、Xのリソース設定により、別のキー配列に切り替えたり、キーボードの大きさを変えることもできる。

●マウスを使ってキー入力

xvkbdはソース一式がtar＋gzipされたtarボールのみ配布されている。「xmkmf」→「make」でビルドし、「su」でrootになってから「make install install.man」としてインストールする。このとき、複数のリソース設定ファイルが/usr/X11R6/lib/X11/app-defaultsにコピーされる。

ktermなどから「xvkbd&」として起動すると、US配列のテンキー付きフルキーボードが表示される（**画面1**）。この状態で、表示されているキーのボタンをクリックすると、フォーカスを持つウィンドウに対してキー入力が行われる。Shiftキーなどは一度押すとロックされるので、Ctrl－Alt－＋（テンキー）などの組み合わせも可能だ。

左下の「xvkbd」をクリックするとポップアップメニューが開いて、キーパッド（テンキー）のみの表示や、xvkbdの終了を行える。

●ファンクションキーに文字列登録

ファンクションキー（F1～F12）のボタンを押すと、通常は対応するキーコードが送信されるが、その代わりに任意の文字列（日本語不可）を送信することもできる。メールアドレスやURLなど、頻繁に入力する文字列を登録すると便利だ。各キーに登録されている内容は、キーのボタン上にカーソルを合わせるとポップアップするチップヘルプで確認できる。

文字列の登録は、各ユーザーのホームディレクトリの「.xvkbd」で行う。キーの名前と対応する文字列を「F1 This is a test.」といった具合に、1行に1つずつ書けばいい。キー名を「s:F1」や「c:F1」などとすることで、ShiftキーやCtrlキーとの組み合わせも指定できる。

このほか、起動時に-textオプションで指定した文字列を直接送信する機能もある。この場合は、xvkbdのウィンドウは表示されない。

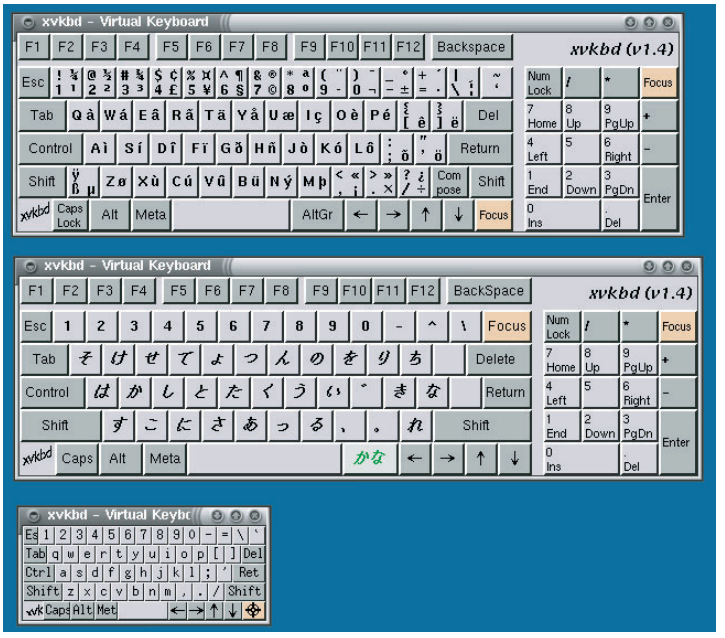
●キー配列を切り替える

キー配列を切り替えるには、ホームディレクトリの.Xdefaults（あるいは.Xresources）を書き換えて、Xのリソースデータベースを更新する。

たとえば、テンキーがなく、よりサイズの小さな「small」キーボードを利用するには、.Xdefaultsに「xvkbd.customization: -small」という行を追加すればいい。

Xを再起動するか「xrdb .Xdefaults」とした後、xvkbdを起動すると新たなキー配列で表示される（**画面2**）。このほか、ドイツ版キーボードや今では幻となった新JISキーボードなどのリソース設定ファイルに切り替えることも可能だ。設定次第では、ユーザーオリジナルのキー配列も実現できる。

画面1
US配列のフルキーボードが表示され、マウスでキー入力を行う。



画面2
リソース設定により、キー配列やキーボードサイズを変更できる。

Software Catch up

設計から運用までをWebブラウザで実現したカード型データベース

サイボウズDBメーカー

使い慣れたWebブラウザでデータベースの設計から運用までを実現したサイボウズDBメーカーは、誰でも簡単に利用できるデータベースである。最近の複雑なデータベースとは逆の発想をした手軽なデータベースといえるだろう。

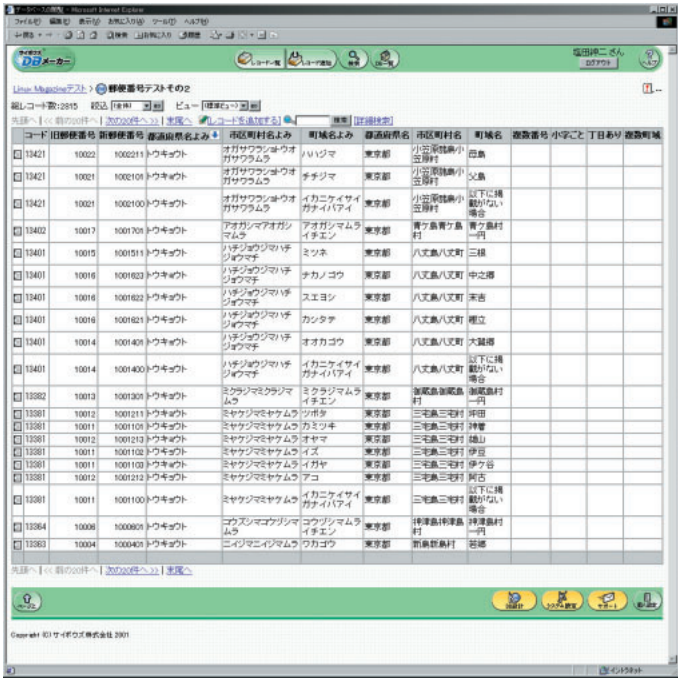
文：塩田紳二
Text : Shinji Shioda

価格	未定（ライセンス方式）
問い合わせ先	サイボウズ株式会社 03-5805-9035 http://cybozu.co.jp/

サイボウズDBメーカーは、Webサーバを経由して利用するデータベースシステムだ。グループウェアであるサイボウズOfficeで有名なサイボウズが開発し、現在はβ版が公開されている。このため、実際の製品版では、この記事とは異なる部分があることをあらかじめお断りしておく。

このサイボウズDBメーカー（**画面1**。以下、DBメーカー）は、同社のサイボウズOfficeで用いられているデータベースエンジン（CyDE：サイド。Cybozu Database Engineの略）を使い、データベースシステムを簡単に構築できるようにしたものだ。ただし、DBメーカーは、リレーショナルデータベースではなく、単一のテーブルのみを扱うものである。かつてはやった言い方をすると「フラットファイルデータベース」というところか。

リレーショナルデータベースは、複数のテーブルを定義してそれらの関係をさらに指定してデータベースの検索などを行う。もっとも、リレーショナルデータベースシステムを使っているからといって、世の中のすべてのデータベースが「リレーショナル」なわけではなく、中には単一の表のみのデータベースというのも数多く存在する。ある意味、「大は小を兼ねる」ということで、複雑な処理を行うリレーショナルデータベースをリレーショナルなしで利用しているわけである。



画面1　DBメーカー
DBメーカーは、Webブラウザ経由で利用するデータベースシステムだ。データベースの設計から入力、閲覧まですべてブラウザ経由で行える。

たとえば顧客のデータベースだけなら、これは単一のテーブルで実現可能である。しかし、在庫や注文のデータベースシステムとなると、製品や在庫、注文などの複数のテーブルを使って、複雑なデータベースを扱う必要がある。

DBメーカーは、前者のような単一のテーブルを使うデータベースを簡単に構築するためのもので、Webサーバと組み合わせて利用し、データの入力や検索など、すべての操作はWebブラウザ経由で行う。このためインストール後の利用は非常に簡単になっており、クライアント側に特にプログラムを用意する必要なく、Webブラウザが利用できる環境があればよい。仕組みとしては、CGI経由で起動するメインプロ

グラムでデータベースのアクセスや処理などを行い、結果をHTMLでクライアントに戻すようになっている。

なお、市販のデータベースシステムは、エンジン部分と、入力やレポート作成のGUI作成ツールやプログラム記述言語との組み合わせになっているが、DBメーカーはエンジン部分を直接CGIから起動し、GUI部分にはHTMLを利用する。現在のβ版にはロジックを記述する言語はなく、ユーザーがDBメーカーをベースにしたアプリケーションを作成できるようにはなっていない。データベースをWebブラウザ経由で作成、編集し、ビューや絞り込みを行い、レコードのブラウズが可能なだけである。このため、DBメーカーはデータベースを直接参照するような使い方、たとえば顧客のデータを入れておいて共有するなどというものになる。とはいえ、これだけでもかなりの用途があり、本来、リレーショナルが必要ないデータに対してリレーショナルデータベースを入れて運用するよりも手軽でいいかもしれない。

このDBメーカーだが、現時点ではβ版で7月31日まで動作するものが同社のサイトに置かれている（本誌付録CD-ROMにも収録）。また、実際の製品化にあたってはデータベースの個数と利用期間（月数）で料金を決めることになるという。ある意味、ソフトウェアのレンタルに近い形の価格体系だ。詳細な価格は公開されていないが、30本のデータベースでひと月あたり2万円程度を予定している。

なお、データベースが11レコード以下であれば仮運用として、この料金にカウントされない形で利用できる。つまり、作成途中のものや、テスト的に作ったデータベースでは課金されず、作成後、データベースの本運用を開始したときから料金が発生するようになっている。

インストールと初期設定

DBメーカーのLinux版はtar.gz形式で配布されており、附属のスクリプトを利用するだけで簡単にインストールできるようになっている。ただし、インストールする前にHTTPサーバ（Apacheなど）が動作していて、CGIが利用できるようになっている必要がある。

付属のスクリプトは、質問に対して答えていくだけのもので、Apacheのインストールが特殊な形で行われていなければ、スクリプトがインストールディレクトリなどを調べてくれるため、ほとんどEnterキーを押すだけでインストールが完了する。

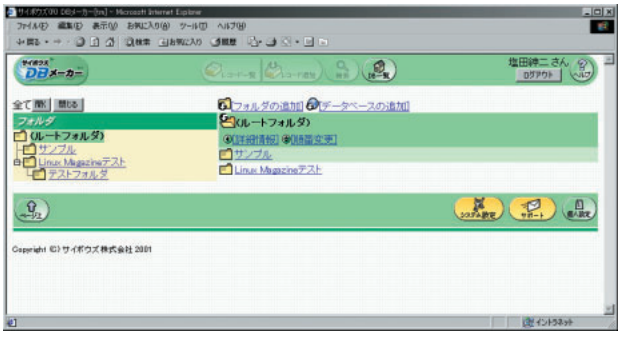
インストール後は、Webブラウザから、

```
http://ホスト/cgi-bin/cbdb/db.cgi?
```

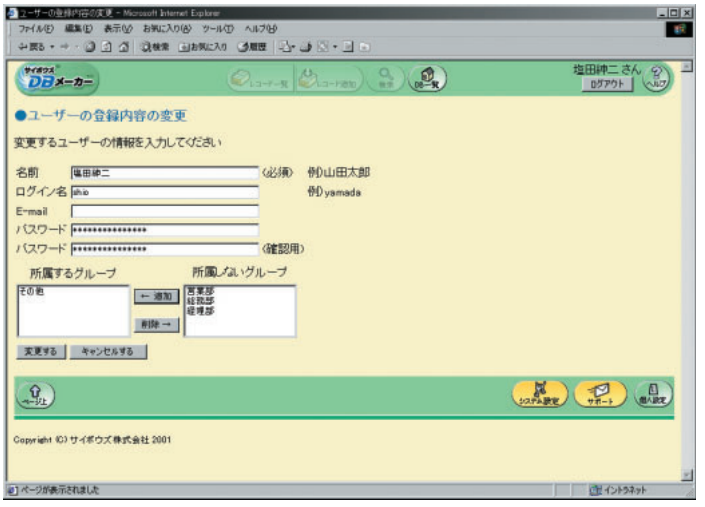
というURLを入力するだけで、DBメーカーの起動画面を表示することができる（**画面2**）。

DBメーカーは、ユーザーごとにデータベースなどへのアクセスを指定することができるため、対象ユーザーを管理する必要がある。ユーザー管理は、基本的にはLinuxのユーザーアカウントとは別で、DBメーカー独自の管理となる。また、サイボウズOfficeのユーザーを引き継ぐことも可能だが、そうでない場合にはDBメーカーのシステム設定メニューを使ってユーザーを登録する必要がある。

ユーザー登録は、名前やメールアドレス、ログイン名、パスワードなどを設定する（**画面3**）。DBメーカーでは、このユーザーをデータベースに対するアクセス管理に利用する。たとえば、データベース自体へのアクセスを可能



画面2　起動画面
起動すると、フォルダツリーとフォルダ内のデータベースの一覧が表示される。



画面3　ユーザー管理
DBメーカーでは、システムのアカウントとは別にユーザーを管理している。このため、インストール後はシステム側のユーザーアカウントなどを設定する必要はなく、単にDBメーカー内でユーザーを定義すればよい。

にするかどうか、あるいはレコード内のフィールドを変更可能にするかどうかなどを設定する際に、このDBメーカーの管理するユーザーをベースに処理を行うのである。

なお、グループを定義することでアクセス管理などをグループ単位で行うこともできる。ユーザーは、複数のグループに所属することができ、アクセス管理では個別のユーザーではなく、このグループで行うこともできるようになっている。

運用のやり方として、利用者にログインを行わせ、ユーザーを特定したうえで使うことが基本だが、設定により、ログイン操作を行わずにデータベースを利用させることもできる。ほとんどのユーザーが単に情報を見るだけで書き込みは特定のユーザーのみといった、回覧板的な利用方法であれば、このようにすることでユーザーはログイン操作することなくデータベースを見ることができる。

なお、DBメーカーではデータベース自体の作成は利用者全員が可能で、ログインしたユーザーのみ作成可能とする制限がかけられるだけとなっている。個々のユーザーに、DBメーカー利用の敷居を低くするためだと思われる。データベースを間違って作ってしまったところでほかのデータベースには影響はないのだが、ユーザーによってはこのようなデータベースの作成の機能などは隠せるような機能が欲しいところだ。必要なデータベースのみ見えていて、リンクをクリックすれば、必要な作業が始められるといった設定があったほうが場合によっては使いやすいからだ。筆者がかつて勤め先で、アルバイト数名にデータベース入力を行わせるといった仕事をしたことがあるが、管理者とすると、データベースを入力できるようにするまでの手順はあまり複雑でないほうがラクであった。たとえば、「ログインして、このコマンドを入れて、それからメニューから××を選んで……」となるよりも、「ログインして、出てきた画面で入力を始めてください」としておいたほうがラクだし、あまり作業内容を理解していないユーザー同士でも教え合うことができたからである。

GUI

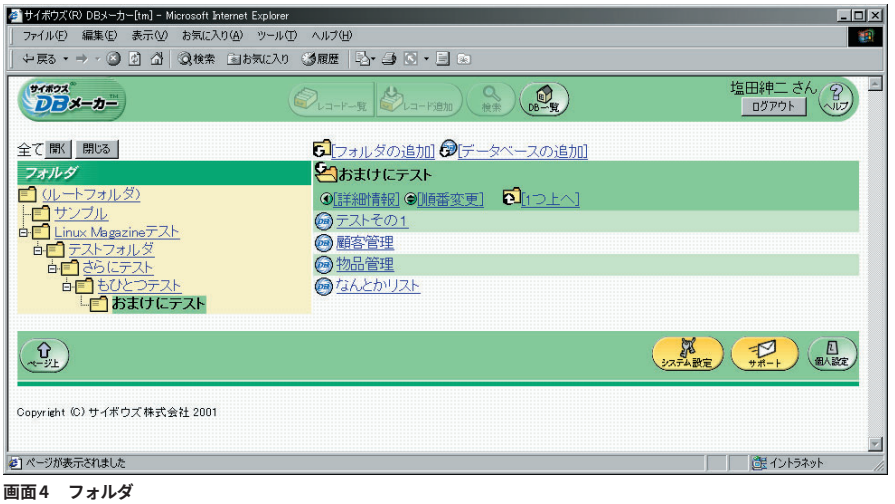
DBメーカーはWebブラウザから操作するため、そのGUIは基本的にWebブラウザが表示可能なHTMLで記述できる範囲となる。このため、GUIはHTMLのリンクやフォームを利用したもので、通常のGUIアプリケーションとはちょっと違った部分もある。また、JavaScriptなどを使うと多少凝ったことが可能だが、DBメーカーではごく一部にのみJavaScriptが使われているだけで、ほとんどはHTMLのみで記述が行われている。

HTMLで表現するためダイアログボックスというものがなく、すべてページを切り替えることで画面が表示される。Formを使うので、基本的にはフィールドなどに値を設定して、最後にSubmitボタンを押して変更を反映させるという作りとなる。またこのとき、キャンセルボタンを押すことで1つ前のページである、上位機能のページへ戻ることができる。

なお、データベースの作成からフィールドの定義あたりまでは、ページが次々と進行している形式になっており、一連の必要な手順を行うことができるが、別途、各フィールドの動作定義（最大文字列長や数値のフォーマットなど）は個別にフィールドの編集機能を使って設定する必要がある。

実際のデータの編集や入力も、レコードごとにフォームを使って入力を行う。DBメーカーでは、フィールドの属性自体もHTMLのフォームを想定した形式になっており、リスト選択（固定文字列選択入力）やチェックボックス（通常のデータベースでいう真偽値に相当）、ラジオボタンなどがある。レコード入力、編集画面では、これらに対応するフィールドが並べられ、ユーザーが入力後、「更新する」ボタンを押すことで編集が完了する。

なお、HTMLページを使うため、文



画面4　フォルダ
フォルダは階層形式になっており、各フォルダにはデータベースを配置できる。なお、DB設計でデータベースを格納するフォルダを変更することもできる。

章などでの説明はあるが状況に応じたヘルプ機能はなく、現時点ではサイボウズのWebサイトにあるオンラインマニュアルなどへのエントリが表示されるだけである。まあ、β版なので、このあたりは最終製品でどうなるか不明だが、できればヘルプボタンを押したときには表示しているページに関係したヘルプページが別ウィンドウで表示されてほしいものである。

データベースの作成

データベース自体は、ツリー構造のフォルダにより分類が可能だ（**画面4**）。これはあくまでも分類のためのものだが、作成したデータベースはあとからフォルダ間で移動することはできる。実際には、データベースは内部的なデータベース番号で管理されているが、1つのフォルダの中にすでにあるデータベースと同名のデータベースやフォルダ名は存在することはできない。フォルダによる処理の違いはないので、同名のデータベースを複数作りたい場合には別のフォルダに作成する。

データベースの作成は、フォルダを選び、データベース名やデータベースに対するメモなどの入力を行うところから始まる。また、ここでクライアント側にあるCSV（カンマ区切りファイル）からデータベースを自動的に定義することも可能である（**画面5**）。

次にフィールド名とタイプの定義を行う。ここでは、フィールド名を入力し、タイプを指定して「追加する」ボタンを押すことで、次々とフィールドを定義できる。ただし、各フィールドの詳細な設定はここでは行わず、あとでデータベース設計のページからフィールドの編集機能を使って行う。あまり複雑でない、たとえば住所録のようなものであれば、特にフィールドの細かい設定などを行わなくともデータベースを作ることは可能である。

最後に「完了する」を押せば、データベースの定義が終了する。また、データベース名やメモもフィールドの定義中に変更が可能で、最後に名前を変更して作成することも可能だ。

細かい設定は、ページの下にある「DB設計」ボタンを使って行う。ただし、データベースに対してアクセス権が設定されると、変更の権利を持つユーザーがログインしている場合のみこのボタンが表示される。

ここでは、個々のデータベースに対してフィールドや絞り込み、ビューの設定および、データベース自体のアクセス権、CSVファイルの読み書きなどが行える。

■データベースのアクセス権の設定

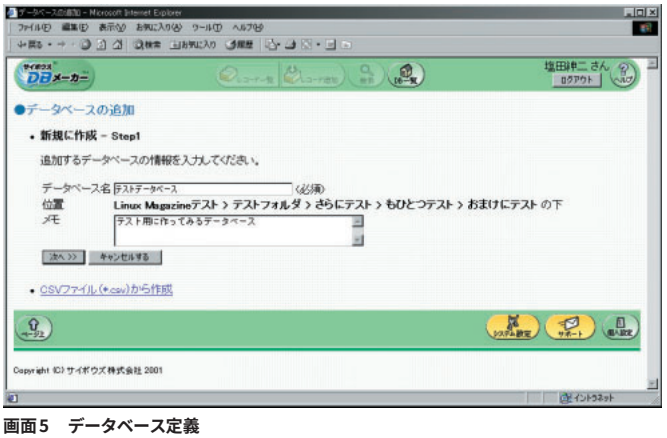
DBメーカーでは、データベース自体に対するアクセス設定として、レコードの、

- **閲覧、編集、追加、削除**
- **閲覧、編集、追加**
- **閲覧、編集**
- **閲覧**

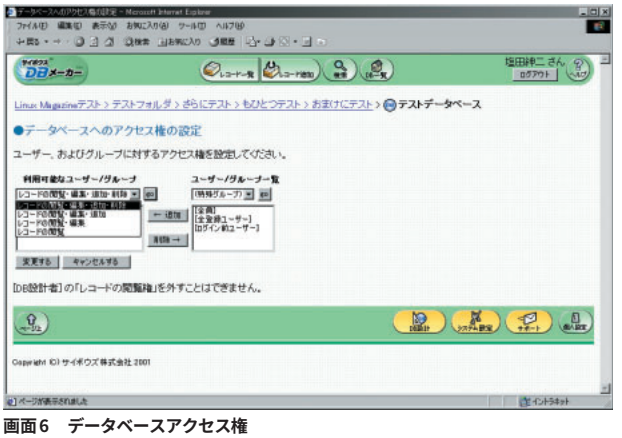
の4つのパターンで個々のユーザーまたは、グループに対して許可を行うことができる（**画面6**）。なお、明示的な禁止の指定はできないので、禁止したいユーザーやグループ以外を許可するという形で行う。

さらに、特殊なユーザーとして「全員」、「全登録ユーザー」、「ログイン前ユーザー」の3つがある。「全員」とは、「全登録ユーザー」と「ログイン前ユーザー」を合わせたものだ。

データベースの設計者（設計者として登録したユーザー。これはシステム設定で行う）は、常に閲覧とデータベース設計が可能となる。このため、アクセス権の設定やデータベース自体の削除は、データベース設計者のみが行える。



画面5　データベース定義
データベース定義は、データベース名の設定のあと、フィールドを設定するような手順でページが進行する。また、この作業もいつでも取り消しが可能だ。



画面6　データベースアクセス権
データベースのアクセス権の設定は、ユーザー／グループごとに行い、アクセスパターンごとに個別に設定が可能になっている。

これらのアクセス権設定により、たとえばあるグループ／ユーザーは、閲覧とレコードの編集は可能だが追加ができないとか、閲覧のみといった設定が行えるわけである。

なお、パターンでは上位のパターンが下位のパターンを包含するので、同じユーザー／グループを別のパターンで指定しても、最後に指定したパターンのみ有効になる。

■フィールドの設定

ここでは、個々のフィールドの属性やコメント初期値、ソートキーであるかどうかの設定を行う（**画面7**）。どのフィールドについても、

- **フィールド名**
- **コメント**
- **初期値**
- **必須項目（省略可能かどうか）**
- **ソートキー(一覧表示でその項目をキーとして、降順、昇順に並べ替え)**

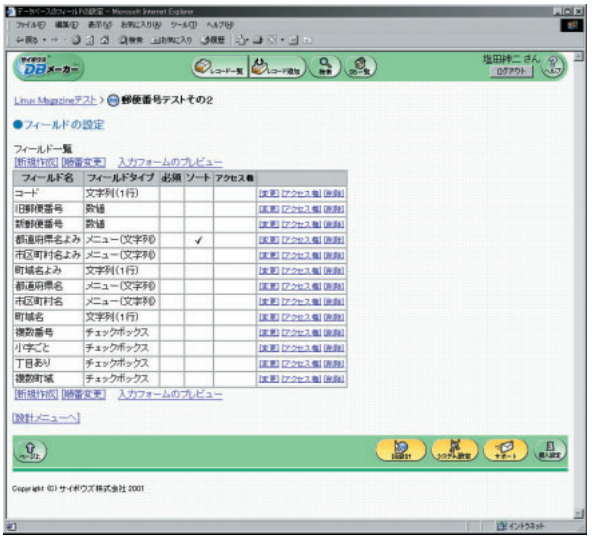
といった設定が可能で、あとはフィールドタイプごとに個別の設定項目がある（**表1**）。たとえば、数値であれば最小値や最大値、単位などを指定できる（**画面8**）。

フィールドタイプは、レコードをHTMLで表示することを考慮したもので（**画面9**）、たとえば、URLやファイル（画像ファイルならインライン表示が可能）といった指定もある。また、ラジオボタンとメニュー（文字列）フィールドタイプは、結果的には同じような固定文字列の入力となるが、それぞれフォームのラジオボタン（INPUT type="radio"タブ）とメニュー（SELECT～OPTIONタグ）に対応する。前者は項目数の少ない場合、後者は項目数の多い場合などに使い分けることができる。

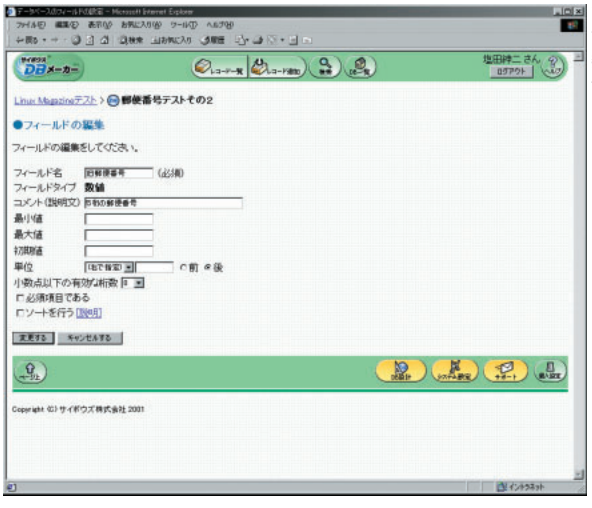
なお、個々のフィールドには、ユーザー／グループによりアクセス権を設定できる。このとき設定できるパターンとしては、

- **フィールドの閲覧、編集**
- **フィールドの閲覧**

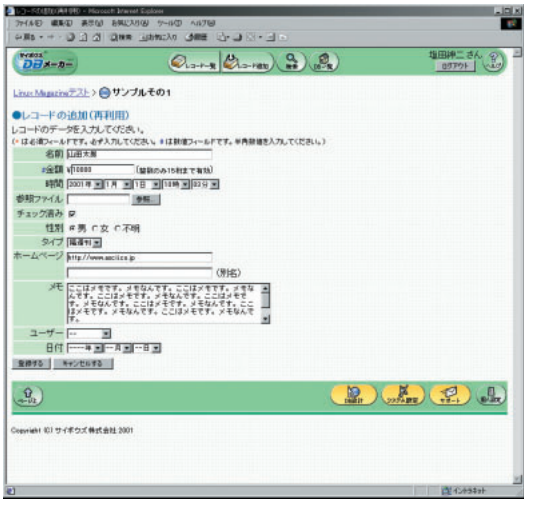
の2つがある。この2つのパターンのどれにも指定されないユーザーは、データベースへのアクセス権はあっても、フィールドを見ることができなくなる。具体的には、「このフィールドを編集する権限はありません」と表示される。また、閲覧のみのユーザーは、レコード一覧でフィールドを見ることができるが、レコードの入力、編集、追加画面での入力ができない。ただ、実際に使ってみた感想として、このような場



画面7　フィールドの編集
フィールドは定義したあとでデータの入力条件などを指定することができるが、フィールドタイプの変更は行えない。



画面8　数値フィールドの設定
数値フィールドでは、最小値、最大値、数値の単位といった設定ができる。



画面9　レコード入力画面
フィールドタイプをベースに自動的にレコード入力、編集画面が作られる。

合でも編集不可能なフィールドの内容は表示されたほうがいいのではないかと思う。というのは、たとえば顧客データベースのようなものを考えたとき、あるユーザーがメモ欄のみ編集可能となった場合、編集画面などで編集が禁止されているフィールドが見えないと、どのレコードを編集しているのかわかりづらくなる。

■絞り込み

絞り込みとは、あらかじめ定義しておく検索条件で、これを使って条件を満たしたレコードのみを一覧表示させることができる。この条件には、名前をつけていくつも登録しておくことが可能だ（**画面10**）。

条件としては、フィールドの内容に対して、

- **含む**
- **含まない**
- **等しい**
- **異なる**
- **（日付、時刻が）以前**
- **（日付、時刻が）以降**
- **（数値が）より大きい**
- **（数値が）以上**
- **（数値が）より小さい**
- **（数値が）以下**
- **チェック済み**
- **未チェック**

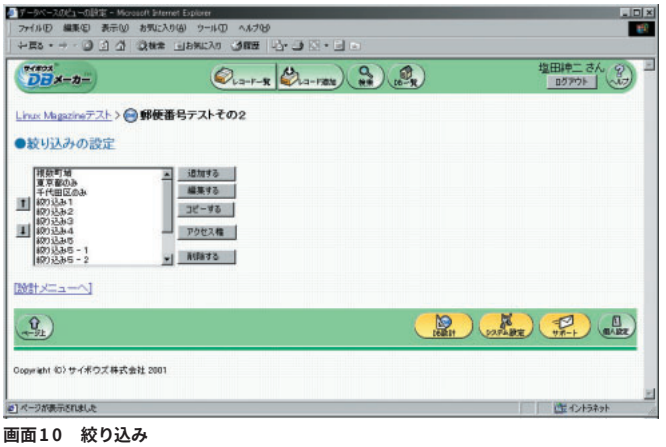
というテストが行える。また、特定のフィールドだけでなく、すべてのフィールドを対象とすることもできる。

こうして定義した条件を複数作り、これらをAND条件（すべての条件を満たす）、OR条件（どれか1つの条件を満たす）とすることができる（**画面11**）。

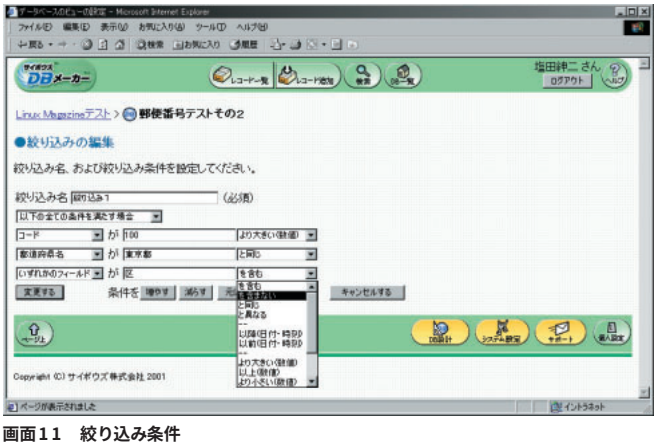
絞り込みは、定義すればレコード一覧からメニューで簡単に指定できる。なお、この絞り込みとは別に、レコード一覧には同じように条件を指定して、それを満足するレコードのみを表示する検索機能が別にある。こちらは機能的には絞り込みと同じだが、条件を保存することができない。

■ビュー

ビューは、レコードを構成するフィールドの一部を使って表示を行うもので、レコード一覧の表示方法を指定するものといえる。このビューでは、通常のレコード一覧では表示されないレコード番号（インデックス）や登録者、レコードの更新日時といったシステム側のフィールドも表示可能で、さらに



画面10　絞り込み
絞り込みやビューはいくつも名前をつけて定義でき、レコード一覧画面からメニュー形式で選択が可能。



画面11　絞り込み条件
絞り込み条件は複数指定可能で、基本的にはフィールドと値の比較でテストする。個々の条件のAND条件、OR条件が可能。

タイプ	説明	個別の設定項目など
文字列（1行）	改行を含まない文字列	入力幅、最大文字数
文字列（複数行）	改行を含む文字列	
数値	数値	最小値、最大値、単位、小数点以下桁数、単位（金額、%など）
URL	文字列だが、表示時にはリンクになる	入力幅、最大文字数、別名表示
チェックボックス	真偽値	
ラジオボタン	複数文字列から1つ選択。Formのラジオボタンを使う	ラベル文字列
メニュー（ユーザー）	登録ユーザー名から1つ選択。Formのリスト選択を使用	選択可能なユーザー
メニュー（文字列）	複数文字列から1つ選択。Formのリスト選択を使用	ラベル文字列
日付	日付	プルダウンメニュー入力の有無
時刻	時刻	プルダウンメニュー入力の有無
日付時刻	日付と時刻	プルダウンメニュー入力の有無
ファイル	クライアント側のファイル（アップロードが行われる）	受け入れ可能な拡張子、画像のインライン表示

表1　フィールドタイプ

レコード構造を変えることなく、フィールド並びを変更することができる（**画面12**）。DBメーカーでは、1つのデータベースに対していくつものビューを定義し、必要に応じて使うことができる。

たとえば、顧客リストから会社名と電話番号フィールドのビューを作ることで、電話帳とするなどの使い方をする。またビューは、前述の絞り込みや並べ替えとも併用可能で、これらを組み合わせて指定することで1つのデータからさまざまな表を作り出すことができる。

リレーショナルデータベースシステムだと、レポートジェネレータという機能があるが、それの簡易版がレコード一覧表示に組み込まれているような感じだ。

また、ビューではフィールドを表示する際のセル幅や背景色、セル内での位置、文字サイズ、文字色、文字属性（ボールド、イタリック、下線）を指定できる（**画面13**）。逆にいうと、DBメーカーでのレコードの表示は基本的には表形式で、その表示形式を設定するにはビューを作るしかない。通常、データベースではレコード内に管理を簡単にするためのフィールドなどを埋め込むことが多く、また、フィールドの並びも処理の都合で決められることが多い。このため、通常の利用はビューを使って、使いやすい形での一覧表示を作ることになるだろう。

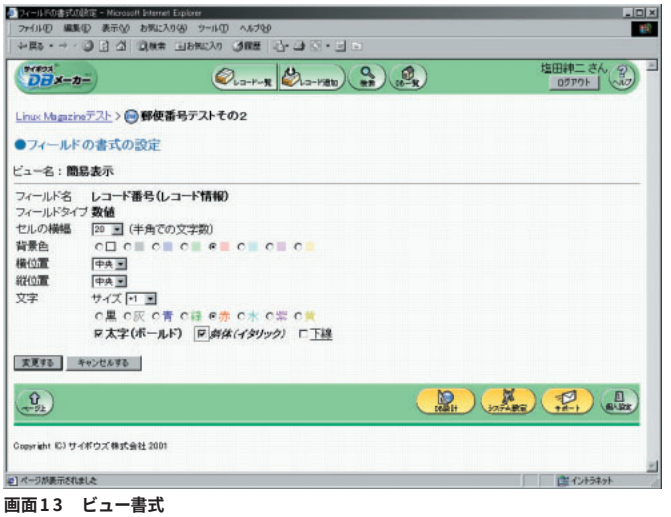
レコードを1つだけ表示する場合には、あらかじめ組み込まれているHTMLのテーブルを使った表示、またはレコードの入力、編集ページを使うしかない。

■CSVファイルの読み書き

DBメーカーは、すでに作成したデータベースに対してCSVファイルを読み込んだり、CSV形式でセーブすることができる。このCSVファイルの読み書きの機能が、DBメーカーのデータベースを外部で扱うための唯一の方法である（もっとも、CGIなので無理やりフォームを送りつければ、結果をHTMLとして受け取ることは可能である。これを外部のプログラムで行えばデータをやりとりできるが、HTMLを経由するためデータの解析が少し大変となる。Excelなどは、HTML内のテーブルをワークシートに取り込む機能などがあるので不可能なことではない）。

CSVに対するセーブは、レコード番号の有無および、先頭行をフィールド名とするかどうかといった指定が可能だ。CSVファイルは、クライアントに対してはダウンロードという形でセーブされる。つまり、Webページ上のzipファイルなどがリンクで指定してあり、それをクリックしたときのように、

画面12　ビュー
ビューはレコード一覧の表示方法を指定するもの。表示するフィールドやその並び、背景色などの指定が可能。



画面13　ビュー書式
ビューでは各フィールドごとに、背景色、文字サイズ、文字色などを個別に指定できる。

Webブラウザのファイルのダウンロードダイアログが開く。これは、HTTPのレスポンスとしてTEXT/HTML以外のブラウザで直接扱えない形式のファイルが送られてきたときのブラウザの動作である。

CSVファイルからのレコード読み込みは少々複雑である。まず、CSVファイルはデータベースのフィールドと並びが対応している必要がある。さらに、各フィールドの設定とCSVファイルの各データが矛盾しないようになっている必要がある。また、CSVファイルからレコード番号を読み取る場合、すでにデータベース内にあるレコードと同じレコード番号があれば、そのレコードが上書きされる。逆にレコード番号をCSVファイルから読み込まなければ、すべてのレコードが追加される。

読み込みのときのCSVファイルもクライアント側から送信する。つまり、クライアント上にあるファイルを指定するのである。

このような使い方の場合、クライアントマシンはWindowsマシンである可能性が高いが、DBメーカーではこのCSVファイルの文字列はシフトJISコードであることを想定している（書き出し時にはシフトJISコードで書き出される）。つまり、WindowsマシンであればExcelなどの出力をそのままDBメーカーにアップロードすればよいが、それ以外の場合にはシフトJISコードにあらかじめ変換しておかねばならない。

そのほか

そのほか、DBメーカーではシステム管理として、データベースの管理やシステム管理者の設定がある。データベース管理は、設計者や作成者、レコード数などの情報を見ることができる。また、ここでデータベースの仮運用から本運用への切り替えなどを行う。

DBメーカーでは、レコードの編集時にはタイムアウト機能付きのロックを使う。これは、編集ページを表示したときから一定時間はレコードをロックし、ほかのユーザーからの変更を禁止するが、一定時間内に更新が行われなければロックが解除され、ほかのユーザーが変更可能となるものである。なお、このタイムアウト時間はデフォルトで5分だが、データベースごとに設定することが可能だ。

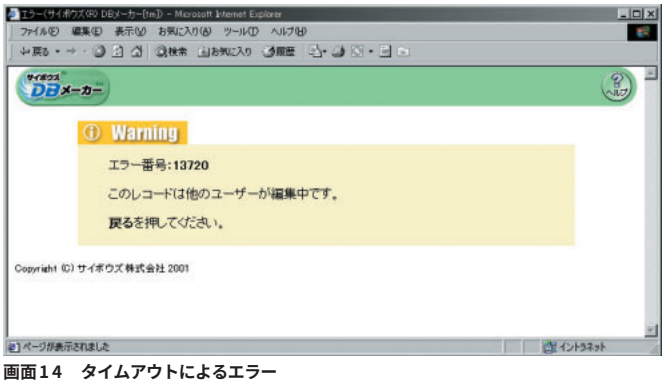
もし、この時間を過ぎたのちに変更を加えようとすると、ほかのユーザーがロックしていなければ更新が行われるが、ほかのユーザーが変更を加えたらそのページからは更新ができなくなるようになっている。

リレーショナルデータベースシステムなどでは、レコードの更新時には変更指示をまずデータベースエンジンに送り、それを記録させる。そのあと、コミットと呼ばれるコマンドを使って、データベース自体に変更を加えるようになっている。コミットを行う前であれば、変更の回復なども可能だ。

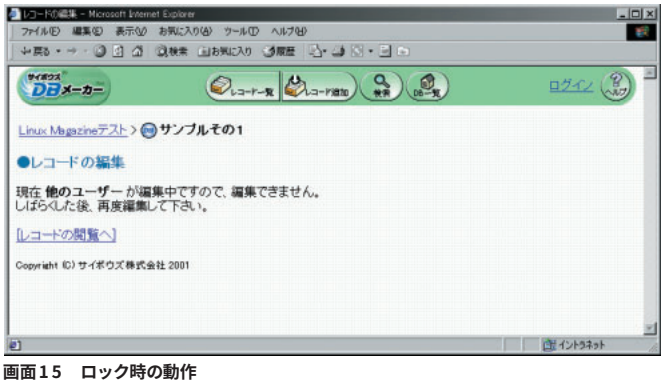
しかし、DBメーカーではロックしている間に変更を加えると、その時点でデータベースが直接変更されるようになる。このあたりが、大規模なシステムを構築するリレーショナルデータベースシステムと違うところだ。

実際、2万件程度のデータを読み込ませて簡単な検索などを行ってみたが、400MHzのCeleronマシンでも、10秒程度で結果が表示される。簡単にいえば、インターネット上のサイトをWebブラウザでアクセスしているのとさほど変わらない感じだ。SOHOといった用途であれば、何百万件ものレコードを扱うことは少ないと思われる。このような用途であれば、十分性能を発揮できるのではないだろうか（もっとも、筆者はデータベースの専門家というわけではないので、利用の場合にはご自分で評価していただきたいが……）。

また、DBメーカーのライセンスがデータベースの個数と期間というのも面白い点だ。こういったライセンスがどう受け入れられるのかも興味深い。



画面14　タイムアウトによるエラー
レコードの編集中にロックがタイムアウトし、他人が編集すると、更新時にエラーが表示される。



画面15　ロック時の動作
他人がロックしている状態でレコードを編集しようとすると警告が表示される。

隠喩としてのコンピュータ

概念マップとXML事始め

文：豊福 剛
Text : Tsuyoshi Toyofuku
illustration : hmm

いわゆる西洋占星術に興味があって、自分のホロスコープを作るときなど、昔はその手の本の天文歴を見ながら、コンパスと分度器を使って作図していたのだが、最近はネット上のサイトで誕生日と出生時刻と出生地を入力すると、ホロスコープの画像を自動作成してくれたりするので、そういうサイトを利用して知り合いのホロスコープを調べてみては、ふむふむ、この星の座相があの性格を物語っているのだな、などと分析するのが密かな楽しみだったりもする。

ところが、よく使っていたサイトのホロスコープ画像が、GIF形式からPNG形式に変わってしまったり、かなり詳細な結果を出力してくれる別のサイトが使えなくなったりしたこともあって、5月の連休の夜、ホロスコープ作成のCGIを作ってみることにした。

ホロスコープ用のデータをゲット

数年前にアナログ時計のCGIをPerlで作ったことがあったので、そのプログラムを読み直してgdグラフィックスライブラリの使い方を思い出しながら、同心円上に太陽、月、水星、その他の惑星の画像を並べてみた。ついでに、それぞれの星と星の間の角度が120度前後（トライン）だったり90度前後（スクエア）だったら、星の間に色違いの線を引いて、ホロスコープの座相がひと目でわかるようにしてみた。ここまで作ってしまうと、12ハウス分割の機能もほしくなったので、これも追加した。

このCGIは、太陽が山羊座の17度24分、とかいった具合に、星の位置をフォームで指定するインターフェイスなのだけれども、そこで使う入力データについては、とりあえずほかのホロスコープの結果を流用している。パソコンを初めて買ったときも、ホロスコープ作成プログラムを作りたいなと思ったのだが、そのときは星の暦データをひとつひとつ入力するのが手間で諦めていた。ところが、インターネットでいろいろ探していたら、1880年から2009年までの暦データを公開しているサイト（http://www.zodiacal.com/ephemerides/）を見つけることができた。

ここまで材料が揃ってしまうと、あとは、出生時刻と出生地の経緯度をもとに、星の位置を補正する処理を組み込めばいい。自分のホロスコープと現在の星の位置との座相を計算して、今週の運勢をチェックしたりもできるわけで

ある。このように、ホロスコープの作図そのものは、データさえあれば簡単なプログラミングで作れるのだが、本格的な占いにするには、星のデータをどう解釈するかがすべてであって、ここのところは、占い師の営業ノウハウということになってしまうのである。

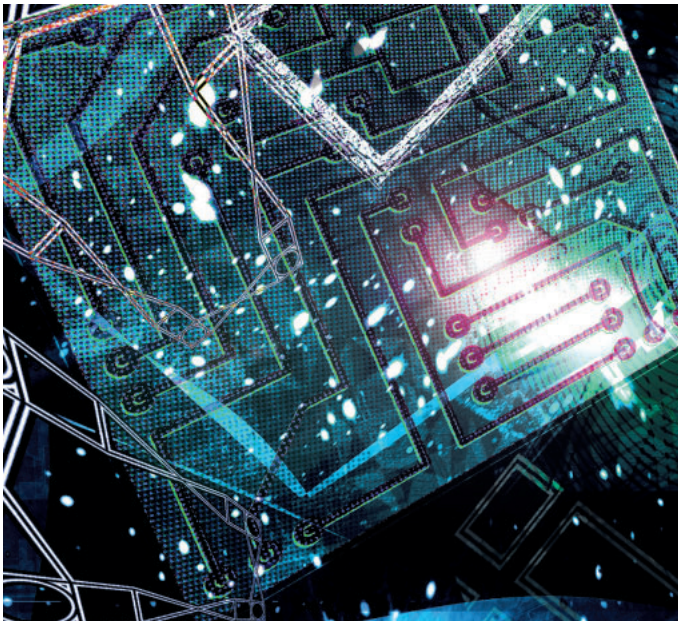
ホロスコープCGIの出力画像からは、たくさんの情報が読み取れる。生まれたとき、10個の星がどの星座に位置していたか、というだけでなく、星がどのハウスに属しているか、だとか、個々の星は他の星とどのような座相にあるか、などなどを読み取っていく。ただし、あまりに多くの情報がありすぎて、どれが重要な情報なのか判断するのが難しくなる。ホロスコープが定量的なデータだとすると、そこから、ある重要なパターンを認識し、定性的な解釈を導き出す方法を考えるのは、趣味のプログラミングとしては、なかなか面白いテーマだと思う。それに、占い師の解釈をありがたがるだけでなく、自分でソースにアクセスできるってことは、占いにおいても大切なことかなと思うのだ。

XMLはオブジェクト、オブジェクトは概念

世の中の事件や出来事と星の配置は関係している、というのが占星術の前提になっているわけだが、そうなると、何年何月何日にどういう事件が起きたのか、簡単に検索できるサービスがないかと思う。仮に、5W1Hだけを要素とする事実データの集合があれば、それに対するいろいろな抽出でもって、さまざまな目的別の年表が作れる。そういう事実データは、膨大にあるわけだから、どこか1カ所のデータベースに格納するというのは困難だ。そう考えると、年表を作るためのソースになるデータのスキーマは、ある程度共通化されているのが望ましい。というわけで、事実データのためのXMLイディオムがあったら便利かなと考えている。

去年くらいから、XMLはB2Bやマーケットプレイスのためのプラットフォームということで、盛り上がってきていることもあって、XMLを使ったプログラムを何か作ってみようかと思ったものの、けっきょくPostgreSQLやHTMLで間に合ってしまうことが多くて、いまひとつXMLに取り組むきっかけがなくて、XMLの動向のフォローだけで終わってしまっていた。

ところが、最近になって、XMLに対する認識が少し変



った。その直接のきっかけは、UMLツールで作成した図の情報をXMLデータとして外部出力できることを知ったことが大きい。UMLとは、システムの分析や開発の仕様書に使う図として最近普及してきたものであり、オブジェクト指向にもとづいてシステムをモデル化するときに使われる。UMLがオブジェクトの構造を記述するための言語だとすれば、そのUMLで記述した情報から作成されるXMLは、オブジェクトの構造（そしてその値）をそのまま反映しているわけだから、単純化すれば、XMLはオブジェクトを表現するための手段だ、と考えることができる。

それで、UMLのクラス図を眺めながら、これを概念マップの記述に転用できないかと考えていたら、西フロリダ大学の人間・機械認知研究所IHMC（Institute for Human and Machine Cognition）が公開しているJavaで開発された概念マップ作成支援ツールCmapがあることを知った（http://cmap.coginst.uwf.edu/）。たしか、ひと昔前のJDKのサンプルプログラムには、踊るデュークのアプレットなんかと一緒に、簡単な概念マップのアプレットが収録されていた記憶がある。四角で囲まれている単語が複数あって、関連する単語が線で結ばれていて、単語をドラッグすると、それに合わせて他の単語や関連を示す線も移動する、そういうアプレットだった。

このCmapでは、ひとつひとつの概念は、やはり四角で囲まれたラベルで示されるのだが、それだけでなく、概念間の関連に対しても名前が設定できるようになっている。Cmapのチュートリアルを読むと、概念を「イベントやオブジェクトにおいて知覚される規則性、またはイベントやオブジェクトの記録」として定義してあった。

概念マップとメタデータの管理

概念マップといっても、単に作図するだけだったら、Visioなんかでも十分なんだけれども、このCmapの面白いところは、作成ツールとは別にサーバ用のプログラムがあって、これと併用すると、作成した概念マップをインターネットにそのまま公開できる点だ。それで、概念マップの個々の概念には、ハイパーリンクを設定することができて、それをクリックすると、その概念についての詳しい説明文や画像などを表示させることもできる。

UMLツールとのアナロジーで考えれば、このCmapも、作成した概念マップをXMLで外部出力できる機能があれば、もっといろんな応用が広がるように思うのだ。たとえば、電総研の橋田浩一氏が展開されている大域文書修飾Global Document Annotation (GDA)というプロジェクト(http://www.i-content.org/gda/）では、文章の統語や意味に関するXMLタグの標準を作ることが試みられているのだが、GDAタグが付加されたテキストから意味ネットワークが生成できるのであれば、その意味ネットワークをCmapのような概念マップ用のツールと連携させることで、意味ネットワークをビジュアライズすることができるだろう。あるいは逆に、概念マップ上のある概念を選択すると、それに関連するテキストがピックアップできる、といったようなインターフェイスを使って、テキストの検索やナビゲーションが実現できるだろう。

XMLの応用分野といえば、ついついB2Bやマーケットプレイスといったデータ交換の領域ばかりに限定して考えてしまいがちだが、知らないだけで、実はいろんなXMLのイディオムが存在していたのだ。ただし、GDAはさまざまな文法情報を付加するだけに、これらのタグをすべて正確にタイプするのは、なかなか骨の折れる作業になりそうだ。完全とはいかないまでも、たとえばChaSenなどの形態素解析ソフトウェアを利用して、ある程度のGDAタグを自動生成できないものかなと思う。

自然言語処理の専門家でもないのに、GDAや概念マップに興味があるのは、たぶんテキストそのものの処理というよりも、そのテキストあるいは複数のテキストにおけるメタデータの処理や操作が、これからますます重要になっていく予感がしているからだ。

この連載では、ここしばらく地域通貨とP2Pが関心の中心にあったのだが、地域通貨における参加者の帳簿管理にコンピュータを使うというだけでなく、参加者の求めるモノやサービスと提供できるモノやサービスのマッチングや検索を実現するうえで、漠然とXMLを利用するといいことがありそうな気がしていたのだが、いまひとつXMLを使う必然性がつかめないでいた。しかし、的確なデータを自動的に探し出せるような、ある種のエージェント機能が求められるとしたら、そのようなエージェント機能の実現するためのメタデータを提供するのにXMLは役立つように思うのだ。

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。

商売は楽じゃない

文：安田幸弘
Text : Yukihiro Yasuda

MS、ソースを公開、だけどGPLは批判

マイクロソフトが「シェアードソース」という制度で、Windowsのソースコード公開の幅をほんの少し広げたという。アマチュアのハッカーには縁がなさそうな話だけれど、これもオープンソースの影響なんだろう。限られた範囲での厳格な条件付き公開とはいえ、マイクロソフトもソースコードの形で情報を提供することの意義を認めたということになるのだろうか。

もちろん、マイクロソフトが始めた「シェアードソース」は、オープンソースとはまったく別もので、ソースコードは完全にマイクロソフトの統制下にある。マイクロソフトのライセンス・ビジネスを考えれば、ソースコードを公開するかどうかは、どちらかといえば二次的なことなのかもしれない。彼らにとって、要はソースコードに含まれる知的所有権が保護されることが重要なのであって、「シェアードソース」はこれまでのマイクロソフトのライセンス・ビジネスの枠組みをいささかも変更するものではない。

それでも、やっぱりマイクロソフトにとって、ソースコード公開の圧力というのは、あまり愉快なことではないのだろう。マイクロソフトの幹部の口からは、何かにつけてオープンソースはダメだとか、GPLはけしからんといった言葉が聞こえてくる。

独占的なライセンスでビジネスをやっている彼らが反独占的なライセンスに反発したくなる気持ちはよくわかる。何しろ、GPLのフリーソフトウェアには「買収」という必殺技がかけられないのだから、さぞかし苛立たしいことだろう。もっとも、無償のブラウザでライバルを蹴落とすようなビジネスをやるんだったら、無償のソースコードで独占企業のシェアを食うというビジネスも認めるべきだとは思うけどね。

オープンソース・ビジネス

それはともかく知的所有権を前提とする現在のソフトウェア・ビジネスのスタイルは、マイクロソフトが確立したものだといっても過言ではない。無政府的な一種のフリーソフトウェア状態にあった初期のソフトウェアの世界の中で、ソフトウェアに対する知的所有権という考え方を苦労して広めたことで成功したのが彼らである。それだけに、マイクロソフトにとって知的所有権を無力化するようなGNUの挑戦は、彼らのビジネスの根幹にかかわる大問題だ。

マイクロソフトの幹部は、オープンソースはビジネスとしてうまいやり方ではないという主張を繰り返しているのだそうだが、ソフトウェアを商品だと考えるビジネスマンにとって、マイクロソフトの主張は合理的なのかもしれない。正直言って、10年前、ぼくはGPLのソフトウェアで本当にビジネスが成立するなんて思ってもいなかったし、たぶん誰もそんなことを信じてなんかいなかっただろう。

しかし、世の中のすべての人がソフトウェア・ビジネスにかかわっているわけではない。商品ではないソフトウェアもあり、ソフトウェアの権利でビジネスをするつもりがない人も多い。そして、Linuxのように商品ではないソフトウェアが実用的に使われるようになってくると、「同じことができるのに、何でわざわざライセンスで縛られた商品を使わなければいけないのか」という素朴な疑問も湧いてくる。アメリカのように納税者意識が高い国では、下手をすると「公共機関は無料のソフトを使うべきだ」なんて訴訟が起きないとも限らない。ソースコードには知的所有権なんて認めるべきじゃないというGNUの言い分が、どのような形にせよ、社会的な合意を形成することだけは阻止したいと思うのは当然かもしれない。

その反面、GNU的、オープンソース的には、マイクロソフトと反対に、ソフトウェアには知的所有権が必要だという迷信を打ち砕き、知的所有権に縛られない自由なソフトウェアがいかに優秀で使いやすいかという社会的合意を広げていきたいわけだ。そのためには、ソフトウェアそのものを縛らなくても、ちゃんとソフトウェア・ビジネスが成立し、経済的な動力になるということを示すことが重要なポイントだった。企業の影響力が強まることへの警戒感は残るものの、ビジネスセクターが動くことによって、それまでボランティアベースで開発されてきたフリーソフトウェアに、豊富な資金や人材、技術などが投入されることになり、多くの人々がオープンソースのビジネスに参入している。最近は、オープンソースでのビジネスを展開する企業も増えている。ボランティアベースでは困難な高度なソフトウェアの開発に、こうしたビジネスセクターの参入したことが、オープンソースをマイクロソフトさえ脅かす勢力に育てたわけだ。

Eazel死してもNautilus死せず

もっとも、そうしたビジネスの中には成功もあれば失敗もある。先日、GNOME上のファイルマネージャであるNautilusを開発していたEazelが倒産したというちょっと残念なニュースが流れた。どうやらドット・コム・バブルを過信した経営戦略の問題らしい。

ドット・コム的なオープンソース・ビジネスは、ビジネスには向かないと主張するマイクロソフトを喜ばせそうなニュースだが、しかしEazelがマイクロソフト流のシェアード・ソースとやらでライセンス・ビジネスを展開していたらどうだったろうか。倒産云々の前に、そもそもオープンソースの世界でなければ彼らの挑戦そのものが成立しなかっただろう。うまいやり方ではなかったのだろうが、オープンソースが新しいビジネスを可能にしたという事実のほうが、ずっと面白いと思う。

そのうえ、さらに重要なことは、ドット・コム・ビジネスとしてのEazelの失敗は、オープンソース・プロダクトとしてのNautilusの失敗を意味するものではないということ。Nautilusがオープンソースだったおかげで、開発元のEazelが倒れてもコードが債権団に持ち去られることもなく、オープンソース・コミュニティのものであり続け、これまでと同じように使い続けられる。

だからEazelが活動を停止してもNautilusは健在で、現在も開発が続けられているわけだ。古くからのパソコンユーザーなら、きっと1つや2つは、気に入っていたソフトの開発元が消滅したり、開発完了で悲しい思いをした経験を持っている思う。しかし、フリーソフトウェアなら開発元が消えてなくなろうが、サポートを停止しようが、そのソフトが失われる心配はない。

Eazelの失敗は、マイクロソフト的にはオープンソース・ビジネスの脆弱性を証明するものかもしれない。それと同時にオープンソース・コミュニティにとっては、オープンソースやGPLで保護されたソフトウェアが、特定の企業や団体の影響を受けにくいことを証明するものかもしれない。

いずれにしても、パソコンのビジネスなんてたかだか数十年の歴史しか持たないニュー・ビジネスだ。GNUもオープンソースも、そしてシェアードソースとやらも、それぞれのビジネスのスタイルに合わせてさまざまな可能性がある。オープンソースには、果敢なチャレンジを続けてもらいたいものだ。

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。

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

4・22 Samba 2.2リリース
4・24 Informixデータベース事業をIBMに売却
4・26 PS2用Linuxキット商品化発表
5・7 Turbolinux、Linuxcare合併計画を撤回
5・10 Windows XP米国出荷は10月25日と発表

最近、コンパックのiPaq H3630というPocket PCを買った。海外のLinux関係のイベントにいくと必ずといっていいほど、このマシンでLinuxを動かして、いろいろなシステムをデモしている。それで、なんだか、持ってないとちょっと肩身が狭い感じもしていて、ずっと狙っていたのだが、結局、日本語版が発売されてようやく入手した次第。あちこちに原稿を書いたあと、ゆっくりとLinuxでも入れてと思っているのだが……。

XPは予定通り年内というウワサ

Microsoftは、次期WindowsであるWindows XPのβ版を公開するなどして、出荷の準備を進めているらしい。原稿執筆時点の予定だと、米国で10月には出荷されるのだとか。

今回のXPはWindows 2000のマイナーチェンジに過ぎないという説もある。たしかにGUIなどは大きく変更されるが、基本部分はWindows 2000であり、本来2000が持つはずだったWindows 98との互換性を組み込んだだけとも言われているからである。そもそも、Windows 2000の開発が難航したのは、これをWindows 98の後継OSとするために、従来のWindows 98からのアップグレードが可能なよう、ある程度の互換性を持たせる部分の開発に手間取ったというのだ。しかし、あまりに開発が遅れたため、Windows 2000を後継OSとすることを諦め、その部分を除いて出荷したのだと。

Windows 98マシンからのアップグレードを考えると、98用デバイスドライバなどがXP環境で動作できるようになっていなければならない。Windows 98では、デバイス関係がかなり込み入っていた。たとえば、キーボードやマウスは、USB接続もPS/2接続もあって、どちらもアプリケーションからは同じように見えるように作られている。しかも、そのうちのいくつかのアプリケーションは、ハードウェアを直接覗いたり、BIOSのデータ領域を見て動くなんてシロモノ。Windows Meまでは、ハードウェアそのものをドライバを使って仮想化していたのだが、そのドライバの下にちゃんとしたドライバを持つUSBキーボードがつながるなんておかしな状態になっていたのである。

デバイスやドライバが入り組んだ状態になっているうえ、本来Windows 2000でも使えるデバイスドライバモデルとしてWindows 98で導入したWDM（Win32 Driver Model）は、結局のところ、Windows 2000とはバイナリレベルでの互換性を持たせることができなかった（Windows 2000では仕様が変更された）。なので、サードパーティから見れば、Windows 2000用のドライバは作り直し。そのためか、いまでもWindows 2000に対応していない周辺装置は結構残っている。

Microsoftとしても頭が痛いのは、市場に大量に出回っているくせに、ユーザーサポートがほとんど期待できない、ノーブランドに近い周辺装置の数々である。秋葉あたりで売っている周辺装置は、メーカー名や製品名が書かれているにも関わらず、連絡先がなにも記載されていないものがある。もっとも、ドライバなんかは、内蔵している半導体メーカーが作っているものがそのまま入っている（ときどき、OEM向けの注意書きがそのまま入っているフロッピーを見かけることがある）。こうしたデバイスに関しては、ある程度Microsoftも対応せざるを得ないわけで、汎用のドライバを用意する必要がある。これもOS開発における大きな負担となっている。

だが、XPがWindows 98の後継OSと

して登場することで、ようやく、一般向けのOSが完全に32ビット化する。Windowsの最初のバージョンが1985年に登場して以来、16年もかかっているのだ。その意味では、今回がWindows最大のバージョンアップともいえる。さて、これから10月の出荷まで、Microsoftはいろいろとキャンペーンを始めるはずだが、いったいどういうことをやるのか？そして、日本での発売日当日、一体何人が夜中、秋葉原にXPを買いに行くのだろうか？　なんかそろそろ、そういうのに並ぶのって、カッコ悪いような気がするんだけど、みんなカメラ向けたら、顔隠したりしてね。それとも、もう深夜販売はしないのかな？

IntelとAMDの競争は激化

さて、変わってCPUのほうだが、Intelは、AMDのシェア拡大を許してしまった感じだが、Pentium 4を値下げするなど、反撃も行いつつある。そして、AMDの次のターゲットは、モバイル系である。Intelは、Crusoeを相手に、日本メーカー数社の採用を許すという失敗をした。そこにAMDは、付け入ろうと、モバイルAthlon／Duronを投入する。しかも、今度のAthronは、Athron 4である。名前からしてPentium 4と対抗しそうな名前。しかも、Pentium 4のモバイル投入はまだ先になる予定だ。そこにIntelが持ち込もうとしているのが0.13μで作るPentiumIIIである。プロセスルールを小さくすることで、より低消費電力を狙い、さらに2段階しかなかったSpeedStepの段階を多くする。

しかし、名前だけを見ると、PentiumIIIよりAthlon 4のほうが良さそうに見える。何も知らないユーザーからすれば、IntelがモバイルPentium 4をしてようやく追いついたようにしか見えないかも。しかし、問題は、こうしたユーザーの心理をメーカーがどれだけ気にするかであろう。Compaq／HPがモバイルAthlon 4採用のノートPCを出荷するようだが、日本メーカーがどう動くか？　Crusoeに続いてモバイルAthlon 4も採用するのかどうか。勝負の分かれ目は、日本のノートPCメーカーの動向って気がしている。

デスクトップ系は、1.7GHzが登場して、Pentium 4も人気が盛り上がってきた感じがあるのだが、同CPUが使うRD-RAMメモリの開発元であるRambus社はちょっと雲行きが怪しくなってきた。というのも、Rambusが取得したDDRメモリが引っかかる特許についての裁判で、Infineonの主張が通った判決が米国の連邦地裁で出てしまったのである。

この話、Rambus社がメモリの標準化団体であるJEDECに一時的に参加していたとき、適切な情報開示を行わず、その後脱退したことが原因。つまり、標準化に向けて各社が討議するとき、自社の技術や特許との関連を申告しなければならなかったのに、それをせず、特許を設立させたわけだ。Rambusは、このほか、Micronなどとも同様の訴訟を行っており、これが影響する可能性はある。また、日立や東芝といった日本メーカーとは、すでにDDRメモリ製造に対して特許のライセンス料を取る契約を結んでしまったが、特許そのものが問題となると、この契約も見直しとなる可能性もあるわけだ。

そうした動きが、RD-RAMやDDR-DRAMの価格やメーカー戦略にどう影響するのか？　それは、ひいてはRD-RAMを使うPentium 4対DDR-DRAMのAthlonという対立にも影響してくるわけだ。

PS2でLinux、申し込みました？

かねてより署名活動などが行われていたソニーのPlayStation 2用Linuxキットがとうとう発売になった。しかし、オンライン販売のみで、発売開始後8分で2000台が完売したのだとか。そのうえ、キャンセル待ちが3500人もいるらしい。筆者も当日、ちょっとサイトを覗いてみようかとアクセスしてみたが、ページが開けないまま、売り切れとなってしまい、3500人の中の1人となってしまった。

中には、Linuxに興味はないけど、買ったら、オークションで高く売れるかもなんてよこしまな気持ちで買った人もいるのだろうけど、これだけ集中するとは、Linux人気いまだ衰えずって感じもする。Linuxはインストールが面倒でなんていっていた人も、PS2で簡単に動くのなら、試してみようって気にもなるだろうし、PS2用にさまざまなソフトが登場（もちろん、グラフィックデバイスも使い放題）すると、ちょっとしたブームになるかも。それに、飽きてもゲームできるしね。Microsoftも対抗してXboxでWindows XPキットを発売、任天堂もGAMECUBEにMac OS Xを搭載、なんてことはないか。

初期ロット2000台が8分間で売り切れたPS2は、追加出荷が行われることになった。
http://www.jp.playstation.com/linux/

正しきタコへの道

初級Linuxer養成講座

シェルスクリプト入門〜繰り返し処理

前回、シェルスクリプトでの「変数」の使い方について説明したのは、Linuxのコマンドラインで、同じ処理を自動的に繰り返す方法をマスターしてもらいたかったからだ。GUIのプログラムが不得意とするこのような処理を使えるようになれば、Linuxの「ありがたみ」が実感できるようになるだろう。

文：竹田善太郎
Text : Zentaro Takeda

ADSLが自宅に敷設されて1カ月ほどたった。性能については、やはり、インターネット全体が混雑している時間帯には、目立って低下するのが実感できるが、調子のよいときには、ほぼスペックどおり、ダウンロードで約1.5Mbps（160Kバイト／秒）の性能が出ている。サービス開始からまだ日が浅いため、不意のメンテナンスや障害の発生で、なんの予告もなく接続が切られてしまうことも頻繁にある。だから、**常時接続**とはほど遠い状態ではあるのだが、少なくとも、メガバイト単位のファイルでも、何の抵抗もなくダウンロードできるようになったのはありがたい。

ところで、**ADSL導入の副作用**として、最近、**音楽CDの購入枚数が増えて**しまった。ブロードバンドネットワークが普及すると、オンデマンドビデオなどの映像コンテントをネットワーク経由で配信できる、というような話ばかりが報道されているのだが、現実に見ることのできる映像コンテンツは、どこも**映画の予告編**や**アニメビデオ**ばかりで、半日もしないうちに飽きてしまった。中には「NASA TV」のような、リアルタイムの良質な映像配信もあるのだが、仕事をしながら見続けるというわけにはいかない。また、ファイルのやりとりなどを始めると、さすがのADSLでも映像がとぎれることになる。

ネットワークのストリーム配信に関しては、以前から**ラジオ配信**をよく利用していたのだが、ISDNを使っていたときは、たとえ16kbpsや32kbpsの低ビットレートのストリームであっても、受信中に音がとぎれることが多くて、かなり不満を持っていた。ADSLになってからは、これがかなり快適に受信できるようになった。120kbps以上の高ビットレートのストリームを受信しているときに、大量のファイルのダウンロードを行ったとしても、音声がとぎれるようなことはない。

そんなわけで、主にドイツやアメリカのクラシック音楽専門のラジオ局を**掛け流し**にするようになったのだが、その**副作用**として、放送されたCDをインターネットの通販サイトで購入する頻度が**激増**してしまったのだ。放送中の曲目の紹介を部分的にでも聞き取ることができれば、メモをとる必要なく、直接、通販のサイトで検索して、その場で購入できてしまう。ちょっと**安易すぎる**な気もするのだが、自分としては、これこそが**ブロードバンドインターネットの正しい利用法**なのだと信じている。音声の配信だけなら、ネットワークに不要な負荷をかけることもないし、**リアルの世界**での経済活動にも貢献しているわけだから、よいではないか。

繰り返し処理とは

パソコンを含めて、コンピュータ（計算機）という装置が唯一得意とするのは、**同じ作業を繰り返し行うこと**である。人間にはとうてい不可能な速さで、単調な計算を膨大な回数繰り返すことができるのだ。最近では、コンピュータを**計算の道具**として使うのは、表計算ソフトで領収書の集計するような場合しかない、というユーザーも少なくないかもしれないが、たとえ、ワープロソフトやメールソフトしか使っていない場合でも、パソコンの内部では**文字**も単なる**数値データ**として扱われていて、それを**計算**する処理が繰り返し行われているわけだ。

このような繰り返し処理は、ユーザ

ーが直接行わせることもできる。**プログラミング言語**を使って**プログラム**を作成することで、同じような計算を繰り返し実行させることができるのだ。しかし、最近ではよほどの**マニア**でもない限り、パソコンでプログラミングをする人は少ないだろう。Linuxを使っているユーザーでも、日常的に自分でプログラムを書いている人は、おそらく少数派なのではないだろうか。

しかし、ごく簡単なプログラムであっても、自分で書けると便利なことがある。**同じような外見のデータを大量に処理する必要**が生じたような場合だ。たとえば、次のような場合を考えてみよう。仕事の取引先から、ある資料をテキストファイルで受け取ったとする。そのファイル名が次のようになっていたとする。

```
shiryou_1.doc
shiryou_2.doc
shiryou_3.doc
      :
      :
```

このような感じで、たとえば**100個のテキストファイル**を受け取ったとする。このファイルをWindowsのアプリケーションで開いて使いたいと思うのだが、**拡張子が「doc」**になっているので、ファイルを開こうとすると**Word**が起動してしまい、いちいち「テキストファイルとして開く」というような指定をするか、あるいはテキストエディタを起動してから、**「開く」ダイアログボックス**を開けて、1つずつファイルを読み込む操作をすることになってしまう。

これでは面倒なので、**ファイルの拡張子をすべて「txt」に変更**しようと思うのだが、たとえば、Windowsのエクスプローラで100個ものファイルの名前を変更しようとすると、**気が遠くなる**ほど面倒な作業になる（**画面1**）。かといって、Linuxのコマンドラインで、

```
$ mv shiryou_1.doc shiryou_1.txt
$ mv shiryou_2.doc shiryou_2.txt
      :
      :
```

と入力し続けるのもさらに面倒な作業だし、これを1つのミスもなく行うのは**至難の業**としかいいようがない。コマンドラインの操作にちょっと慣れてきた人だと、

```
$ mv shiryou_*.doc shiryou_*.txt
```

のように実行できないかと考えるかもしれないが、シェルのコマンドラインは**そこまでは賢くない**。

このような繰り返し作業こそ、コンピュータがもっとも得意とする分野のはずなので、本来なら、簡単なコマンド操作だけで、このような処理を実行できる仕組みを持たせるべきだと思う。

実は、Windowsのコマンドプロンプトでは、

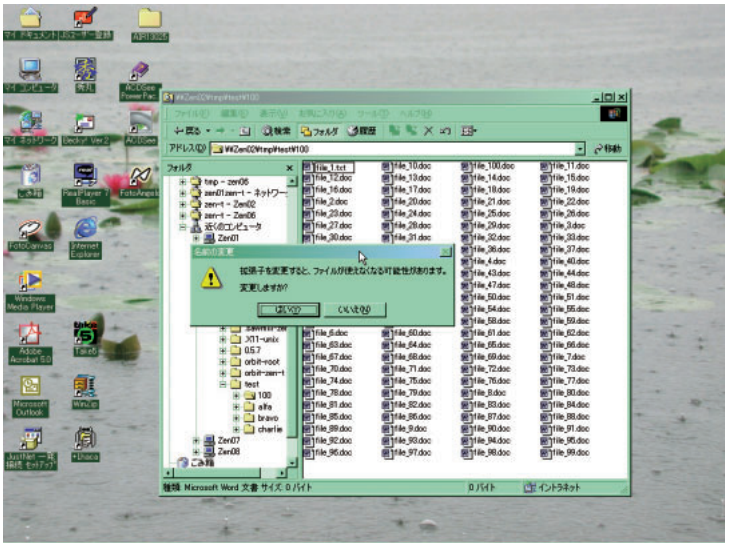
```
C> ren shiryou_*.doc shiryou_*.txt
```

とすることで、一括変換ができるのだが、なぜかLinuxのmvコマンドではできないのだ。

結局、どうやってこの問題を解決するかといえば、**プログラムを書く**しかないのである。プログラムといってもそれほど大げさなものではない。たかだか2〜3行の**シェルスクリプト**を書けばよいだけだ。また、プログラムだからといって、「テキストエディタを起動して、シェルスクリプトのテキストファイルを作成して、ファイルのパーミッションを変更して……」、というような面倒な作業は必要ない。簡単なプログラムなら、**コマンドラインから直接入力して実行**できるのである。

コマンドラインでの繰り返し処理

話は前後するが、先ほど例のように、大量のファイルの名前を変更するような場合、**図1**のような処理を行うこと



画面1　100個のファイルの名前を変更するGUIのOSがもっとも不得意とする作業なのだが、だれもがこのような仕事を手作業で行った経験があるはずだ。

になる。このような処理は、プログラミング言語の入門書では、必ずといってよいほど登場するのだが、このような繰り返し処理を行う場合、処理の途中経過を記憶させたり、何回目の処理を行っているかを記憶させるために**変数**が必要になってくる。前回、シェル変数について説明したのは、実はこのためだったのだ。

bashで繰り返し処理を行う場合は、**for**というシェルの内部コマンドを利用する。正確には「for」は「内部コマンド」というよりは、「シェルスクリプト」というプログラミング言語の「制御構造」を表す構文要素なのだが、そういう面倒な話はここでは忘れてしまってよい。bashのforコマンドは、次のように利用する。

```
$ for i in 1 2 3 4
```

上のコマンドラインを入力すると、コマンドラインのプロンプトが次のように変化する。

```
>
```

これは、forコマンドを使って繰り返し実行したい処理の内容（すなわちコマンドライン）を入力せよ、という意味のプロンプトである。ここで、実行させたいプログラムを開始する**おまじない**として、doという単語を入力する。

```
> do
```

すると、さらにプロンプトの「>」が表示されるので、繰り返し実行させたいコマンドラインの本体を入力する。

```
> echo hello
```

またプロンプト「>」が表示されるので、プログラムの終了を指示するための**おまじない**として、doneという単語を入力する。

```
> done
```

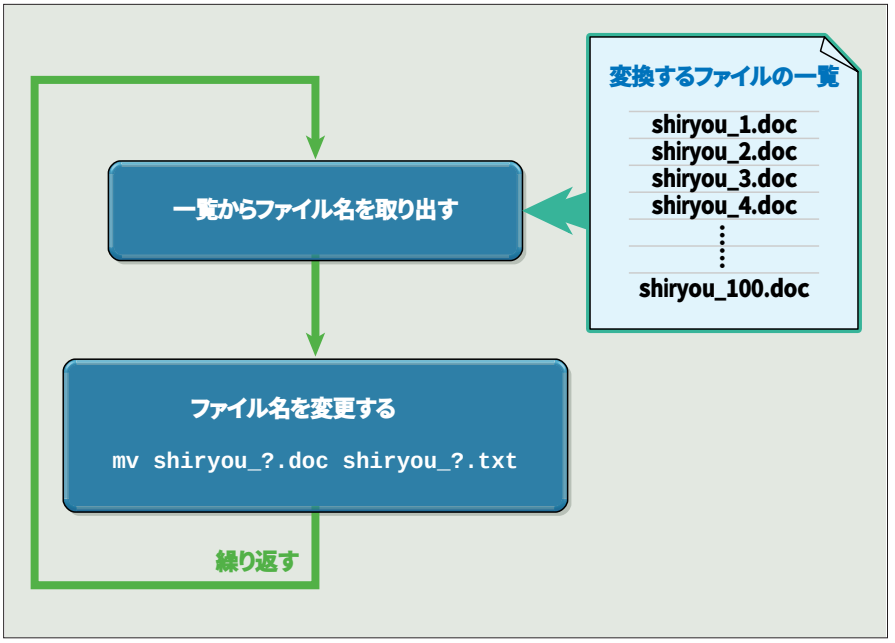


図1　複数のファイルの名前を変更する処理
作業の手順をこのような「流れ図」に表すと、全体が「輪」（ループ）のような形をしているのがわかるだろう。このため、繰り返し同じ処理を行うことを「ループ処理」と呼ぶことも多い。

すると、「hello」という文字列が4回繰り返して表示されるはずだ（**画面2**）。

上のコマンドの**意味**をもう少し詳しく説明すると、「変数iに、1、2、3、4の値を代入しながら、『do～done』の間のコマンドを繰り返して実行せよ」ということになる。1回目の実行では、変数iの値は「1」に、2回目の実行では「2」に、といった具合に、4回実行することになる。変数にどのような値が代入されているかを確かめてみよう。

```
$ for i in 1 2 3 4
> do
> echo $i
> done
1
2
3
4
```

このように、変数iの内容をechoコマンドで表示させてみれば、1、2、3、4という値が順番に代入されていることがわかる。

ところで、for～の部分で変数名を指定するときに**「$i」ではなく「i」**となっているのを不審に思う読者もいるかもしれない。**シェル変数なら先頭に$がつく**のが決まりではないかと思うだろう。筆者自身も不思議に思ったことがあるし、今でもときどき間違えて、

```
for $i in 1 2 3 4
```

のように入力してしまって、

```
bash: `$i': not a valid identifier
```

という、**わけのわからないエラーメッセージ**を突き返されること

がしばしばある。

どうしてこのようなことになっているかを説明しようとすると、話が長くなってしまうのだが、要するに、シェル変数に値を代入するときには変数名の先頭に「$」を付けないのと同じことである。ともかくforコマンドの使い方については、**こういう決まりになっている**と覚えてしまうしかない。

```
for $のつかない変数名 in ××
> do
> コマンド（変数を使うときは頭に$をつける）
> done
```

という構文は、丸暗記してしまうか、あるいは**付箋紙に書き留めてディスプレイの脇に張っておく**ことをおすすめする。

もちろん、do〜doneの間で実行するコマンドの中で変数の値を呼び出すときには、通常通り、「$i」のように先頭に$をつける。

●変数には何でも代入できる

ところで、前回にシェル変数について説明したときにも述べたが、シェル変数には数値、文字列、プログラムなど、なんでも代入できるということを覚えているだろうか。forコマンドを使った繰り返し処理の場合でも、変数にはどんな値でも代入できるのだ。たとえば、次のような繰り返し処理ができる。

```
$ for i in alfa bravo charlie
delta
> do
> echo $i
> done
alfa
bravo
charlie
delta
```

●do〜doneの間では何でも実行できる

いままでの例では、繰り返し実行するコマンドとして、bashの内部コマンドの一種である「echoコマンド」しか使っていなかったが、ここでは実際には**どんなコマンドでも実行できる**。また、**複数のコマンドを並べることも可能**だ。

```
$ for i in alfa bravo charlie
delta
> do
> mkdir $i
> touch $i/$i.txt
> done
```

上のコマンドを実行すると、カレントディレクトリに「alfa」、「bravo」、「charlie」、「delta」という4つのサブディレクトリが作成され、それぞれのディレクトリの中に「alfa.txt」、「bravo.txt」、「charlie.txt」、「delta.txt」というファイルがそれぞれ作成される（**画面3**）。

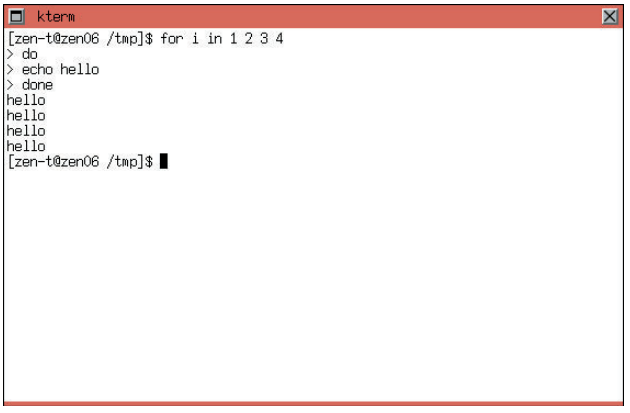
大量のファイルの名前を変える

さて、forコマンドのだいたいのようすがわかったところで、最初の問題、すなわち、**「shiryou_＊.doc」という名前の100個のテキストファイルの拡張子を「txt」に変更する**方法を考えてみよう。

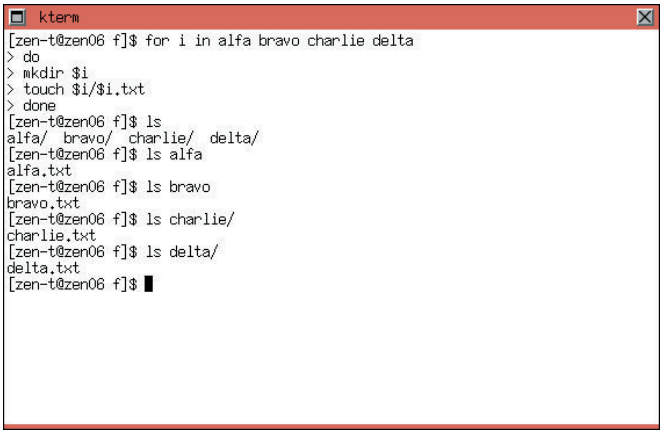
まず、問題のファイルが置いてあるディレクトリに移動して、最初のfor〜コマンドを入力する。

```
$ for i in *.doc
```

ここで、「for i in shiryou_1.doc shiryou_2.doc…」とファイル名をいちいち並べる必要はない。ワイルドカー



画面2　forコマンドの使い方
繰り返し処理の「練習」では、このように同じ文字列を繰り返し表示するという例が使われる。なんの意味もない「プログラム」だが、これをマスターすればプログラミングの半分はマスターできたといってもよい。



画面3　複数のディレクトリとファイルを作成する
do〜doneの間には、何行でもコマンドを追加することができる。複数のファイルに対して、さまざまな処理を繰り返すことも可能だ。

ドを使ってファイル名のパターンを指定すればよいのだ。あとは、指定したパターンにマッチするファイルをシェルが見つけ出し、この位置にすべてのファイル名を並べたのと同じ処理を行ってくれるのだ（**図2**）。

次に、実際のファイル名変更を行うコマンドを入力するわけだが、ファイルを移動したり削除したり内容を変更するようなコマンドを実行する前には、いきなりそのコマンドを実行するのではなく、どのようなコマンドが実行されるかを**echoコマンドで確認**しておいたほうがよい。

```
> do
> echo mv $i $i.txt
> done
```

すると、次のような文字列がずらっと表示されるはずだ。

```
mv shiryou_1.doc shiryou_1.doc.txt
mv shiryou_2.doc shiryou_2.doc.txt
mv shiryou_3.doc shiryou_3.doc.txt
        ⋮
```

ここでは、echoコマンドで文字列を表示しているだけなので、実際のファイルの名前は変更されていない。

表示された文字列（実行しようとたくらんでいたコマンド文字列）を見ると、「××.doc」というファイルを「××.txt」に変えているわけではなく、「××.doc.txt」というように、.txtを末尾に追加しているだけである。Windowsのアプリケーションは、ファイル名の一番末尾の「.××」という文字列を「拡張子」と見なすので、これでも**当初の目的の大半は達成**できている。つまり、このファイルのアイコンをクリックすれば、Wordではなくテキストエディタが開くようになるわけだ。

もし、これでもかまわないということなら、先ほどのコマンド列の「echo」を取り除いて、改めてforコマンドを実行する。

```
$ for i in *.doc
> mv $i $i.txt
> done
```

どうしても「shiryou_1.doc.txt」ではなく「shiryou_1.txt」でなければ困るということであれば、次のようにforコマンドを実行することになる。

```
$ for i in *.doc
> mv $i `echo $i | sed s/doc$/txt/`
> done
```

ちょっと複雑すぎてわからないというのであれば、ここは読み飛ばしてもらってかまわない。ここでは、「`」（バッククォート記号）を使って、コマンド列の実行結果をコマンド中でさらに利用するという、**そこそこに高度な技**を使っているのだ。この方法については、あらためてくわしく解説するが、シェルのコマンドラインやシェルスクリプトの中で「`」で囲まれた部分は、その文字列の内容をコマンドとして実行して、その結果として出力された文字列で置き換えられるのである。たとえば、コマンドラインで次のようなコマンドを実行する場合を考えてみる。

```
$ echo "echo hello"
$ echo `echo hello`
```

1番目のコマンドラインを実行すると、「echo hello」という文字列がそのまま出力される。しかし、2番目のコマンドラインでは、**`echo hello`の部分をコマンドとして実行した結果、すなわち「hello」という文字列**が出力される（**画面4**）。

同様に、先ほどのforコマンドの例の中の「mv $i `echo $i | sed s/doc$/txt/`」という行は、**`echo $i | sed s/doc$/txt/`の部分が、これを実行した結果、すなわち、iの内容をsedコマンドで処理して、末尾の「doc」を「txt」で置**

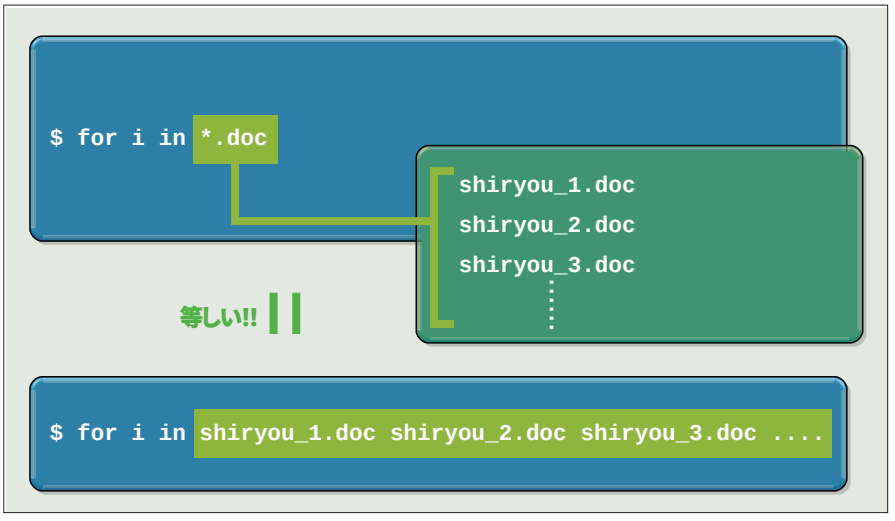


図2　複数のファイルを「ワイルドカード」で指定する
シェルのコマンドラインでは、ワイルドカードを使ってファイル名を指定すると、コマンドラインを実行する前に、その部分に該当するファイル名を「並べて書いた」のと同じ効果がある。

き換えた結果に置き換えられる。たとえば、変数iの内容が「shiryou_10.doc」だった場合、「echo "shiryou_10.doc" | sed s/doc$/txt/」の結果、すなわち「shiryou_10.txt」に置き換えられ、最終的にこのコマンド行は「mv shiryou_10.doc shiryou_10.txt」になって、目的の結果を得られるようになるのだ。

大量の画像ファイルを変換する

forコマンドの利用例として、もうひとつ、大量の画像ファイルの形式を変換する方法を紹介してみよう。

たとえば、BMP形式の画像ファイルが大量にあって、これをすべてPCX形式に変換する必要が生じたとしよう。Linuxで画像ファイルの形式を変換する場合、現在ならGIMPなどのGUIアプリケーションを使うのが一般的だし、GIMPでも大量のファイルの形式を一括変換する方法が存在する。

しかし、Xを立ち上げてGUIアプリケーションを開くのも面倒というような場合もあって、そのような場合にはコマンドラインから起動できる画像ファイル変換ツールを利用することになる。

LinuxにはXXXtoYYY（XXXやYYYには画像ファイル形式を表す略号が入る）という名前のコマンドがたくさんあって、これらをさまざまに組み合わせることで、画像ファイルをさまざまな形式に変換できるようになっている。詳しくは、

```
$ man ppm
```

を実行して、オンラインマニュアルを流し読みしてもらいたい。

ともかく、BMP形式のファイルをPCX形式に変換する場合、次のようなコマンドを実行することになる。

```
$ bmptoppm inputfile.bmp | ppmtopcx > outputfile.pcx
```

ここでは、bmptoppmコマンドでBMP形式の画像データをいったん「PPM形式」に変換し、さらにppmbopcxコマンドでPPM形式のデータをPCX形式のデータに変換している。この方法を使って、あるディレクトリ中にある大量のBMPファイル（拡張子が.bmp）をPCX形式に変換したい場合、forコマンドを使うと次のようになる。

```
$ for i in *.bmp
> do
> bmptoppm $i | ppmtopcx > $i.pcx
> done
```

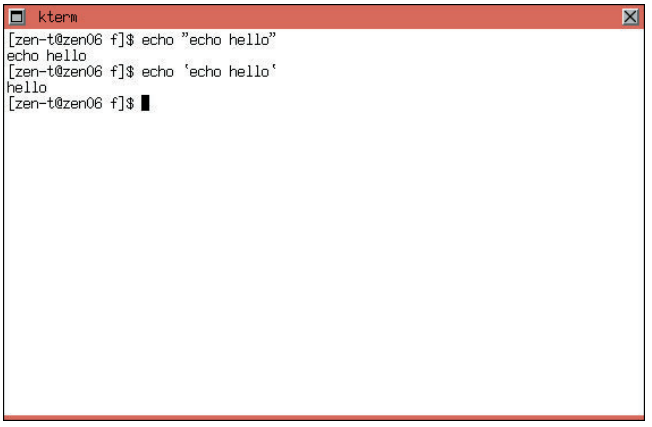
あるいは、先ほどの例のように、ファイル名が「××.bmp.pcx」ではまずいということであれば、

```
$ for i in *.bmp
> do
> bmptoppm $i | ppmtopcx > `echo $i | sed s/bmp$/pcx/`
> done
```

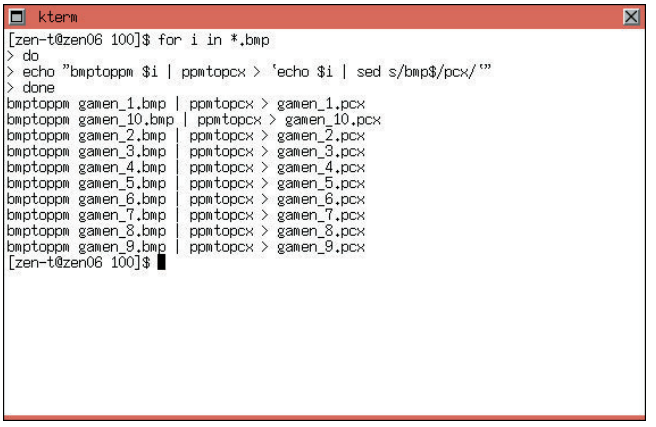
というように実行してやればよい。もちろん、コマンドの構文に自信がない場合には、実際に実行する前に、次のようにコマンド列全体を「echo」コマンドの引数にしてしまって、どのようなコマンドが実行されるかを確認したほうがよい（**画面5**）。

●

次回は、forコマンドのもう少し高度な使い方と、forコマンド以外の繰り返し処理について触れることにする。



画面4　echoコマンドを活用してコマンドの内容を確認する
コマンドを「`」（バッククォート）で囲むと、シェルはその部分のコマンドを先に実行して、出力された文字列と置き換える。この例では、先に`echo hello`を実行し、その結果出力された「hello」という文字列で`echo hello`の部分を置き換えたため、「echo hello」というコマンドを入力したのと同じになったのだ。



画面5
テキストファイルの拡張子を変える場合と同様に、echoコマンドを使って実行されるコマンドの内容を確認しておいたほうがよい。ただし、このように「長い」コマンドラインを実行する場合は、コマンド列を「"」（ダブルクォート）で囲っておかないと、エラーが発生することがある。

新連載

Oracle8iで作るWebサイト入門

本連載では、Oracle8iのLinux版とMiracle Linuxを使用して、データベースと連動するWebサイトを構築する方法を紹介します。

第1回　Miracle Linuxと Oracle8iのインストール

文：おもてじゅんいち／かざぐるま
Text : Junichi Omote/Kazaguruma

eビジネスという言葉があたり前のようになってきた昨今、単なるホームページを脱皮して、オンラインサイトと呼べるものへのレベルアップを目指す方も多いのではないでしょうか。ショッピングやコミュニケーション、情報検索サイト、B2BのECサイトにいたるまで、さまざまなオンラインサイトが世の中に現れるようになり、HTMLをうまく書けるだけではなかなか、「サービス」できるサイトを構築するのが難しくなっています。いずれにせよ、Webでちょっとしたサービスをするためにはデータベースの力を借りなければならないようです。

データベースとして有力なOracleからもLinuxに対応した製品がリリースされていて、マルチプラットフォームのデータベース製品として確固たるラインナップを揃えるようになっています。これでサーバとしてのLinuxの環境も、Oracleとともにビジネスでの利用が促進されることでしょう。

本連載では、Linux＋OracleでWebサイトを構築するためのプログラミング入門を、実践を通じて学習していきます。Oracleを使ったWebサイトのプログラミングには、CGIをはじめ、本格的にはJavaなど、いろいろあるのですが、ここでは比較的簡単に使えて、しかもちょっとした規模であれば十分実用になるPHPを使います。

OSのディストリビューションにはOracle向けにチューニングされたMiracle Linuxを採用します。Miracle LinuxではOracleのセットアップやWebサーバのApacheだけでなく、PHPまで標準でセットアップされますから、めんどうなカーネルの再構築や、特別なパッケージを入手してインストールするなどの必要がいっさいありません。

本連載では、Miracle Linux V1.0とOracle8i Workgroup Server for Linux R8.1.6（5指名ユーザー）がセットになった「Miracle Linux with Oracle8i Workgroup Server」を使用します。

Miracle Linuxは、商用ソフトとサポートが付属しないFTP版がミラクル・リナックスのFTPサイト（ftp://ftp.miraclelinux.com/pub/Miracle/1.0/iso/）で公開されています。また、日本オラクルのOTN Japan（Oracle Technology Network Japan）のWebサイト（http://otn.oracle.co.jp/）から、Oracle8iの期間限定トライアル版をダウンロードすることができます。

「ビジネスで使うリナックス活用マガジンLinux Business Vol.1」（ http://www.ascii.co.jp/linuxmag/business/vol1.html）の付録CD-ROMにも、Miracle Linux V1.0 FTP版とOracle8i Workgroup Server for Linux R8.1.6 120日間限定トライアル版が収録されていますので、これらを利用して本連載の内容を体験するのもよいでしょう。

Miracle Linuxのインストール

それではまず、OSであるMiracle Linuxをインストー

ルします。最近のLinuxのディストリビューションでは、デバイスの設定などがかなり自動化されるようになったため、簡単にインストールすることができるようになりました。なかでもMiracle Linuxのインストーラは目的がある程度決まっていることもあって、けっこう簡単なものになっています。

PCがCD-ROMからの起動に対応しているなら、Miracle LinuxのインストールCDをセットして電源を入れることでインストーラが起動します（**画面1**）。CD-ROMから起動できない場合は、インストールディスク（フロッピー）を作成する必要があります。

インストーラが起動すると、使用するキーボードの選択画面になります（**画面2**）。デフォルトで「jp106」が選択されているので、日本語キーボードの場合は、TABキーで「OK」に移動してEnterを押します。インストーラでは、このようなキー入力用のGUIになっていますので、矢印キーで項目を選択して、TABキーで反転カーソルを移動し、「OK」や「はい」「いいえ」などのところでEnterキーを押して確定します。

キーボードを選択すると、PCMCIAを使用するかどうかを聞いてきます。デスクトップPCの場合は「いいえ」を、ノートPCの場合は「はい」を選択してください。

次にインストール元のメディアを聞いてきますので、「CD-ROMドライブ」を選択します。すると、「CDを挿入してください」というメッセージが表示されますので、「OK」を押します。

次は、ネットワークの設定です。まず「TCP/IPの設定」画面になりますから、ダイヤルアップルータなどでインターネットに接続したい場合や、DHCPサーバがある場合には、「DHCPで設定する」のところに「＊」を付けます。「＊」はスペースキーでON／OFFできます。それ以外でネットワークに接続する場合は、適切なIPアドレス、ネットマスク、ゲートウェイ、DNSを設定します。ネットワークに接続しない場合でも、とりあえず以下のようにローカルIPアドレスを設定しておきましょう。

IPアドレス	192.168.0.1
ネットマスク	255.255.255.0
ゲートウェイ	192.168.0.1
ネームサーバ	192.168.0.1

TCP/IPを設定すると、ネットワークの設定になります。ドメイン内なら、ドメイン名とホスト名を設定してください。ダイヤルアップ接続なら、プロバイダのドメイン名と、適当なホスト名を入力すればOKです。2、3番目のネームサーバは空欄でもかまいません。

■パーティションの作成

次にハードディスクのパーティション設定ですが、Oracle8iを稼動させる場合は、パーティションの設定に気をつけてください。OSのMiracle Linuxはともかく、Oracle8iは十分なパーティションサイズがないと調子よく稼動してくれません。

とはいっても、そこはOracle8i用のディストリビューションであるMiracle Linuxです。Oracle8iをいっしょにインストールする場合のパーティション設定を自動で行ってくれます。それには、「パーティショニングモードの設定」画面で、「オートパーティショニングモード（for Oracle）」を選択すればOKです（**画面3**）。

このオートパーティショニングモードは、Linux用に「/」、「/boot」とOracle8i用に「/u01」、およびスワップ

画面1　インストーラ起動画面

画面2　キーボードの選択

用の4つのパーティションを作成してくれます。設定その他の操作は必要ありません。

とても便利なこのオートパーティショニングモードですが、ちょっと残念なことに、ハードディスクの容量が4Gバイト以上ないと作動しないことと、ディスクのほとんどの容量が/u01に割り当てられてしまうという条件があります。

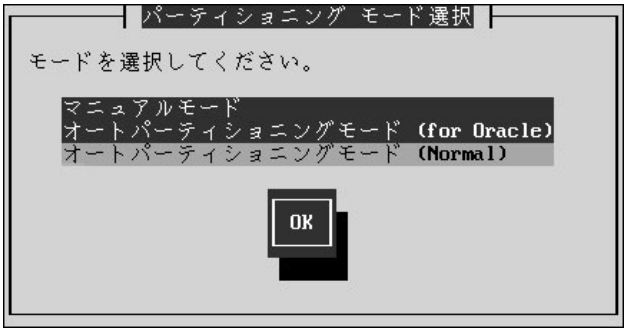
ディスク容量4Gバイト以上というのは、最近のPCであれば最新機種でなくても搭載されているでしょうし、ハードディスクだけ増設しても20Gバイトや30Gバイトのものが安価で出まわっていますから、そんなに問題にはならないでしょうが、各パーティションへの割り当て容量のほうはちょっとつらいかもしれません。

筆者のシステムの場合、30Gバイトのハードディスクをオートパーティショニング（for Oracle）で作成させたところ、おおよそ、

/	2Gバイト
/u01	27Gバイト
/boot	15Mバイト
swap	400Mバイト

という構成になってしまいました。

まあ、当面このマシンで本格的な運用はしないだろうからこれでもいいか、と思っているのですが、ルート（/）に2Gバイトで、/u01に27Gバイトというのは、いくらOracle8iといってもちょっとアンバランスな気がします。もちろん必要もないのにルートに割り当てて眠らせておくのはもっと悪いのですが、Oracle以外のアプリケーションサーバやJavaでのサーバ運用などを行うのであれば、もう少し調整したほうが良いと思います。メモリが128Mバイトというのを気にするなら、swap領域ももう少し多いほうが、と思ってみたり。



画面3　パーティショニングモード選択

このようにパーティショニングを調整したいなら、残念ながら**画面3**で、「マニュアルモード」を選択して、自分で各パーティションを設定しなければなりません。とはいえ、さほど難しいことではありませんし、一度、オートパーティショニングモードで作成してMiracle Linuxのインストールを完了し、パーティションの状況を確認してから、気に入らなければ、今度はマニュアルモードでインストールしなおす、といったことを繰り返すのも良いかもしれません。後述のOracle8iのインストールはけっこう時間がかかりますが、Miracle Linuxだけならそれほどかからないので、コーヒー片手に、慣れるまで何度かインストールを経験してみてはどうでしょうか。

パーティションの状況を確認するには、X Window System（GNOME）上なら、ktermを起動して、dfコマンドを実行するか、テキストモードでのコンソールなら、ログインして、同様にdfコマンドを実行してください。

ちなみに筆者のシステムでのdfコマンドの実行結果は**画面4**のようになりました。

あと、オートパーティショニングモードではハードディスクの内容がすべて消去されますので、他のOSと共存させたい場合なども、マニュアルモードで設定することになります。

■インストールタイプの選択

パーティションの設定が無事すんだら、インストールすべきパッケージを選択しなければなりません。ここでも、Miracle Linuxでは、おおまかな目的によって必要とするパッケージを組み合わせた、「インストールタイプ」というのを用意してくれています（**画面5**）。

かつてのLinuxディストリビューションでは、何百とあるパッケージ（サービスやコマンドの類）を自分で選んでいかなければならないものでした。最近になって、「サーバタイプ」や「ワークステーションタイプ」などの選択だ

```
[root@linux oracle]# df
Filesystem           1k-blocks      Used   Available   Use%   Mounted on
/dev/hda6              2071384    518420     1447740    26%   /
/dev/hda1                15522      2885       11836    20%   /boot
/dev/hda7             27047256    932308    24740992     4%   /u01
```
画面4　dfコマンドの結果

けですむようになりましたが、それでもOracleを利用する場合のタイプがあるのはさすがMiracle Linuxといったところでしょう。

本連載ではWebサーバとして運用しますから、「Apache利用オラクルRDBMS用サーバ」の選択をお勧めします。それ以外の利用目的があるのなら、必要に応じてインストールタイプを選択してください。

インストールタイプを選択後、「インストール」を押すと、パッケージのインストールが始まります。少し時間がかかりますので、コーヒーブレークにしてください。次々とパッケージ名と説明が表示されますから、眺めているのもLinuxのパッケージを何となく理解するのに役立つかもしれません。

このとき、インストールを続けるか、カスタマイズするかを聞いてきます。特に追加したいパッケージがない場合や、よくわからない場合は「続ける」を選択してください。

パッケージのインストールが完了すると、ブートディスクを作成するかどうかを聞いてきます。これは、何らかの理由により、ハードディスクからシステムを起動できなくなったときに、フロッピーディスクから起動して、システムを修復したり、ハードディスク内のファイルだけでも救済するために必要なものです。できるだけ作成しておくことをお勧めします。

最初に使ったインストールディスクとこのブートディスクは別のものですから、混同しないようにラベルを貼っておきましょう。インストールディスクは、また最初からインストールする場合に必要になります。

次はタイムゾーンの設定です。特別な理由がないかぎりはデフォルト通りの「Japan」で良いでしょう。

続いて、システム管理者（rootユーザー）のパスワード設定です。長くてむずかしいパスワードにするのは良いことですが、くれぐれも忘れないようにしてください。このあとも必要になりますから。

画面5　インストールタイプ選択

システム管理者のパスワード設定が完了すると、コンピュータを再起動するというメッセージが表示されます。メッセージにもありますが、念のためCD-ROMは抜いておきましょう（CD-ROMから起動できるマシンの場合は必ず抜いておいてください）。

コンピュータが再起動してしばらくすると、ふたたびインストールが続行されます。ここでは、コンパニオンCDなど追加のCDがあるかどうかを聞いてきます。コンパニオンCDは製品版に同梱されていますので、画面のメッセージおよびインストールマニュアルを参考にインストールしてください。

追加CDがあるかどうかを聞かれたところで「いいえ」を選択すると、次はX Window Systemの設定になります。

■X Window Systemの設定

まずキーボードタイプの設定です。設定画面が3つ表示されますので、順に「jp106」「日本語 jp106」「日本語」と選択します。

マウスの設定では、最近のPCで使われるのはほとんどが2ボタンのPS/2マウス（キーボードと同じ形のコネクタのもの）ですから、メーカーと機種名がわかっているならそのモデルを、よくわからないなら、「一般的なPS/2マウス」を選択します（**画面6**）。このとき、「3ボタンのエミュレーション」に「＊」を付けておいてください。

次の画面でマウスボタンの数を聞いてきますので、「ボタン2つ」を選択しておきましょう。

次はビデオカードです。たいていの場合は自動認識されますので問題はありませんが、どうしても自動認識されないときは、自分でビデオカードの種類を選択しなければなりません。そのためにマシンのビデオカードの種類をあらかじめマニュアルやカタログなどで調べておく必要があります。メーカー製で現行モデルならWebサイトに記載されていることもあります。いずれにせよ、「ビデオカードの自動認識」画面では一度、自動認識をさせてみましょう。

ビデオカードがうまく自動認識されると、検出できたというメッセージが表示されますので、「検出値で設定」を選択してください。

次のディスプレイの設定ですが、これは自動認識されないので、メーカーとモデル名を選択します。もし、お使いのディスプレイのモデル名が一覧にない場合は、手動設定することになります。その場合は、ディスプレイの水平／垂直周波数帯が必要になりますので、マニュアルなどで確認してください。また、一覧にあるモデルの詳細情報を見

て、似たようなモデルに設定するというのも良いかもしれません。ディスプレイの設定に関しては、あまり厳密に合っていなければならないというものではありませんから。

その後、色数、画面サイズ、フォント解像度、デフォルト画面サイズの設定に進みます。これらは設定して表示テストをして、再度設定しなおすということが可能ですから、どういった設定が適切なのかわからない場合は、

色数は16bpp（65536色）以上
画面サイズは1024×768以上
フォント解像度は100DPI
デフォルト画面サイズは1024×768（以上）

に、それぞれいったん設定してみましょう。すると設定のテストをするというメッセージが表示されますから、「続ける」を押してください。

カラーのテスト画面と、「Quit」「Next」という2つのボタンが表示されたら正常です。位置がずれている場合は、ディスプレイのつまみで水平位置などを直すか、設定の「画面サイズ」のところで、違う周波数に設定する必要があるかもしれません。

また、「画面サイズ」のところで複数のサイズを選択していると、テスト画面の「Next」ボタンをマウスでクリックすることにより、順に各サイズをテストすることができます。

「Quit」をクリックすると、インストーラに戻って、「すべてうまく動作しましたか？」というメッセージが表示されます。ここで「いいえ」を押せば、ふたたび設定画面に戻りますので、必要があれば設定を変更して、テスト表示させてみましょう。最終的に、ちらつきがなく画面内におさまる最大の画面サイズを選択してください。色数に24bppや32bppを設定している場合、大きな画面サイズが表示できないことがありますので、色数を下げてテストしてみましょう。

設定が完了したら、「すべてうまく動作しましたか？」というメッセージのところで「はい」を押してください。

次にログイン方法の設定画面になりますので、「グラフィカルログイン」か「テキストログイン」かを選択します。「グラフィカルログイン」は、ログイン後X Window Systemが起動されます。「テキストログイン」はテキストベースのコンソール画面です。このあとに続くOracle8iのインストールは、X Window System上で行うので、「グラフィカルログイン」を選択しておきましょう。

X Window System設定の最後はウィンドウマネージャの選択です。Miracle Linuxディストリビューションに用意されているのは、「GNOME」と「TWM」です。「GNOME」のほうが見映えや使い勝手が良いのでこちらを選択しておきましょう。「TWM」はシンプルです。

■自動起動の設定

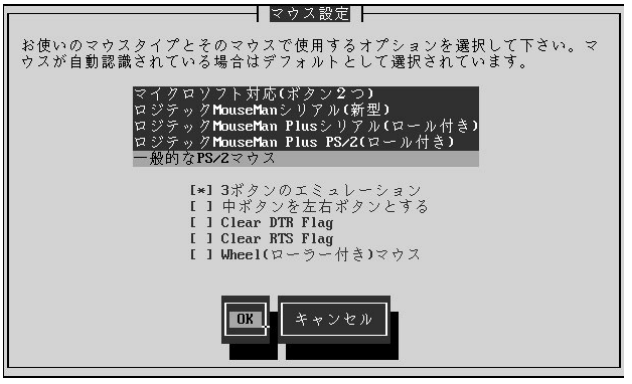
いよいよ最後は、起動サービスの設定です。この一覧で起動に「＊」のついているサービス（デーモン）が、システム起動時に自動的に起動されます（**画面7**）。通常自動起動すべきサービスはあらかじめ「＊」が付けられているのですが、実は今回インストールタイプで選択した、「Apache」と「Oracle8i」が自動起動になっていませんので、自動起動するように設定します。インストール後でも設定することは可能ですが、ここで設定しておくほうが簡単です。

サービス名の一覧を下にたどっていってください。「dbora」というサービス名のところを反転させて選んだまま、「有効［E］」を押してください。これがOracle8iの自動起動の設定です。続いて、「httpd」も有効にしてください。こちらがhttpdサービス、つまりApacheサーバのことです。これで、ApacheとOracle8iが自動起動することになりました。ESCを押して終了させてください。

これでMiracle Linuxのインストールが終了しました。再起動したら、ログイン名rootと、先ほど設定したrootのパスワードでログインしてください。

Oracle8iのインストール

さて、無事にログインもできてGNOMEの画面が表示されたことと思いますが、引き続きOracle8iのインストールを行います。



画面6　マウス設定

実は、Miracle Linuxには、Oracle8iのグラフィカルインストーラと、インストールガイドがセットされているのです。

まずはGNOMEメニュー（足のアイコン）の「管理者ツール」から、「インストールナビゲーター for Oracleユーザーズガイド」を選択してください。これがインストールガイドで、とてもわかりやすく、詳細に書かれているので、ぜひこのガイドを画面に残して参照しながらインストールを進めてください。

インストーラの起動は、同じくGNOMEメニューの「管理者ツール」から、「インストールナビゲーター for Oracle」を選択します。

ここで、製品に含まれるCD-ROM「Oracle8i Standard Edition R8.1.6 for Linux」をドライブにセットします。

インストールの手順は、上記のガイドに沿って進めていくものとして、ここではいくつか参考になる点を挙げていきます。

■インストール前に決めておくこと

ガイドの「1．準備」では、インストール作業に入る前に、

インストール用アカウント名
データベース管理者用アカウント名
データベースのSID
データベースを格納する場所（パス名）

をそれぞれ決めておくようにと書かれていますが、これらはすべて規定値があるので、特に決めておかなくても大丈夫です。SIDはガイドに明記されていませんが、「orcl」を指定してください（ガイドの「5.ユーザーアカウント情報の入力」の画面には「orcl」が入力されている）。

もちろん実運用を考えて、セキュリティの面からもそれぞれを別のものに設定することも良いのですが、とりあえずOracle8iを体験することが目的なら、本連載や他の書籍などでの説明はたいてい規定値で行われるので、最初は規定値のままインストールしてみることをお勧めします。

インストールガイドでは、Oracle8iのバージョンとして、R8.1.5とR8.1.6について書かれています。R8.1.5のほうが作業はいくらか多くなっていますが、今回インストールするのはR8.1.6ですから、R8.1.5の部分は読み飛ばしてもOKです。

画面7　起動サービス設定

■シェルスクリプトの実行

このインストーラは基本的にはウィザードにしたがっていくだけなのですが、2カ所だけ、コマンドラインでシェルスクリプトを実行しなければならない場面があります（**画面8**）。ガイドでは、「9.シェルスクリプトの実行 -Part1-」「17.シェルスクリプトの実行 -Part2-」として記述されている部分です。インストーラはOSに対してrootユーザーとしてではなくインストールユーザーとして起動されているのですが、この2カ所についてはroot権限でないと実行できない処理が含まれているために、用意されたシェルスクリプトをインストーラ外で実行してやらなければならないのです。

画面8が表示されたら、そのままボタンをクリックせずに、ktermを立ち上げます。画面のメッセージとガイドに書かれているスクリプト名、

```
/u01/app/oracle/product/8.1.6/orainstRoot.sh
```

をktermに入力してEnterを押して実行します。スクリプトが実行されて、再びプロンプトが表示されたら、ktermを終了して**画面8**に戻り、「Retry」ボタンをクリックしてください。「17.シェルスクリプトの実行 -Part2-」の時も同様に行ってください。

■データベースの作成

ガイドの「14.データベースの作成」のところで、「データベースの作成はOracle8iインストール後に、dbassistコマンドを使って行うことをお勧めします」とありますが、ここはお勧めにしたがってください。ガイドにも実際にここでデータベースを作成する場合の手順は書いてありませんし、あとでdbassistを使えば、サンプル付きのデータベースが簡単に作成できるからです。

ガイドの19.でOracle8iのインストールが完了し、引き

続きdbassistによるデータベース作成になります。ここでも、ガイドにしたがって「CDから既存データベース・ファイルをコピー」を選択してください。「23.データベース名とSIDの設定」では、グローバル・データベース名、SIDともに「orcl」と入力します。

■メモリ使用量のチューニング

インストール、データベースの作成が終わったら、ガイドにしたがってデータベースの接続確認を行います。ガイドどおりに表示されれば、インストールも無事完了していて、データベースが正常に動いています。ではここで、もう一度、SQL＊Plusを起動して、SQL>プロンプトが表示されたら、show sgaと入力してください。現在のOracle8iのメモリ使用量が表示されます（**画面9**）。特に1行目の約56Mバイトというのはつねに使用される量ですから、マシンの搭載メモリが256Mバイト以上あればそれほど問題になりませんが、筆者のように128Mバイトの環境だと、他のサービス類が圧迫されてしまうことになり、全体に動作スピードが遅くなってしまいます。そこで、メモリ使用量を減らすようチューニングしてみます。

この作業はデータベースサーバを停止させてから行ってください。データベースサーバの起動／停止は、ガイドの「26.データベース・サーバーの起動と停止」にしたがって、svrmgrlを使います。svrmgrlを起動して、

```
SVRMGR> connect internal
SVRMGR> shutdown
```

画面8　スクリプトの実行メッセージ

とすれば停止します。

次に、/u01/app/oracle/admin/orcl/pfile/initorcl.oraという初期化パラメータファイルを編集します。**リスト1**のように編集してください。このとき、オリジナルのファイルはinitorcl.ora.orgといった名前でコピーしておきましょう。

initorcl.oraファイルの編集は、基本的にあらかじめ用意されている#SMALLの設定値を、行頭の＃を削除して有効にしているだけです。同時にデフォルト値である#INITIALの行をコメントにしています（行頭に＃を付ける）。

java_pool_sizeだけは、#SMALLの設定値がなかったのですが、本連載ではJavaを使わないので、1Mバイト程度にしておきます。

ファイルを編集後、再びsvrmgrlでデータベースサーバを起動します。

```
SVRMGR> connect internal
SVRMGR> startup
```

サーバが起動したら、このチューニングの結果を確認してみます（**画面10**）。50Mバイト以上あったエリアが7Mバイトほどになり、他のバッファ類も小さくなっています。

■Oracle8iの自動起動

Miracle Linuxの「サービスの起動設定」で、doraを起動するように設定しましたが、残念ながらそれだけではデータベースが起動しないため、自動起動のための設定を行います。まず、/etc/oratabというファイルを編集します。このファイルの最終行が、

```
orcl:/u01/app/oracle/product/8.1.6:N
```

となっているはずなので行末のNを、

```
orcl:/u01/app/oracle/product/8.1.6:Y
```

```
SQL> show sga

Total System Global Area   56008688 bytes
Fixed Size                    69616 bytes
Variable Size              38989824 bytes
Database Buffers           16777216 bytes
Redo Buffers                 172032 bytes
SQL>
```

画面9　デフォルトのメモリ使用量

と、Yに修正してください。これは、Oracle8i本体ではなく、データベースごとに設定される自動起動のスイッチです。今は、行頭にある「orcl」というデータベースだけの設定しかありませんが、今後複数のデータベースを扱うようになると、データベースごとに同様の記述がありますので、システム起動時にどのデータベースを自動起動するかを設定できます。

次に、/u01/app/oracle/product/8.1.6/bin/dbstartというスクリプトを修正します。これは、データベースの起動を行うOracle8i側のスクリプトで、Miracle Linuxのインストール時に設定した「サービスの起動」が、このスクリプトを実行することによって、実際にデータベースを起動するのですが、このスクリプト内で実行される起動コマンドのsvrmgrlのバージョンとこのスクリプトのバージョンがうまく合っていないために、起動に失敗してしまうようです。

今後のマイナーバージョンか、サポートパッチなどでいずれ修正されると思いますが、とりあえず現バージョンでは修正が必要です。

修正は、**リスト2**のように行ってください。これは、スクリプトが、svrmgrlのバージョンをチェックする部分で、svrmgrlによって出力されるメッセージに、「PL/SQL Release…」が含まれているかどうかを検査しているのですが、今回のバージョンのメッセージは、

```
JServer Release 8.1.6.0.0
```

と出力されるため、PL/SQLという部分が一致しなくなって、バージョン情報を取り出せなくなってしまうのです。そこで、「PL/SQL」か「JServer」に続くメッセージを検出できるように修正しています。

以上の修正が完了したら、システムを再起動して、無事データベースサーバが起動するかを確認しましょう。

さて、これでMiracle LinuxとOracle8iがインストールできました。次回からはPHPを使って、Webサイトを構築していきます。

▼リスト1　initorcl.oraファイルを編集

(52行目から)

```
db_files = 80                                   # SMALL
# db_files = 400                                # MEDIUM
# db_files = 1500                               # LARGE

open_cursors = 100
max_enabled_roles = 30
db_file_multiblock_read_count=8                 # SMALL
# db_file_multiblock_read_count=16              # MEDIUM
# db_file_multiblock_read_count=32              # LARGE

_use_ism = false
# db_block_buffers = 2048                       # INITIAL
db_block_buffers = 100                          # SMALL
# db_block_buffers = 550                        # MEDIUM
# db_block_buffers = 3200                       # LARGE

# shared_pool_size = 15728640                   # INITIAL
shared_pool_size = 3500000                      # SMALL
# shared_pool_size = 5000000                    # MEDIUM
# shared_pool_size = 9000000                    # LARGE

large_pool_size = 614400
# java_pool_size = 20971520
java_pool_size = 1000000

log_checkpoint_interval = 10000
log_checkpoint_timeout = 1800

# processes = 50                                # INITIAL
processes = 50                                  # SMALL
# processes = 100                               # MEDIUM
# processes = 200                               # LARGE

# log_buffer = 163840                           # INITIAL
log_buffer = 32768                              # SMALL
# log_buffer = 32768                            # MEDIUM
# log_buffer = 163840                           # LARGE
```

(後略)

```
SQL> show sga

Total System Global Area    7319536 bytes
Fixed Size                    69616 bytes
Variable Size               6352896 bytes
Database Buffers             819200 bytes
Redo Buffers                  77824 bytes
SQL>
```

画面10　チューニング後のメモリ使用量

▼リスト2　dbstartスクリプトを修正

(dbstartスクリプト64行目)

(変更前)　`/PL\/SQL (Release|Version)/ {substr($3,1,3) ;`

(変更後)　`/(PL\/SQL|JServer) (Release|Version)/ {substr($3,1,3) ;`

InterBase 6.0

IBPerlを使用することでCGIからInterBaseを呼び出すことが簡単に実現できます。IBPerlによりInterBaseの手軽さがさらに高まり、さまざまな用途でデータベースを利用していくことができるでしょう。最終回では、見積書発行プログラムを完成させましょう。

最終回　見積書発行プログラムの作成

文：加藤大受
Text:Daiju Kato　dkato@jcom.home.ne.jp

前回は、IBPerlを使用して見積書発行システムへのユーザー認証を行いました。ログイン画面のHTMLから、IBPerlを使用したCGIを呼び出すことで、InterBaseのデータベースを利用したユーザー認証を行うことができます。ユーザー認証はすでに解説したように、ストアドプロシージャを利用することで手軽に行うことができます。前回までの復習を含めて、ログインから見積書発行までの流れを再度確認してみましょう。

前回はユーザー認証までですので、ib-login.plからestimate.plを呼び出すようなメニューで終えていましたが、操作性を高めるため、ユーザー認証が成功したらすぐに見積書の発行プロセスへと移行します。見積書を作成するには、商品の選択、数量の入力が必要となりますので、ユーザー認証後にこれらの画面となるHTMLを生成します。ここで重要なのは、次のCGIを呼び出すときにユーザー名を渡してやることです。今回のシステムではユーザー

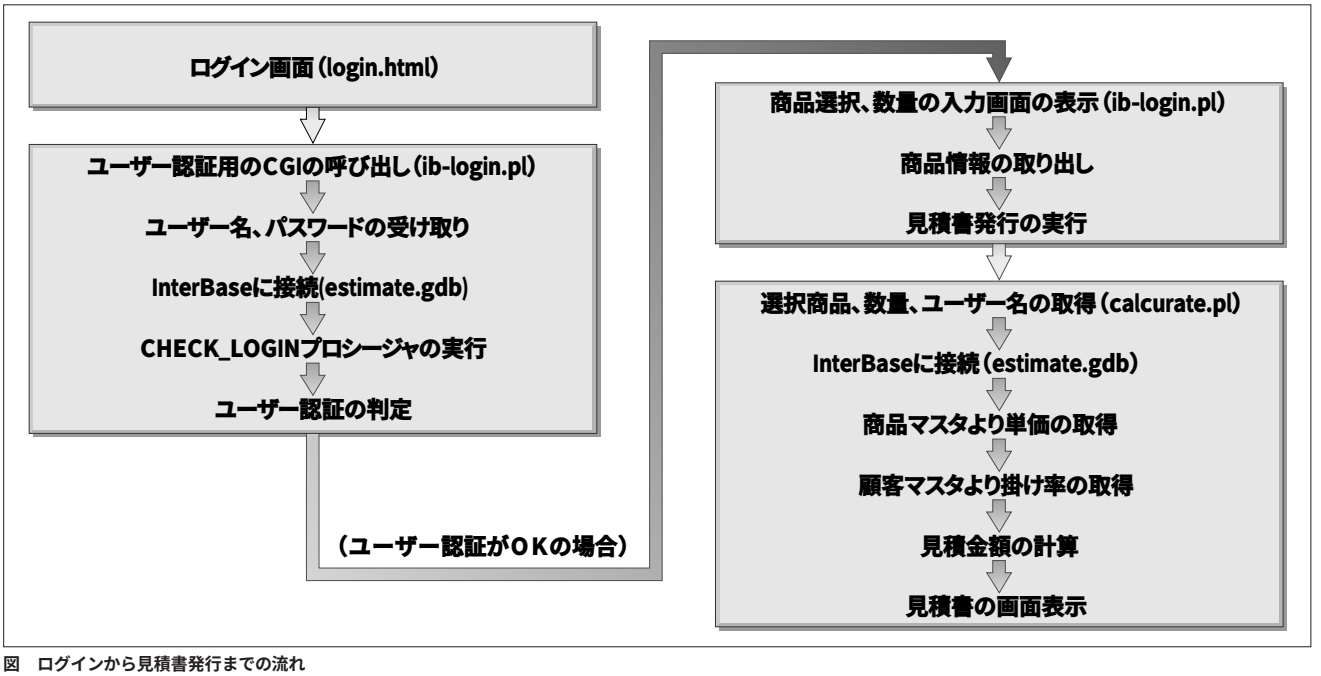
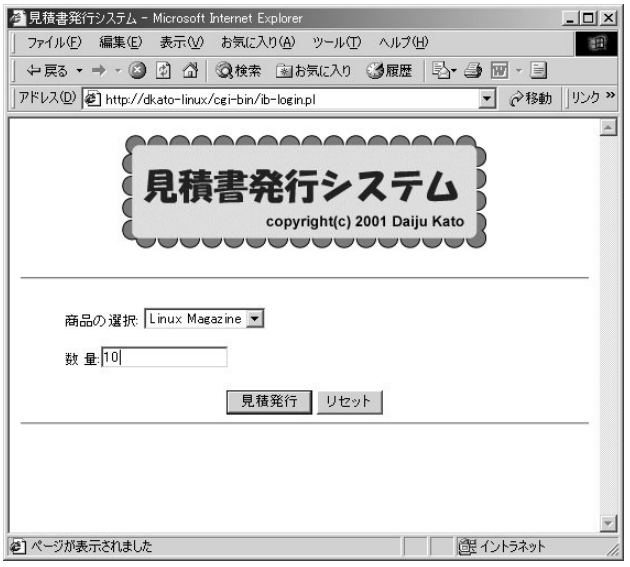


図　ログインから見積書発行までの流れ

ごとに掛け率が異なりますので、実際に見積書を作成するためにはユーザー名が必要となります。これをINPUTタグのHIDDEN属性などを利用して渡してやることを考慮する必要があります。

リスト1は、前回のib-login.plのユーザー認証がOKの場合の処理に、見積書発行のための画面生成を加えたものです。商品名を選択するリストボックス、数量を入力するテキストフィールド、ユーザー名を格納するHIDDEN属性などを生成していることがわかると思います。生成された画面は**画面1**のようになります。商品名を生成する部分はprod_outというサブルーチンで行います（**リスト2**）。



画面1　商品選択画面（Windowsから呼び出し）

prod_outサブルーチン内では、商品マスタから取り出した商品情報をリストボックスのアイテムとして追加しています。行がある限り、fetchコマンドを呼び出して商品を取り出します。

商品名の選択、数量の入力が終了したら見積発行ボタンを押して、見積書の計算と発行を行うcalcurate.plを実行します。**図**で流れをまとめたように、calcurate.plでは選択された商品、数量、ユーザー名を受け取り、商品単価と掛け率の検索を行い、見積金額の計算、見積書の発行を行います。

ib-login.plから送られてくる商品名には日本語が含まれる可能性がありますので、日本語のデコードを考慮してやる必要があります。Webベースのシステムを作成する場合、このデコード処理が結構やっかいなのですが、Perlの

▼リスト1　商品選択画面の生成(ib-login.plより抜粋)

```
    #ユーザー名が正しいかどうかのチェック
    if ($islogin eq 1) {
        print "<FORM METHOD=POST ACTION='/cgi-bin/calcurate.pl'>¥n";
        print "<P align='center'>";
        print "<IMG src='../title.gif' width='332' height='103' border='0'></P>¥n";
        print "<HR><!商品の選択>¥n";
        print "<blockquote>商品の選択:¥n";

        #商品名の選択
        &prod_out;
        print "</SELECT><BR><BR>¥n";

        #数量の入力
        print "数　　量:<INPUT TYPE='text' name='volume'><BR><BR></blockquote>¥n";

        #見積書発行ボタン
        print "<CENTER><INPUT TYPE='SUBMIT' VALUE='見積発行'>¥n";
        print "<INPUT TYPE='RESET' VALUE='リセット'><BR>¥n";
        #ユーザー名の格納
        print "<INPUT TYPE='HIDDEN' NAME='userid' VALUE='$user'>¥n";
        print "<HR></FORM></BODY></HTML>¥n";

        #データベースから切断
        $db->disconnect();
    } else   {
        print "<H1><CENTER>ユーザー名が正しくありません。</H1></CENTER><BR>";
        print "<CENTER><A HREF='http://dkato-linux/login.html'>[戻る]</A></CENTER><HR>";
    }
```

場合は受け取ったパラメータを指定された文字コードに変換してやるだけで簡単に処理することができます。この処理を行うのがjcode.plで、Kazumasa Utashiro氏によって作成されたスクリプトです。最新版は**表1**のURLから入手することができます。

jcode.plはApacheのcgi-binディレクトリに入れてもいいですが、use lib構文を使用して場所を指定することもできます。使い方は単純に、**リスト3**のようにデコードしたい文字コードを指定するだけです。ここではEUCJISに変換しています。

パラメータを受け取ったら見積書の生成を行います（**リスト4**）。商品の選択と数量の入力画面と同じように、画面生成の処理の中から単価の取得、掛け率の取得を行うサブルーチンを呼び出します（**リスト5・6**）。これらのサブルーチンにはそれぞれ商品名、ユーザー名をパラメータで渡し、結果を受け取る必要があります。このため、パラメータ渡しと結果の受け取りを行っています。商品単価の検索はget_priceサブルーチンで、掛け率の検索はget_rateサブルーチンで行います。これらのサブルーチンで行うことは、単にSELECT構文のWHERE条件に受け取った商品名、またはユーザー名を渡して、商品単価または掛け率を検索しているだけです。これらの部分は言語が変わっても基本的にデータベースとのやりとりなので、Perlが苦手な方も流れを理解できると思います。ここではそれぞれのサブルーチン内でデータベースへの接続・解除を行っていますが、データベースの接続・解除をメインルーチンで行い、各サブルーチンは検索処理だけにしてもかまいません。

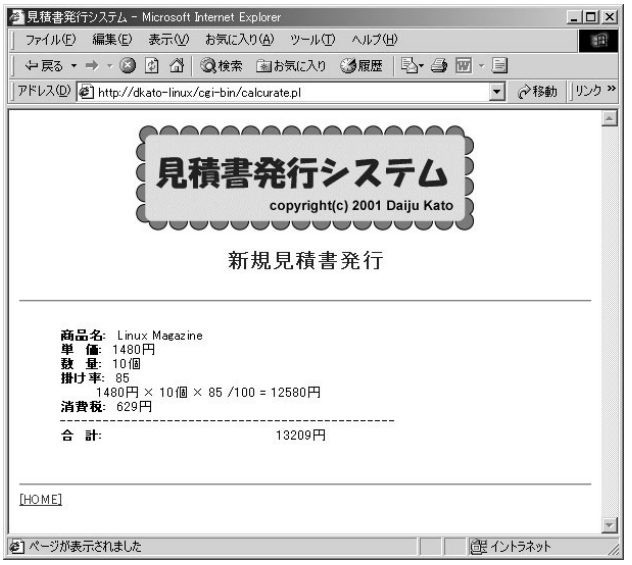
画面2は、見積書が生成された画面です。掛け率が適用

▼リスト2　商品名の出力(prod_outサブルーチン)

```
#商品名を出力するサブルーチン
sub prod_out {
    #トランザクションの開始
    $tr = new IBPerl::Transaction( Database => $db );
    if ($tr->{Handle}<0) { print "データベースへの接続に失敗しました<BR>¥n"; }

    #商品情報の取り出し
    $query = "SELECT PRODUCT_NAME from PRODUCTS";
    $st = new IBPerl::Statement( Transaction => $tr, SQL=>$query);
    $st->execute();
    #商品名の取得
    print "<SELECT NAME='product'>";

    while (($status=$st->fetch(¥$result))==0 ) {
        print "<OPTION>$result¥n";
    }
}
```

▼リスト3　受け取った商品情報をEUCに変換(calcurate.plより抜粋)

```
#!/usr/bin/perl
$|=1;

#初期設定
use lib '/home/db';
require 'cgi-lib.pl';
require 'jcode.pl';
use IBPerl;

&ReadParse(*in_data);
    $product = $in_data{'product'};
    $volume  = $in_data{'volume'};
    $userid  = $in_data{'userid'};

#EUCJISに変換
&jcode::convert(*product, 'euc');
```



画面2　見積書の生成

されて見積金額が表示されていることがわかると思います。また、消費税のところでは円未満は切り捨てになるように処理が書かれている点も確認してください。

凝った画面の作りにしていないので、なんとなく単純に思えますが、会社ごとに掛け率の変更などを行っていますので、ある程度は実用的な流れになっているでしょう。また、LOGSテーブルにはいつ誰が見積書を発行したかのログが書かれています。

システムの改良

では、より実際に運用できる形に改良するにはどうすればいいのでしょうか？　まずは見積番号の生成と、それらの履歴を見積マスタに格納することです。

現状では、estimate_masterテーブルを使用していませんが、見積を生成するときにこのテーブルに結果を書き込んでやればいいのです。ただし、このテーブルでは会社コードと商品コードが必要になりますので、商品単価の取得を行うサブルーチンで商品コードも単価と同時に取得します。会社コードも同様です。これらは、get_rateおよび、get_priceサブルーチンの改良ですみます。

商品数が多い場合は、商品をカテゴリ分けして選択する必要があるでしょう。その場合は、カテゴリを規定するカテゴリマスタを作成し、カテゴリコードのみを商品マスタに入れる形にします。カテゴリなどを直接マスタに書き込むと、あとでメンテナンスや改良が行いづらいスキーマになってしまいますので注意してください。

また、このマスタを参照するための画面を作成してもよいかもしれません。ただし、この画面を表示できるのはadminユーザーのみに限るなどの仕組みが必要です。たとえば、ib-login.plのユーザー認証で、ユーザー名がadminの場合のときだけ別のメニューを表示するなどの工夫が必要でしょう。

また、複数商品の同時選択などについても対応する必要があるかもしれません。この場合は、フレームをうまく利用して上画面では選択、下画面では選択されたものがテー

▼リスト4　見積書の生成

```
print "content-type: text/html; charset=EUC-JP¥n¥n";
print "<HTML><HEAD>¥n";
print "<META http-equiv='Content-Type' content='text/html; charset=EUC-JP'>¥n";
print "<TITLE>見積書発行システム</TITLE>¥n";
print "</HEAD><BODY>¥n";
print "<P align='center'><IMG src='../title.gif' width='332' height='103' border='0'></P>¥n";
print "<P align='center'><FONT size='+2' color='#000099'><B>新規見積書発行</B></FONT></P>¥n";
print "<HR>¥n";

#単価の取得
$price=&get_price($product);
$rate=&get_rate($userid);
$subtotal=$price*$volume*$rate/100;
$tax=$subtotal*0.05;
if ((index $tax,".")!=-1)
    {
    $tax=substr $tax, 0, (index $tax, ".")-1;
    }
$total=$subtotal+$tax;
print "<BLOCKQUOTE>";
print "<B>商品名: </B>$product<BR>";
print "<B>単　価: </B>$price円<BR>";
print "<B>数　量: </B>$volume個<BR>";
print "<B>掛け率: </B>$rate<BR>";
print "       $price円 × $volume個 × $rate /100 = $subtotal円<BR>";
print "<B>消費税: </B>$tax円<BR>";
print "-----------------------------------------------<BR>";
print "<B>合　計:                              </B>$total円<BR>";
print "</BLOCKQUOTE><BR>";
print "<HR><A HREF='../index.html'>[HOME]</A>";
print "</FORM></BODY></HTML>¥n";
```

ブル形式で表示されるなどの仕組みを考えてみるとよいでしょう。

しかし、何といっても、システムを作成する上で一番重要なのはスキーマ構造となります。もちろんシステム全体のパフォーマンスも非常に大切ですが、パフォーマンスはハードウェアリソースを改善することで容易に対応することができます。しかし、スキーマ構造に拡張性がないと、将来システムを拡張できないというようにシステムに先がなくなってしまいます。この連載の中で何度も説明してきましたが、システムを設計するときにできるだけ長い時間をスキーマ設計に割り当て、拡張性に優れたスキーマを設計しておく必要があります。また、あまりデータベースに固有な機能を使わずに、汎用的に使用できる構造にしておくことも必要でしょう。開発者はどうしてもプログラム言語の違いや、オペレーションシステム、ネットワークインフラに目を向けがちですが、データベースをシステムの中心として考え、拡張性と将来性のあるスキーマが設計できるように考えてほしいと思います。

InterBaseの利用価値

これまで8回に渡り、オープンソースになったInterBaseの最新バージョンであるInterBase 6.0について、前半はInterBaseの機能、後半はIBPerlというPerl用のドライバを利用して簡単なアプリケーションを構築してみました。

InterBaseはPerl用のドライバ以外にも、InterClientというJavaプログラムからの呼び出しを行えるJDBCドライバや、IBExpressという、DelphiやC++Builderからネイティブに接続できるコンポーネントなど、数多くのユーティリティが存在します。また、PHP3で使用するためのドライバなどについても、非常に速いペースで開発が続けられています。さらに、ODBCも同様に有志によって開発が進められています。

日本ではあまり知名度のないデータベースですが、Oracleの登場と同じくらいに歴史があり、数多くの会社で基幹系のデータベースとして使用されてきた実績がありま

▼リスト5　単価の取得

```
sub get_price {
    $select_product=$_[0];
    #データベースに接続
    $db = new IBPerl::Connection(
        Path     => '/home/db/estimate.gdb',
        User     => 'SYSDBA',
        Password => 'masterkey',
        Charset  => 'EUCJ_0208');
    die "Connection->new() error:¥n$db->{Error}¥n" if ($db->{Handle} < 0);
    if ($db->{Handle}>0)
    #データベースに接続できた場合
    {
        #トランザクションの開始
        $tr = new IBPerl::Transaction( Database => $db );
        if ($tr->{Handle}<0) { print "データベースへの接続に失敗しました<BR>¥n";}

        #商品情報の取り出し
        $query = "SELECT List_price from PRODUCTS where PRODUCT_NAME='$select_product'";
        $st = new IBPerl::Statement( Transaction => $tr, SQL=>$query);
        $st->execute();

        #単価の取得
        $st->fetch(¥$result);

        #データベースから切断
        $db->disconnect();
        return $result;
    } else {
    #データベースに接続できない場合
        print "データベースへの接続に失敗しました<BR>¥n";
    }
}
```

す。オープンソースになったから信頼性が低くなったということもありません。信頼性のあるデータベースがオープンソースになり、だれでも手軽に使えるようになったことは非常に喜ばしいことです。

また、InterBaseの特徴であるメンテナンスレスは、データベースを利用したシステムを運用していく場合に非常に管理者の負担を軽減します。ファイルコピーでデータバックアップが取れることや、特にパフォーマンスチューニングを必要としない点は初心者でも手軽に使える要素でもあります。

すでにOracleやDB2 UDBなどの商用データベースを使用されている方も、これからデータベースを利用していこうと思っている初心者の方も、InterBaseを候補のひとつとして挙げてみるといいと思います。私自身、すでに5年近くInterBaseを使用していて、非常に使いやすい、管理しやすいものであると評価しています。もちろん、大規模システムにおけるトランザクションには向いていませんが、世の中の70％程度のシステムで使用できるデータベースであり、このデータベースを使用することにより、データベースに関して問題となっている多くの事例を解決することができるのではないかと思っています。

日本国内では、まだまだInterBaseに関する情報は非常に少ないですが、欧米では活発的にメーリングリストやニュースグループでの情報交換が進んでいます。そういった情報が少しずつ伝わり、日本でもさらに多くの開発者が使用していくことになるでしょう。

私の稚拙な記事がより多くの開発者にとって少しでも有益なものであることを祈って、InterBaseの連載を終えたいと思います。

InterBaseのホームページ	http://www.interbase2000.com/
jcode.plのダウンロードサイト	ftp://ftp.iij.ad.jp/pub/IIJ/dist/utashiro/perl/

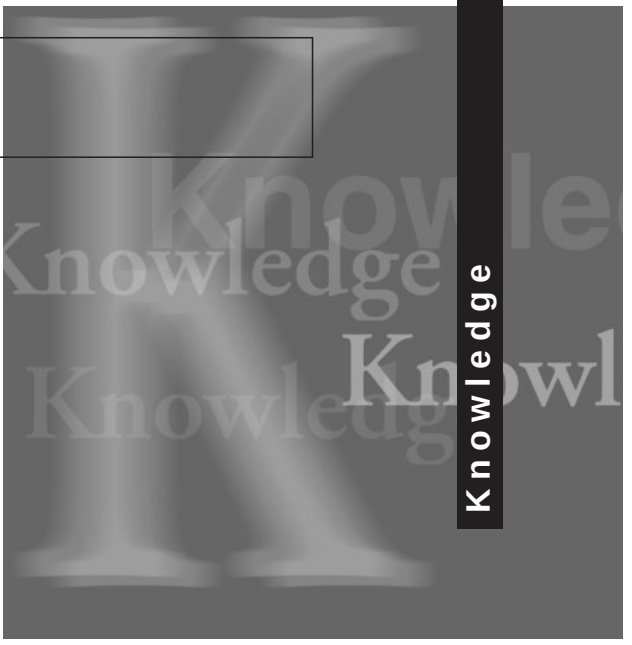
表1　URL一覧

▼リスト6　掛け率の取得

```
sub get_rate {
    $loginid=$_[0];
    #データベースに接続
    $db = new IBPerl::Connection(
        Path     => '/home/db/estimate.gdb',
        User     => 'SYSDBA',
        Password => 'masterkey',
        Charset  => 'EUCJ_0208');
    die "Connection->new() error:¥n$db->{Error}¥n" if ($db->{Handle} < 0);
    if ($db->{Handle}>0)
    #データベースに接続できた場合
    {
        #トランザクションの開始
        $tr = new IBPerl::Transaction( Database => $db );
        if ($tr->{Handle}<0) { print "データベースへの接続に失敗しました<BR>¥n"; }

        #掛け率の取り出し
        $query = "SELECT RATE from COMPANY where LOGIN_ID='$loginid'";
        $st = new IBPerl::Statement( Transaction => $tr, SQL=>$query);
        $st->execute();

        #掛け率の取得
        $st->fetch(¥$result);
        #print "掛け率:$result¥n";

        #データベースから切断
        $db->disconnect();
        return $result;
    } else {
    #データベースに接続できない場合
        print "データベースへの接続に失敗しました<BR>¥n";
    }
}
```

新連載

RPMパッケージ開発講座



Red Hat LinuxやVine Linux、Turbolinuxなど商用ディストリビューションの多くは、アプリケーションプログラムの配布に、RPMパッケージを利用しています。本連載ではRPMパッケージの作成方法を紹介します。

第1回　RPMパッケージの使い方

文：佐藤大輔（densuke@usi.dyndns.org）
アミュレット株式会社・開発部
Text : Daisuke Sato

RPMとは

RPMとはRed Hat Package Managerの略称で、名前の通りもともとはRed HatがLinuxのソフトウェアのインストールやアンインストールを簡単に処理できるようにしたいということから開発されたものです。現在では、名前はそのままRed Hatが使われていますが、www.rpm.org（**画面1**）で、独立して管理されています。

RPMを使うことで、ソフトウェアのインストールやアンインストールが簡単になり、バイナリ形式（コンパイル済みのもの）の配布も手軽になりました。また、ソフトウェアのインストールやアンインストールの際に依存関係というルールを利用してインストールしても動かないことや、不用意なアンインストールでシステムを壊さないようにする防御装置ともなっております。

その反面、RPMが広まりすぎたことによる弊害もあります。なかでもディストリビューション間でのパッケージの互換性が怪しいということです。この部分を反面教師としたのか、Debianのパッケージ形式ではかなり厳密な作成基準になっています。この部分は見習うだけの価値はあると思います。しかしながら、まったく使えないわけではなく、パッケージに若干の修正を加えたうえで再構築してあげればだいたいのパッケージは流用することもできます。

ここではまず、RPMの基本的な使い方を知っていただき、そのうえで実際のパッケージ作成をしてみたいと思います。

なお、今回の説明に用いているディストリビューションはKondara MNU/LinuxのJirai版となっております。そのため、本来のKondaraとはかなり違う部分がありますが、それでもRPMの使い方という部分においての差異はほとんどありません。固有の問題がある場合はその時に補

画面1　www.rpm.orgのWebサイト
http://www.rpm.org/

足させていただきます。

なお、現在Kondara Projectにおいて、最新バージョンであるMary（2.0）の開発が最終段階に入っています。現状で出ている範囲で可能な限り最新版、最新の環境を使えるように調整しているところです（5/26現在、RC1版が配布中です）。本誌が出る頃にはリリースされているかもしれません。リリースされた場合は、次回以降、Mary上での作成にしたいと思います。

RPMパッケージの使い方

まずは既存のRPMパッケージの利用方法をおさらいしておきましょう。RPMパッケージの操作にはrpmコマンドを使います。

rpmコマンドには多くのオプションが用意されています。一度は、オンラインマニュアル（日本語版ディストリビューションなら日本語のオンラインマニュアルも含まれているでしょう）をじっくりと見てください。

ここではそのうちの重要な部分として、以下の4つの利用方法を紹介しましょう。

- **パッケージの確認（-qオプション）**
- **パッケージのインストール（-U／-iオプション）**
- **パッケージのアンインストール（-eオプション）**
- **パッケージの検証（-Vオプション）**

パッケージの確認（-q）

パッケージの確認とは、主に、

Column

パッケージの名前の付け方

パッケージの名前には長い書き方と短い書き方の2つが存在します。長い書き方の場合、下記のようなフォーマットになります。

<ソフトウェアの名前>-<バージョン>-<リリース番号>

■ソフトウェアの名前

配布されているソフトウェア自身の名前ですが、パッケージによっては機能分割をして配布している場合があります。その時は名前の部分にハイフン（-）が付く場合があります。たとえば、tetexパッケージの場合は、

```
$ rpm -qa | grep ^tetex|sort
tetex-1.0.7-13k
tetex-dvips-1.0.7-13k
tetex-fonts-1.0.7-13k
tetex-latex-1.0.7-13k
tetex-xdvi-1.0.7-13k
```

のようになります。この場合、tetex-1.0.7-13kが大元となり、通称メインパッケージ、tetex-dvips-1.0.7-13kなどをサブパッケージと呼びます。

■バージョン

そのソフトウェアの作者が付けているバージョン。基本的には作者の意向にあわせて付けることになりますが、ハイフン（-）を付けることができないため、アンダースコア（_）に変換している場合もあります。また、パッケージを作成した人がなんらかの理由でバージョンを改変する場合も非常に稀ながら存在します。

■リリース番号

この番号はパッケージの作者が付けるもので、同一バージョンに対する何回目のリリースかを示すためのものです。

これらの情報はRPMパッケージのヘッダ部分に入っていて、バージョンの比較をする際に利用されます。RPMは基本的に古いバージョンの同一パッケージはインストールしないという方針で作られているため（当然ですが無効にもできます）、特殊な事情にならないかぎり同一のソフトウェア名で複数のバージョンが同居することはありません。

このことから、ソフトウェア名のみでパッケージ名を特定することができます。たとえば上記のtetexシリーズの場合、それぞれ、

```
$ rpm -qa --qf '%{NAME}\n' | grep ^tetex|sort
tetex
tetex-dvips
tetex-fonts
tetex-latex
tetex-xdvi
```

のようになります。これが短い名前です。

パッケージは、特に規模が大きくなってきた際に、必要ないものまでインストールすることを防ぐためにも、用途ごとに分割することができます。

このように分割されたものをサブパッケージと呼びます。便宜上、大元のものはメインパッケージと呼びます。サブパッケージは、"<メインパッケージ名>-<なんたら>"という形式になり、tetexの場合は、tetex-latexやtetex-xdviが該当します。たいていのサブパッケージは、メインパッケージを必要とするので、単体で入れられるケースはあまり多くありません。

ここで指定している--qfオプションは、rpmパッケージの内部情報を指定したフォーマットにより出力するためのもので、その後のフォーマット文字列に従います。"%{NAME}\n"は「パッケージの名前を表示し、改行する」というものです。

利用可能なフォーマット文字列については、rpmのドキュメントのqueryformatを参照してください。

オプション	説明
-a	パッケージ名をパラメータとして与えるのでなく、インストールされているすべてのパッケージを対象とすることを示す。
-i	パッケージの情報を表示する。パッケージの名前、バージョン、リリースナンバーなどの主な情報が表示される。
-l	パッケージに含まれるファイルの一覧を表示する。
-p	インストールされているパッケージではなく、RPMファイル名をパラメータにして、そのファイルの情報を表示する。
--qf (--queryformat)	パッケージに含まれている内部情報を取得する（例：--qf "%{NAME}\n" でパッケージ名だけを取得するなど）。

表1　-qオプションと一緒に使用するスイッチ（一部抜粋）

・パッケージの概要を知りたい
・パッケージによりインストールされるものを知りたい

という時に用います、もちろんインストールする前のパッケージについても情報が欲しいということもあるでしょう。ほかにも知ることのできる情報はいろいろあります。

このように、情報を知りたい時には-qオプションを使用します、この後に欲しい情報のタイプを示すオプションを付けていくことで、いろいろな情報を得ることができます（**表1**）。

```
[densuke@yuzu rpm]$ rpm -qa
wdic_namazudb-2001_last-0.0002001k
libkmalloc-0.2-5k
edict-V00_001-7k
control-center-devel-1.4.0.1-3k
rpm-build-3.0.6-5k
kondara-release-20001102-3k
perl-Lingua-Romkan-0.14-7k
sawfish-themer-0.38-7k
...
tetex-1.0.7-13k
glibc-devel-2.2.2-9k
tetex-fonts-1.0.7-13k
mozilla-devel-2001041315-0_rh7
gnome-applets-1.4.0.1-3k
kernel-source-2.4.3-9k_mxt
```

画面2　インストールされているパッケージ一覧を得る（-qa）

■インストールされているパッケージ一覧を得る（-qa）

パッケージの概要を知りたい時には、-qで始まるオプションに情報の欲しいパッケージ名を渡します。しかし、システム中にどんなパッケージが入っているのかという情報を知らないのではどうしようもありません。そこで使うのが-qaオプションです。このオプションを使用するとどうなるか、実際に試してみましょう（**画面2**）。

-qaオプションを使うと、インストールされているパッケージの一覧を得ることができますが。ただし見ての通り気を効かせてアルファベット順とかにソートしているわけではないことに注意してください。読みやすくしたいのであれば、ここでsortにパイプで渡してみるといいでしょう。

■パッケージの概要を知りたい（-qi）

パッケージの概要を知りたいと思ったら、そのパッケージ名を-qiオプションと共に渡してみましょう。概要を返してくれます（**画面3**）。いくつかの情報と共に表示してくれますが、そのうち一部を挙げると、

```
[densuke@yuzu rpm]$ rpm -qi tetex
Name        : tetex                         Relocations: (not relocateable)
Distribution: Kondara MNU/Linux (Jirai)          Vendor: (none)
Version     : 1.0.7                              Option: (none)
Release     : 13k                         Build Date: 2001年03月17日 15時32分41秒
Install date: 2001年03月26日 10時41分29秒      Build Host: suwari.kondara.org
Group       : Applications/Publishing      Source RPM: tetex-1.0.7-13k.nosrc.rpm
Size        : 33551958                        License: distributable
URL         : http://www.tug.org/teTeX/
Summary     : The TeX text formatting system.
Description :
TeTeX is an implementation of TeX for Linux or UNIX systems. TeX takes
a text file and a set of formatting commands as input and creates a
typesetter independent .dvi (DeVice Independent) file as output.
Usually, TeX is used in conjunction with a higher level formatting
package like LaTeX or PlainTeX, since TeX by itself is not very
user-friendly.
(以下長いので省略)
```

画面3　パッケージの概要を表示（-qi）

- **Name／Version／Release**
  それぞれパッケージの名前とバージョン、リリース番号
- **Distribution**
  パッケージの対応するディストリビューション
- **Group**
  パッケージのカテゴリ
- **Build date／Install date**
  そのパッケージの作成された時間、インストールされた時間
- **License**
  パッケージの配布条件。（L）GPLやBSDライセンスと書かれる

といった内容があります。内部的にはさらにいくつかの情報が含まれています。興味のある方はRPMの付属ドキュメントを読んでみましょう。

```
[densuke@yuzu rpm]$ rpm -ql tetex
/etc/cron.daily/tetex.cron
/usr/bin/MakeTeXPK
/usr/bin/access
/usr/bin/allcm
/usr/bin/allec
/usr/bin/allneeded
/usr/bin/amstex
/usr/bin/bamstex
/usr/bin/bibtex
/usr/bin/bplain
/usr/bin/dmp
...
/usr/share/texmf/web2c/plain.base
/usr/share/texmf/web2c/plain.fmt
/usr/share/texmf/web2c/plain.mem
/usr/share/texmf/web2c/ptex.fmt
/usr/share/texmf/web2c/ptex.pool
/usr/share/texmf/web2c/tex.fmt
/usr/share/texmf/web2c/tex.log
/usr/share/texmf/web2c/tex.pool
/usr/share/texmf/web2c/texmf.cnf
/var/lib/texmf
```

画面4　インストールされているファイルを表示（-ql）

■パッケージの内容確認（-ql）

パッケージをインストールすることで配置されるファイルの一覧は、-qlに知りたいパッケージ名を渡すことで得ることができます（**画面4**）。

この時に-vをつけておくと、ls -lのような詳細な表示になります（**画面5**）。

■インストール前に情報が欲しい時（-qp）

ここまでの問い合わせ用オプション（-qi／-ql）は、インストールされているパッケージに対してしか使えません。

しかし実際にはインストールする前にダウンロードしてきたファイルの内容を確認したいということもあるでしょう。このような時には-pオプションを一緒に付けてみましょう。-pが付いている時は、パッケージ名でなくファイル

```
[densuke@yuzu rpm]$ rpm -qlv tetex
-rwxr-xr-x     root     root          103  3月 17 15:30 /etc/cron.daily/tetex.cron
-rwxr-xr-x     root     root          437  3月 17 15:30 /usr/bin/MakeTeXPK
-rwxr-xr-x     root     root         4048  3月 17 15:30 /usr/bin/access
-rwxr-xr-x     root     root         3212  3月 17 15:30 /usr/bin/allcm
lrwxrwxrwx     root     root            5  3月 17 15:30 /usr/bin/allec -> allcm
```

画面5　インストールされているファイルの詳細表示（-qlv）

```
[densuke@yuzu i586]$ rpm -qip mozilla-0.8-3k.i586.rpm
Name        : mozilla                      Relocations: /usr
Distribution: Kondara MNU/Linux (Jirai)         Vendor: (none)
Version     : 0.8                               Option: (none)
Release     : 3k                         Build Date: 2001年03月27日 15時48分02秒
Install date: (not installed)            Build Host: suwari.kondara.org
Group       : Applications/Internet      Source RPM: mozilla-0.8-3k.nosrc.rpm
Size        : 26746089                      License: NPL/MPL
URL         : http://www.mozilla.org
Summary     : Mozilla - an open source web browser, mail and news client
Description :
Mozilla is an open-source web browser, mail and news client,
designed for standards compliance, performance and portability.
```

画面6　RPMファイルの内容を確認（-qip）

```
[densuke@yuzu i586]$ rpm -qlp mozilla-0.8-3k.i586.rpm
/etc/X11/applnk/Internet/mozilla.desktop
/usr/bin/mozilla
/usr/lib/mozilla
/usr/lib/mozilla/TestLibxpnet
/usr/lib/mozilla/bloaturls.txt
/usr/lib/mozilla/chrome
/usr/lib/mozilla/chrome/all-locales.rdf
/usr/lib/mozilla/chrome/all-packages.rdf
/usr/lib/mozilla/chrome/all-skins.rdf
/usr/lib/mozilla/chrome/blue.jar
```

画面7　RPMパッケージの中身の一覧を表示（-qlp）

```
[densuke@yuzu i586]$ rpm -qf /bin/ls
fileutils-4.0w-9k
[densuke@yuzu i586]$ rpm -qf /usr/local/bin/ez-ipupdate
ファイル /usr/local/bin/ez-ipupdate はどのパッケージにも属していません
```

画面8　RPMパッケージの逆引き（-qf）

オプション	説明
-i	インストールする時に、同名のパッケージがある場合にはインストールしない。結果的に新規インストールのみ行うことができる。
-U	新規インストールも、既存のパッケージの置き換えも行うことができる。
-F	既存パッケージの置き換えのみを行う。新規インストールは行わない。アップグレードパッケージをパラメータに指定して、インストールされているRPMパッケージだけをアップグレードする場合に使用する。

表2　インストール用のオプション

名を渡すことができます（**画面6**）。

当然ながら、（このファイルからは）まだインストールされていないため、Install dateはありません。同様に-qlpとすることで中身の一覧を得ることもできます（**画面7**）。

■パッケージの逆引き（-qf）

ここまではパッケージを指定することでそのパッケージの情報を得ようとしてきたけど、よくある疑問として、このファイルがどのパッケージに属しているのかを知りたいというものです。この問題に対処できるのが、-qfというオプションです。どのパッケージに含まれているのかを知りたいファイル名をパラメータとして渡してみましょう。

/bin/lsをパラメータとして渡すと、fileutilsパッケージに含まれることが表示されます（**画面8**）。

もし、どのパッケージにも属していない/usr/local/bin/ez-ipupdateというパラメータを指定した場合は、その旨を表示してくれます。もちろん同時に複数のファイルを渡して一気に調べてもらうこともできます。ただ、この場合には渡した順番に返ってくるものの、どのファイルがどのパッケージに属しているかを教えてはくれません。自分の渡した順番と比べて確認するか、個別に調べていくのが適切でしょう。

オプション	説明
-v	インストール作業をしているパッケージの名前を表示する
-h	インストール作業の進行状況をシャープ記号（#）で表示する

表3　インストール用のオプション

パッケージインストール（-i／-U／-F）

パッケージをインストールする時には、-i／-U／-Fのいずれかを使用します。同じことのように見えて、3つには一応違いがあります（**表2**）。

パッケージのインストールの場合、基本的にファイルからインストールするわけで、問い合わせ（-q）の時のようにインストール済みパッケージ相手かファイル相手かを気にする必要はありません、渡すのはあくまでファイルです。しかし、ただ-i／-U／-Fを使うだけでは、実際に作業しているのかが見えにくいため、不安になるかもしれません。そこで通常、**表3**のオプションを一緒に付けておくことになります。

それでは、実際にインストールしてみましょう（**画面9**）。ここでは新規もしくは置き換え（アップグレード）の行える-Uを例にしていますが、ほかのオプションでも同様に行えます。

実は-qオプションは、パッケージ名を渡すことで、イン

ストールされているかを知ることもできます。インストールされていれば、そのパッケージの長い名前を知ることができます。

■パッケージの依存性

RPMパッケージには、依存性という考え方があります。

・特定のパッケージ（＋バージョンを制限に含めることもできる）
・特定のライブラリ
・特定のファイル

が存在しないとインストールできない、もしくは他のパッケージが必要としているパッケージを削除することはできないという仕組みを持っています。しかし、時としてはそれを無効にしてでもインストール／アンインストールしたいという時もあります。

■依存性を無視する

そこで登場するのが--nodepsオプションです。一緒に用いることで依存性を無視してインストール／アンインストールすることができますが、当然ながら依存関係を破壊してしまいます。よほどの事情がない限り使わないようにしましょう。

画面10の例では、libpcapパッケージが必要であることを示しています。libpcapパッケージ（この場合、libpcap-0.6.2-5k.i586.rpm）があれば、それを先に入れればよいものの、どうしてもインストールしてみたいという時には、この--nodepsを併用してみるとうまくいきます。ただし動作は保証しません。

画面11のように、確かにインストールできてしまいますし、etherealコマンドも利用できますが、実際に処理（パケットキャプチャ）させようとするとエラーを返します。この場合、正しくインストールするには、**画面12**のように2つを同時に入れるか、libpcapから個別にインストールする必要があります。

ところで、どのようにして、必要なファイルがlibpcapに含まれているのかを知ることができるのでしょう。残念ながらrpmコマンドの機能の中では直接それを知る方法はありません。今回の場合、必要としていると返されているのが“libpcap”という名前から、あたりを付けて確認が可能です。

しかし、実際のところ表示されるのは、特定のシェアードライブラリ（＊.soなど）の要求です。

このような場合でも、たいていはディストリビューションのバイナリCDに入っているもので対処できるはずなので、パッケージの名前と含まれているパッケージのペアのリストを作成してgrepしてみるのがいいと思います。

```
[root@yuzu i586]# rpm -q zsh
パッケージ zsh はインストールされていません
[root@yuzu i586]# rpm -Uvh zsh-3.1.9-7k.i586.rpm
zsh                 ##################################################
[root@yuzu i586]# rpm -q zsh
zsh-3.1.9-7k
```

-vによるパッケージ名表示
-hによる進行度メータ
存在しているので長い名前で返してくれる

画面9　パッケージインストールの例

```
[root@yuzu i586]# rpm -Uvh ethereal-0.8.14-7k.i586.rpm
エラー: 依存性の欠如:
  libpcapは ethereal-0.8.14-7k に必要とされています
```

画面10　インストール時に必要なパッケージが調査される

```
[root@yuzu i586]# rpm -Uvh --nodeps ethereal-0.8.14-7k.i586.rpm
ethereal                    ##################################################
```

画面11　--nodepsを付けて強制インストールが可能

```
[root@yuzu i586]# rpm -Uvh libpcap-0.6.2-5k.i586.rpm ethereal-0.8.14-7k.i586.rpm
libpcap                     ##################################################
ethereal                    ##################################################
```

画面12　必要なパッケージを一緒に指定すれば普通にインストールできる

たとえば、あるパッケージがlibz.so.1を要求しているとしましょう。まず、**画面13**のスクリプトを実行して、/tmp/listにRPMパッケージに含まれているファイルの一覧表を作ります。あとはこのファイルをgrepで検索してみればいいわけで、zlib-1.1.3-11k.i586.rpmの中にlibz.so.1が含まれていることがわかります（**画面14**）。

■古いパッケージのインストール

パッケージのインストールは、必ずしも新しいものが望む動作をすると限りません。時として、古いパッケージに戻す必要があるかもしれません。この時、通常は古いパッケージをインストールしようとするとエラーになります（**画面15**）。

この場合、rpmが今入れようとしたバージョン（0.5-5k）より新しいバージョン（0.7.99.0-1）が、すでにインストールされていたためにエラーを返して停止しています。もしこのエラーを無視して入れたい時には、--oldpackageというオプションを併用し、古いものであることを明示することで処理できます。なおこのオプション、本当に新しい別のパッケージがあっても処理できるので、古いものと新しいものを混在して処理させても大丈夫です。

■特殊なオプション

このほかにもいくつか重要そうなオプションはありますが、どれも通常は使うべきものではありません、主なオプションを**表4**に挙げておきますので、興味ある方はオンラインマニュアルで調べてみてください。

なんらかの都合で構成しているファイルが消えてしまっ

オプション	説明
--replacefiles	他のパッケージとファイルがぶつかってしまう場合には置換してかまわないことを指示する。
--replacepkgs	インストールしようとしているパッケージのいくつかがすでにインストール済みである場合でもインストールを強行する。
--force	このオプションによって、--oldpackage、--replacefiles、--replacepkgsを指定したことになる。
--test	実際にインストール／アンインストールできるかの確認を行う（いわゆるシミュレーション）。-vや-hと一緒に用いても、パッケージ名の表示や進行度の表示は行わない。

表4　rpmコマンドのそのほかのオプション

```
[densuke@yuzu densuke]$  (for i in *.rpm; do rpm -qlp $i | sed "s%^%$i:%g"; done) > /tmp/list
```

sedのパラメータはシングルクォート(')を使うと変数$iの展開ができなくなるので注意

画面13　カレントディレクトリにあるすべてのRPMファイルのファイル名を保存

```
[densuke@yuzu densuke] $ grep libz.so.1 /tmp/list
zlib-1.1.3-11k.i586.rpm:/usr/lib/libz.so.1
zlib-1.1.3-11k.i586.rpm:/usr/lib/libz.so.1.1.3
zlib-devel-1.1.3-11k.i586.rpm:/usr/lib/libz.so
```

画面14　保存したファイル名リストからファイルを検索表示

```
[root@yuzu i586]# rpm -Uvh gal-0.5-5k.i586.rpm
package gal-0.7.99.0-1 (which is newer than gal-0.5-5k) is already installed
[root@yuzu i586]# rpm -Uvh --oldpackage gal-0.5-5k.i586.rpm
gal                         ##################################################
```

画面15　古いパッケージをインストールするには--oldpackageを付ける

```
[root@yuzu i586]# rpm -q gal
gal-0.5-5k
[root@yuzu i586]# rpm -Uvh gal-0.5-5k.i586.rpm
パッケージ gal-0.5-5k はすでにインストールされています
[root@yuzu i586]# rpm -Uvh --replacepkgs gal-0.5-5k.i586.rpm
gal                         ##################################################
[root@yuzu i586]# rpm -q gal
gal-0.5-5k
```

現在は0.5-5k

バージョンは変わらず

画面16　同一バージョンを再インストールするには--replacepkgsを使う

た時などには、--replacepkgsを使って本来のパッケージを入れ直すことができます（**画面16**）。

パッケージのアンインストール（-e）

パッケージのアンインストールには、-eオプションを使用します。削除したいパッケージ（短い名前で十分です）を指定しておくことで、削除を試みますが、インストールの時と同様に、依存性を壊すアンインストールは簡単にはできません。また、削除できるかどうかわからない時は、一度--testオプションも併用して確認してみましょう。

画面17の場合、使わないのであばetherealパッケージと共にアンインストールすれば問題になりません。しかし、むりやりlibpcapのみ削除したいというのであれば、--nodepsを使うことで依存性チェックを無効化できます。

画面18のように--nodepsを指定すれば、依存性チェックを無効化すれば削除できてしまいました。当然この場合はetherealの動作は保証されません。

なお、-eオプションには、-i／-U／-Vのような作業過程を示す表示を行うためのオプションは用意されていませんのでご注意ください。

パッケージの検証（-V）

たいていのパッケージは、インストールして放っておいても動くもののですが、そうもいかないものもそれなりに存在します。デーモンプログラム（サーバのたぐい）では、設定をあらかじめしておかないといけないでしょう。

また、パッケージの管理をする人が、なんらかの理由でパッケージによりインストールされるスクリプトに手を加えている場合もあります。恐い例としては、クラッキングの被害にあってファイルを書き換えられている場合もあります。これらの例のように、なんらかの理由で本来インストールしたものとファイルの内容が違っているものを検索するのが-Vオプションです。検索したいパッケージを渡して確認できます。

```
[root@yuzu i586]# rpm -V fileutils
```

としてみることで、fileutilsパッケージに含まれるファイルが改ざんされていないかの確認ができます。何も改ざんされていなければ、すぐにプロンプトが表示されます。

実際に改ざん、もしくは変更があると以下のように表示されます。

```
[densuke@yuzu rpm]$ rpm -V junkbuster
S.5....T c /etc/junkbuster/config
S.5....T c /etc/junkbuster/cookiefile
S.5....T c /etc/junkbuster/forward
```

この場合、

S　サイズが（インストール時のものと）違う
5　MD5の値が違う
T　日付が違っている

ことを指摘しています（cは設定ファイルであることを示しているだけ）。もし、fileutilsパッケージを確認して改ざんの記録が出てきた時は、十分注意したほうがいいでしょう。-qlで確認してみればわかりますが、非常に重要な（＝/binに置かれるような）プログラムがいっぱいのパッケージで、まず書き換えられる必要はないからです。クラッキングされてトロイの木馬にされているケースがありますので注意しましょう。

さて、今回はrpmのインストールやアンインストールといった、すでに存在するパッケージの操作に関することを紹介しました。次号からはパッケージの作成手順について解説します。普通にmakeができるのであれば、パッケージの作成はそんなに難しいものでもありません。それではまた次回、お会いしましょう。

```
[root@yuzu i586]# rpm -e --test libpcap
エラー: これらのパッケージを削除すると依存性を破壊します:
        libpcapは ethereal-0.8.14-7k に必要とされています
```

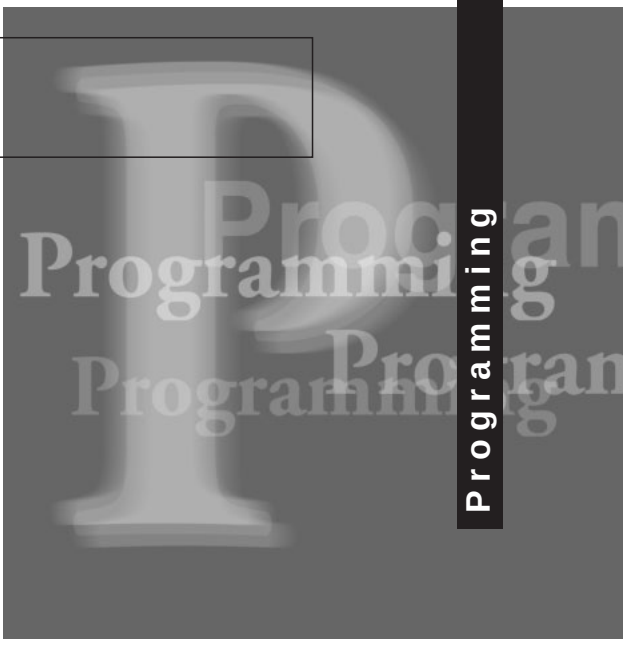
画面17　アンインストールしようとすると依存性のチェックが行われる

```
[root@yuzu i586]# rpm -e --nodeps libpcap
[root@yuzu i586]#  ← 文句を言ってこない
```

画面18　依存性を無視してアンインストールすることもできる

プログラミング工房

先月号ではサーバ管理システムの概要を示した。今月号では、引き続きこのシステムを実現するプログラムについて解説していく。



第19回　サーバ管理プログラム（2）

文：藤沢敏喜
Text : Toshiki Fujisawa

ネットワーク管理者の不幸

多くの会社や学校で日常の連絡に電子メールを使ったり、さまざまなデータファイルを共有したりすることがあたりまえになってきた。ネットワークが停止すると業務や研究が完全にストップしてしまうことも多い。いまやネットワークは、電気やガス、水道、電話と同じように、重要なインフラストラクチャとなっているのである。

しかしながら、ネットワーク管理の重要性に気がついている組織は少なく、ネットワーク管理に適切なリソースが割り当てられないことが多いようである。ネットワーク管理の実担当者には多大な負担がかかり、連日残業をしている管理者も多いのではないかと思う。せめて、ユーザー登録などの単純な作業から解放されて、このような不幸な状況から抜け出せればいいのだが。

Webインターフェイス（登録画面）

さて、今回のサーバ管理システムでは、特別な知識がなくてもユーザー登録などの管理作業が簡単に行えるように、Webインターフェイスを用意している。先月号で説明したように、新規ユーザーを登録する場合はWebブラウザで、

```
http://サーバ名/adm/
```

を指定し、「ユーザーの登録／変更／削除／一覧」の中の「新規登録」を選ぶと、httpdディレクトリにあるcgi-binディレクトリの中のaddというPerlスクリプトが実行される。**画面1**では、あらかじめ登録した共有するディレクトリ名が表示されているが、このディレクトリ名を得るために、add（**リスト1**）では、

画面1　新規ユーザーを登録するためのページ

```
$_ = `sedusr -x cat_file PATH_DIRNAME `;
```

として、sedusrコマンドの引数として「cat_file」というコマンド名と表示すべき内容を示すPATH_DIRNAMEを与えて実行し、「$_」という変数にスペースで区切られたディレクトリ名のリストを代入している。そして、

```
chop($_);          ※改行文字を削除
@dirs = split;     ※スペースで分離（split）する
```

を実行することにより、共有するディレクトリ名のリストをdirsという配列に得ている。

ちなみに、上記のsedusrコマンドの内部では、

```
cat /smb/etc/dir_name.txt
```

を実行しているだけであるので、

```
$_ = `cat /smb/etc/dir_name.txt`;
```

としても同様の結果が得られるが、/smb/etc/のパスを変更したい場合などでも、sedusrコマンドだけを書き換えれ

▼リスト1　新規ユーザー登録ページの生成（add）

```
#!/usr/bin/perl

$ENV{'PATH'} = "/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin";
$\ = "\n";
print 'Content-type: text/html
(省略)
';

$_ = `sedusr -x cat_file PATH_DIRNAME 2>&1`;
chop($_);
@dirs = split;

print '<CENTER>
<HR WIDTH="100%">
<H3>新規ユーザーを登録するためのページ</H3>
<HR WIDTH="100%">
<FORM METHOD=POST ACTION="/cgi-bin/add.cgi">
(省略)
</TR>
</TABLE>
<BR>
<!------------------------------------------------------------------------>
';

if( "@dirs" ne "" ){
    print '【4】許可する共有ディレクトリグループ名にチェックをしてください';
    print '<TABLE BORDER=3 WIDTH="100%">';
    $col = 5;
    for($n=0;;$n++){
        $i = shift @dirs;
        if( $i eq "" ){
            last;
        }
        if( $n % $col == 0 ){ print "<TR>"; }
        print "<TD VALIGN=MIDDLE><INPUT TYPE=CHECKBOX NAME=chk_$i>$i</TD>";
        if( $n % $col == $col-1 ){ print "</TR>"; }
    }
    print '</TABLE>';
}

print '<BR>
<CENTER><INPUT TYPE=submit VALUE=" 登 録 の 実 行 "></CENTER>
</FORM>

<HR WIDTH="100%">
</BODY>
</HTML>';
```

ばすむように、わざわざsedusrコマンドでディレクトリ名を得ている。

また、このページのHTMLでは、

```
<FORM METHOD=POST ACTION="/cgi-bin/add.cgi">
```

を指定しているので、必要な情報を入力してから「登録の実行」ボタンを押すと、cgi-binディレクトリにあるadd.cgiというPerlスクリプトが起動されることになる。

Webインターフェイス（実際の登録）

先ほどのPerlスクリプトaddでは、ユーザーを登録するための情報を入力する欄がフォームとして定義され、CGIへ渡すリクエストメソッドとして“POST”を定義したので、add.cgiには各種の情報が標準入力経由で渡されるこ

▼リスト2　フォームの解析（add.cgi）

```
（省略）
if( $ENV{'REQUEST_METHOD'} ne "POST" ){
    die("not support\n");
}
read( STDIN, $_, $ENV{'CONTENT_LENGTH'} );
foreach $i ( split( /&/, $_ ) ){
    ( $name, $value ) = split( /=/, $i );
    $value =~ tr/+/ /;
    $value =~ s/%([0-9A-Fa-f][0-9A-Fa-f])/pack("C", hex($1))/eg;
    $value =~ s/^ *//g;
    $value =~ s/ *$//g;
    $admpass    = $value if ( $name eq "admpass" );
    $uname      = $value if ( $name eq "uname" );
    $uid        = $value if ( $name eq "uid" );
    $pass1      = $value if ( $name eq "pass1" );
    $pass2      = $value if ( $name eq "pass2" );
    $ename      = $value if ( $name eq "ename" );
    $is_mydir   = $value if ( $name eq "is_mydir" );
    $mydir      = $value if ( $name eq "mydir" );
    if( $name =~ /^chk_/ ){
        $name =~ s/^chk_//;
        $glist = $glist eq "" ? $name : "$glist/$name";
    }
}
$mydir = $is_mydir ne "" ? $mydir : "";

print '<HR WIDTH="100%">';
print '<BR>';
if( $pass1 ne $pass2 ){
    print '「パスワード」が一致しませんでした。<BR>';
    print '<BR>';
    print 'ブラウザの「戻る」を押してやり直してください';
}else{
    &exec( $admpass, $uname, $uid, $pass1, $ename, $mydir, $glist );
}
（省略）
```

Column

バイナリ実行ファイルとセキュリティ

今回のsedusrコマンドはC言語で書かれているので、コンパイルしたあとはバイナリ実行ファイルとなる。

シェルスクリプトやPerlスクリプトの場合、悪意のあるユーザーがその内容を理解したり、一部分を書き換えることは簡単である。しかし、C言語で記述した場合には、ソースファイルなしで部分的に書き換えることは困難であり、セキュリティ的に優れている場合もある。

一方、ソースファイルがないことによりたいへんな問題を引き起こすこともある。たとえば、会社に不満を持つ管理者が、バイナリファイルだけを残して転職してしまうこともあるだろう。そのような場合、期末の忙しいときを狙って「rm -rf /」のような恐ろしいコマンドを実行したり、外部の公開スペースなどへ機密情報をばらまくようなコードが秘かに埋め込まれているかもしれないのである。

とになる。add.cgiスクリプトでは、この情報を解析するために**リスト2**を用いている。

ここでは、「=」で区切られる各フォーム名を解析し、それと同じ名前のPerl変数に代入している。なお、チェックされた共有ディレクトリは、その名前の前に「chk_」を付加してあるので、

```
if( $name =~ /^chk_/ ){
    $name =~ s/^chk_//;
    $glist = $glist eq "" ? $name : "$glist/$name";
}
```

として、$glistに共有ディレクトリのリストを「/」で区切って代入するようにしている。

そして、確認用のパスワードが一致していることを確かめたあとに、exec関数を呼び出している。さらに、exec関数では**リスト3**に示す方法でsedusrコマンドを呼び出している。ここでは、sedusrコマンドを実行した結果の標準出力とエラー出力を「/tmp/sedusr-プロセス番号」という中間ファイルに出力するようにSTDOUTとSTDERRを設定している。そして$cmd変数に、“ sedusr -x add …… ”という文字列をセットし、

```
open(CMD, "| $cmd ") || die("Can't exec $cmd");
printf CMD "$admpass\n";
close(CMD);
$status = $?;
```

としてsedusrコマンドを呼び出している。なお、sedusrコマンドは、スーパーユーザー以外のユーザー（nobody）が実行する場合には、標準入力から管理者パスワードを入力する仕様になっている。このため、上記では管理者パスワードが保存されている$admpassの内容をsedusrコマンドの標準入力へと送っている。

そして、sedusrコマンドの標準出力は、前述した中間ファイル「/tmp/sedusr-プロセス番号」に保存されるた

▼リスト3　ユーザー登録の実行（add.cgi）

```
sub exec
{
    local ( $admpass, $uname, $uid, $pass, $ename, $mydir, $glist ) = @_;
    local $tmpfile = "/tmp/sedusr-$$";
    local $status;

    if( $uname eq "" ){
        print "ユーザー名が空白なので登録できません。<BR>";
        print '<BR>';
        print 'ブラウザの「戻る」を押してやり直してください';
        return;
    }

    $cmd = sprintf("sedusr -x add '%s,%s,%s,%s,%s,%s'",
                   $uname,$uid,$pass,$ename,$mydir,$glist);

    open(SAVEOUT, ">&STDOUT");
    open(SAVEERR, ">&STDERR");

    open(STDOUT, "> $tmpfile") || die "Can't redirect stdout";
    open(STDERR, ">&STDOUT")   || die "Can't dup stdout";

    select(STDERR); $| = 1;     # バッファリングしない
    select(STDOUT); $| = 1;     # バッファリングしない

    open(CMD, "| $cmd ") || die("Can't exec $cmd");
    printf CMD "$admpass\n";
    close(CMD);
    $status = $?;

    close(STDOUT);
    close(STDERR);

    open(STDOUT, ">&SAVEOUT");
    open(STDERR, ">&SAVEERR");
```

め、

```
open(INP, "/tmp/sedusr-$$")
@ans_list = <INP>;
close(INP);
unlink $tmpfile;

$ans = "@ans_list";
$ans =~ s/\n/<BR>\r\n/;
```

として、その出力結果を$ans変数に保存している。

sedusrコマンドが正常終了した場合は、標準出力には何も表示されず、終了ステータス（$status）が0になるので、

```
if( $status == 0 ){
    print "$uname を正常に登録しました。";
    if( $ans eq "" ){
        return;
    }
}
```

という判定により、結果をWebブラウザに表示（HTMLを作成）する。エラーが生じた場合は、そのエラーの内容を**リスト4**のように解析し、Webブラウザに表示することになる。

sedusrコマンドの詳細

さて、Perlスクリプトで書かれたCGIプログラムから、sedusrコマンドがどのように呼び出されるかがわかったところで、今度はsedusrコマンドの詳細について見てみよう。

sedusrコマンドのmain関数（**リスト5**）では、最初にこのコマンドを実行したユーザーのチェックをしている。まず、システムコールgetuid（）により、ユーザーID番号をuidという変数に保存する。sedusrコマンドは、スーパーユーザーがコマンドラインから使用する場合と、CGIプログラムから呼ばれる場合を想定して作成してあるが、CGIプログラムから呼ばれる場合は、“nobody”というユーザーの権限で実行されることが多い。

Vine Linuxでは、nobodyのユーザーIDは99となっているので、

```
#define NOBODY_UID 99
```

として定義してある（CGIプログラムを実行するuidに応じて、この値を書き換える必要がある）。

リスト5の最初のswitch文ではこのuidを判別し、スーパーユーザーの実行またはCGIプログラムからの実行（nobody）以外は拒絶するようにしている。

ところで、sedusrコマンドはユーザー登録などを行うので、パスワードファイルの書き換えなど、root権限が必要な操作を行わなければならない。このため“nobody”ユーザーで実行された場合でも、root権限が得ることができるように実行ファイルにSet User IDビットを立てる必要がある。

▼リスト4　ユーザー登録の実行結果解析

```
open(INP, "$tmpfile") || die("can't open $tmpfile\n");
@ans_list = <INP>;
close(INP);
unlink $tmpfile;

$ans = "@ans_list";
$ans =~ s/\n/<BR>\r\n/;

if( $status == 0  ){
    print "$uname を正常に登録しました。";
    if( $ans eq "" ){
        return;
    }
}elsif( $ans =~ /already regist/ ){
    print "$uname は、すでに登録済みのため再登録できません。<BR>";
    print '<BR>';
    print "$uname を削除してから、やり直してください";
    return;
}elsif( $ans =~ /illegal command line argument/ ){
    print '必要な項目が適切に入力されていません。<BR>';
    print '<BR>';
    print 'ブラウザの「戻る」を押してやり直してください';
    return;
}elsif( $ans =~ /wrong admin passwd/ ){
    print "管理者用パスワードが間違っています。<BR>";
    print '<BR>';
    print 'ブラウザの「戻る」を押してやり直してください';
    return;
}else{
    print '<BR>';
    print '以下の理由により登録ができませんでした。<BR>';
    print '<BR>';
}

print '下記のメッセージをメモして管理者まで御連絡ください<BR>';
print '<BR>';
print '</CENTER>';

if( $ans ne "" ){
    print '<HR WIDTH="100%">';
    print '<LEFT>';
    print '<BR>';
    print $ans;
    print '</LEFT>';
}
```

Set User IDビットが設定されているかどうかは、ls -l sedusrを実行すると、

```
-rws--x--x 1 root wheel 129672 May 5 12:57 sedusr*
```

と表示され、sビットが立っていることで確認できる。sビットが立っている場合は、sedusrコマンドの所有者（この場合“root”）の権限を得ることができる。この権限を得るためのシステムコールがsetuidであり、同様にグループの権限を得るのがsetgidである。**リスト5**では、

```
if( setuid(0) == -1 ){
    fprintf(stderr, "Can't setuid(0)\n");
    exit(1);
}
```

として、スーパーユーザーIDであるゼロを引数としてsetuidを実行することにより、rootユーザーになっている。同様にsetgidも行っている。

そして、いったんrootユーザーになってから、seteuidによって実際に影響を与える権限であるエフェクティブuidとエフェクティブgidを“nobody”に変更している。これは、セキュリティを考慮して、root権限で実行する部分を最小にするためである。

ちなみに、実際にroot権限で実行する必要がある場合には、**リスト6**に示すように“root”ユーザーのIDである0

▼リスト6　エフェクティブUIDの設定

```
static void
set_root_uid(void)
{
    if( seteuid(0) == -1 ){ /* 0 == root */
        fatal("Can't seteuid(0)");
    }

    if( setuid(0) == -1 ){ /* 0 == root */
        fatal("Can't setuid(0)");
    }
}
```

▼リスト5　ユーザー登録の実行結果解析（sedusr.c）

```
int
main(int argc, char **argv)
{
(省略)
    int             uid = getuid();

    switch( uid ){
    case 0: global_is_root_mode = true;
        break;
    case NOBODY_UID:
        global_is_root_mode = false;
        break;
    default:
        fprintf(stderr, "Please run by 'root' or 'nobody'\n");
        exit(1);
    }
    if( setuid(0) == -1 ){
        fprintf(stderr, "Can't setuid(0)\n");
        exit(1);
    }
    if( setgid(0) == -1 ){
        fprintf(stderr, "Can't setgid(0)\n");
        exit(1);
    }
    if( setegid(NOBODY_GID) == -1 ){
        fprintf(stderr, "Can't setegid(nobody)\n");
        exit(1);
    }
    if( seteuid(NOBODY_UID) == -1 ){
        fprintf(stderr, "Can't setuid(nobody)\n");
        exit(1);
    }

    process_enter(); /* make lock file by "nobody" */
    setup_stderr();
    setenv("PATH", "/bin:/usr/bin:/usr/sbin:/usr/local/bin", 1);
```

を引数として、seteuid（0）とsetuid（0）を呼ぶ。

さて、動作する権限を設定したあとは、

```
process_enter(); /* make lock file by "nobody" */
setup_stderr();
setenv("PATH", "/bin:/usr/bin:/usr/sbin: ……
```

によって、

- **ロックファイルの作成**
- **標準エラー出力の設定（ステップアップC言語参照）**
- **環境変数PATHの設定**

▼リスト7　ロックファイルの作成

```
static void
process_enter(void)
{
    int    retry_sec = 3;

    while( mkdir(DIR_LOCK,0755) == -1 ){
        sleep(1);
        if( retry_sec-- == 0 ){
            printf("Can't lock(mkdir) %s,
                        try again\n", DIR_LOCK);
            process_exit(1);
        }
    }
}
```

を行っている。

process_enter（）は、このプロセス（sedusr）が開始されるときに1度だけ実行される関数である。この関数では、**リスト7**に示すようにロックファイルを作成しているが、これは多数のWebクライアントから同時にsedusrコマンドが実行されることを防ぐためである。このロックファイルは、プロセスの終了時に呼び出されるprocess_exit関数の中で消去されることになる。

また、CGIから実行された場合でも、sedusrコマンドから適切に外部コマンドが呼び出せるように、環境変数PATHを再設定している。

今月のまとめ

今月号では、WebインターフェイスからPerlで書いたCGIプログラムによってsedusrコマンドを呼び出す部分と、このシステムの中核をなすsedusrコマンドの起動部分のプログラムについて解説した。今回説明したSTDOUT、STDERRの扱いやsetuidによる実行ユーザーの変更は、ユーザー管理プログラムを作るうえで欠かせないテクニックである。

来月も、引き続いてこのシステムについて取り上げて行きたい。

Column

暗号化パスワード

先月号でも述べたように、多くのLinuxディストリビューションに用意されているuseraddコマンドは、ユーザーの登録時に「-p」オプションを用いて、コマンドラインからパスワードを設定できるようになっている。ここで指定するパスワードは、暗号化されたパスワードでなければならない。先月号のバージョンでは、Linux上でのユーザー登録の際に、パスワードを正しく設定することができなかったが、今月号のCD-ROMに収録したバージョン0.02では、crypt関数を使って正しくパスワードが設定されるように修正してある。

この修正は、生のパスワードをcrypt関数を用いて暗号化してから、-pオプションの引数として渡すことにより実現されている。

なお、パスワードの暗号化には、DESやMD5などいくつかの方法があるが、ここではMD5を用いている。これらについてより詳しく知りたい場合は、http://www.linux.or.jp/JF/にあるユーザー認証（PAM）やシャドウパスワードの解説、そしてcryptシステムコールのオンラインマニュアル（man 3 crypt）を参照するとよいだろう。

さて、MD5を用いる場合、char salt[35]というバッファを用意し、

```
char chars[] =
"0123456789abcdefghijklmnopqrstuvwxyz"
    "ABCDEFGHIJKLMNOPQRSTUVWXYZ.";

salt[0] = '$';
salt[1] = '1';
salt[2] = '$';
for (i = 3; i < sizeof(salt)-1; i++){
    salt[i] = chars[ rand() % 63 ];
}
salt[i] = '\0';
```

というようにsaltを設定する。このsaltを用いて、

```
epasswd = crypt(生パスワード, salt);
```

とすると、epasswdにMD5で暗号化されたパスワードを得ることができる。ちなみに、saltは同一の生パスワードが与えられても、違った暗号化パスワードが得られるように「調理」するための「調味料」である。そのため、この調味料は毎回違った味になるように乱数で初期化している。

なお、Linuxではnewusersというコマンドがあり、このコマンドでは標準入力経由で生パスワードを受け渡すことができる。cryptを使ってuseraddコマンドを呼び出す代わりに、このコマンドを呼び出すようにしてもよいだろう。

特別講座

ステップアップC言語

標準入出力とファイルディスクリプタ

■標準出力と標準エラー出力

いわゆるUNIX環境には、標準出力と標準エラー出力という概念がある。シェルではパイプを使ってデータを受け渡すことが多いが、エラーが生じたときにはデータを渡すパイプにエラーメッセージを表示しても見ることができないため、エラーメッセージは「標準エラー出力」に送られるのである。

標準出力は1番、標準エラー出力は2番に割り当てられるので、たとえば、

```
int main(void)
{
    write(1/*番*/,"stdout\n",7/*文字*/);
write(2/*番*/,"stderr\n",7/*文字*/);
    return 0;
}
```

というプログラムをコンパイルして、実行ファイルtest-1を作成し、

```
./test-1 | cat -n
```

として、「cat -n」で行番号を付加するようにすると、

```
stderr
    1  stdout
```

という結果が得られる。stderrという文字列は標準エラー出力（2番）に送られるので、catコマンドには渡らないのである。

一方、

```
./test-1 2>&1 | cat -n
```

を実行すると、標準エラー出力（2番）は、標準出力（1番）に送られるため、

```
    1  stdout
    2  stderr
```

という出力が得られる。

■dup関数

ここで、次のようにプログラムの中から上で示したtest-1を呼び出すことを考えてみよう。

```
int main(void)
{
    return system("./test-1");
}
```

このプログラムで、「./test-1 2>&1」としないで、標準エラー出力を標準出力に送るには、どのようにしたらよいだろうか？

答えは、次のプログラムである。

```
#define STDOUT 1
#define STDERR 2

close(STDERR);
if( dup(STDOUT) != STDERR ){
    fprintf(stderr,"Can't dup");
    exit(1);
}
return system("./test-1");
```

上記では、まず最初にcloseシステムコールを呼んで、標準エラー出力（2番）のファイルデスクリプタをクローズし、その後にdupシステムコールを呼ぶ。

dupシステムコールは、その引数のファイルディスクリプタをデュプリケート（コピー）して、新しいファイルディスクリプタを作成するシステムコールである。このシステムコールの戻り値は、新しく作成されたファイルディスクリプタ番号であるが、この番号は小さい順に使われることになっている。ファイルディスクリプタの0番は標準入力であり、1番は標準出力である。そして、2番が標準エラー出力であるが、2番は先ほどのcloseシステムコールにより、閉じられて空いている状態である。

したがって、上記のdup（STDOUT）を実行したときの戻り値は必ず2番となり、無事に標準出力（1番）が標準エラー出力（2番）に割り当てられることになる。

■dupシステムコールの応用

dupシステムコールを応用すると、パスワードを標準入力から読み込むプログラムに特定のファイルを入力として与えることができる。たとえば、

```
echo fujisawa:NewPass > /tmp/inp
```

として“/tmp/inp”を用意し、次のプログラムを実行すればよい。

```
#include <stdio.h>
#include <fcntl.h>
#define STDIN 0
int main(void)
{
   int     fd_save;
   int     fd_inp;

   fd_save = dup(STDIN);    /*保存*/
   close(STDIN);

   fd_inp = open("/tmp/inp", O_RDONLY);
   if( fd_inp != STDIN ){
     fprintf(stdout, "open err\n");
     exit(1);
   }
   system("chpasswd");

   close(STDIN);
   dup(fd_save); /*保存したfdを回復*/
}
```

なお、ここでは標準入力STDINの内容を保存するようにしてあるので、外部コマンドを実行した後でも、元の標準入力からの読み込みを行なうことができる。

短期集中連載

Rubyで作るWeb日記システム

『Rubyで行こう』が先月で終わり、みなさんもRubyで何か作りたくなってきた頃でしょう。今月から2回にわたって、CGIを使ったWeb日記システムを作ります。

日記更新システムの作成（前編）

文：ただただし
Text : TADA Tadashi

RubyでCGIする前に

読者のみなさんの多くは、おそらくISPやレンタルサーバ業者からWebサーバの一部を借りて、自分のWebサイトを運営されているでしょう。最近は常時接続環境で自前のLinuxマシンを直接インターネットに接続している方も多いかもしれませんが（そういう方はここは読み飛ばして結構です）、そうでない場合はまず、そのWebサーバにRubyが入っていなければ話になりません。

実はこれがこの記事で最大のハードルかもしれません。CGIを動かせると謳っているISPでも、使えるのはPerlだけ、というところが少なくありません。というか、ほとんどがそうでしょう。Rubyコミュニティでは、Rubyが使える（良心的な）ISPをリストアップする活動がありますが（http://Web.jin.gr.jp/~nahi/Ruby/anywhere.html）、その数はまだまだごく少数というのが実際のところです。以前はISPにRubyを入れるように要請しても断られることもあったようです。

ただ、この状況は少しずつ改善されてきていると考えていいでしょう。Rubyのパッケージが標準で入っているLinuxディストリビューションはずいぶん増えてきましたし、書店に入ってみればわかるように、Rubyの書籍は毎月1冊（！）の勢いで出版されています。この夏にはついに、アメリカで初のRuby専門のカンファレンスが開催される予定です（http://Web.rubyconf.org/）。これほどメジャーになった以上、ISPにRubyを導入させない理由はもはやなくなったと思います。むしろ入っていて当然と言ってよいくらいです。

というわけで、RubyでCGIする前にまずすべきことは、あなたが使っているWebサーバにRubyが入っていることを確認すること、そして万が一入っていなかった場合は胸を張ってRubyを導入するように要請することです。健闘を祈ります。もし要請が聞き入れられなかったら？　そんなISPはとっとと乗り換えてしまいましょう。良貨は悪貨を駆逐しなくてはいけません ;-）

Apacheの設定

それでは首尾よくRubyの入っているWebサーバを確保したとしましょう。そのWebサーバにはまず間違いなくApacheが使われているはずですから、そのセンで話を進めます。ISPがCGIの実行を許可している場合、その設定にはいくつかのパターンがあります。利用しているISPの説明をよく読んで、Rubyで作成したCGIを実行できるように設定しましょう。

■特定のディレクトリに置いたファイルがCGIとして実行できるようになっている場合

ホームディレクトリにcgi-binというディレクトリがあっ

て、その下に置いた実行可能属性の付いたファイルがCGIとして実行できる場合が当てはまります。ディレクトリ名や場所が異なっている場合もあるでしょう。特にファイルのサフィックス（拡張子）に関する指定がない場合はこれだと思って良いと思います。この場合は特に何もする必要はありません。これから作るRubyスクリプトを指定されたディレクトリに置き、実行可能属性を付けるだけです。ファイルの先頭行には以下のように、「#!」に続いてそのWebサーバ上のrubyコマンドのパスを書きます。

```
#!/usr/local/ruby
```

■特定のサフィックスだけがCGIとして実行できる場合

特にディレクトリの指定がなく、そのかわりファイルのサフィックスを特定のものにしなければいけない場合です。たいていの場合、サフィックスには「.cgi」が指定されているでしょう。通常、Rubyスクリプトのサフィックスは「.rb」ですが、この場合には「.cgi」に変更する必要があります。サフィックスに「.rb」が付いたCGIで「このCGIはRubyで動いているんだぜ！」と主張することができないのは残念ですが、あきらめましょう。記事中の「.rb」は適宜読み替えてください。

■自分で好きなように設定できる場合

自分で「.htaccess」ファイルを書いて、好きな場所、好きなサフィックスを設定できる場合です。上記の2つの場合にもこの手を使って設定を変更できるISPもあるかもしれません。たとえば、サフィックス「.rb」のファイルを実行できるようにする場合には、**リスト1**のような内容の「.htaccess」というファイルを、CGIスクリプトを入れたいディレクトリに置いておけばOKです。この例では、以下のような設定をしています。

- **CGIの実行を許可する**
- **サフィックス「.rb」の付いたファイルをCGIとして実行可能にする**
- **「index.rb」というファイルがあればそれをディレクトリのデフォルトファイルとする**

▼リスト1　.htaccessの例

```
Options ExecCGI
AddHandler cgi-script .rb
DirectoryIndex index.rb
```

もちろんこのディレクトリはWebからアクセスできなければいけません。スクリプトに実行可能属性を付けたり、最初の行にrubyのパスを記述しておかなければならないのは、他の場合と同様です。

Web日記システムとは?

それでは、ようやく本題であるWeb日記システムを作り始めるとしましょう。

Web上で日記を公開するのが流行り始めたのは、かれこれ数年前になると思います。それほど工夫をしなくても毎日ホームページを更新できるから、というのがその流行の理由ではないかと筆者は考えていますが、次第に内容が面白かったり文章がうまかったりする「日記ライター」が多くの読者を得るようになります。さらには日記サイトへのリンクを集めて、アクセス数でランキングを競うサイトまでできるようになりました。

そうこうするうち、妙な現象が起き始めます。日記ライターどうしが、お互いの日記を文章中でリンクしあうことでコミュニケーションを取り始めたのです。ある人の日記に書かれていたことに対して自分の日記の中でコメントし、それに対して元の日記で返事が書かれるというようなことです。本来、きわめて個人的なコンテンツであったはずの日記が、Webという場を得て大きく変貌してしまいました。この、いわば「ツッコミ」を補助するために生まれたのが「段落アンカー」と呼ばれる、日記の段落ごとに付けられたリンク先のタグです。これである人の日記の特定の段落を直接リンクすることができるようになり、「日記コミュニケーション」は格段に取りやすくなりました。

最初はHTMLを自分で書いていた日記ライターたちですが、このような機能を自分の日記に埋め込むのはなかなか面倒です。必要は発明の母、そういうことを補助するシステムが生まれました。いわゆるWeb日記システムです。現在では、かなりの数のオリジナリティあふれるシステムが書かれています。Rubyで書かれたシステムもあります。実は筆者も日記を公開しているのですが、なかなか気に入ったものが見つからず、まだ日記システムを使っていません。しかし毎日HTMLを書くのにもそろそろ疲れてきたので、この記事を使って自分の日記システムを作ってしまおう、というもくろみです:-)

基本的な仕様は以下のとおりです。

1. フォームに入力することで日記の更新ができる。

2. HTMLを意識しなくても書けるが、HTMLタグを使うこともできる。
3. 段落アンカー等、日記システムの「デファクトスタンダード」に準拠する。

1.の機能は当たり前だと思う方があるかもしれませんが、実はかなり多くの日記システムでは、別に用意したファイルを（FTPなどで）Webサーバに置くことで日記の更新を行います。つまり日記の作成ではなく、閲覧を補助するシステムというわけです。しかし今回作るシステムでは、Webブラウザさえあればどこからでも日記がつけられるようにしたいと思います。

2.は、HTMLをあまり知らなくても日記をつけられるようにしたいと考えて採用しました。ただ、文字を大きくしたり色をつけたりといった装飾や、リンクを埋め込むためのHTMLタグまでは制限しません。日記システムによっては完全に専用のフォーマットを採用しているものも少なくありませんが、そこまではしないことにします。

3.のスタンダード準拠は、日記コミュニケーションをサポートする以上、当然でしょう。このほかにも、最新の数日分を逆順に表示する機能や、過去の日記を読む機能、段落ごとのサブタイトルのような、今日びの日記システムに求められる最低限の機能は実装します。余力があればオリジナルの機能も考えてみましょう。

おっと、忘れていました。プログラミングでもっとも重要な最初の一歩、「名前」です。このシステムの名前は「tDiary」とします。単に筆者の頭文字を付けただけなのですが、実はもうひとつの意味があったりします。それは次回に明かしましょう。

画面1　更新用フォーム

まずは更新用フォーム

最初に、日記を更新するのに使うフォームを作ります（**画面1**、**リスト2**）。日記の日付とその日のタイトル、そして本文というシンプルな構造です。日記にタイトルというのも変な話ですが、これもWeb日記のスタンダードなので入れられるようにしておきます。もちろん入力しなくてもよいことにします。

さて、本文の欄に日記を記すわけですが、サブタイトルや段落アンカーをサポートするために少しルールを設けましょう（**リスト3**）。

段落アンカーをどの単位につけるかが問題ですが、このシステムではサブタイトルが付く単位でアンカーをつけるようにします。改行ごと（つまり本来の段落）ごとにアンカーを付けるシステムもありますが、本来、特定の話題をポイントするのが目的のアンカーですから、これで十分ということにしましょう。また、話題ごとにサブタイトルを付けるのが面倒な場合は、それも可能にしてあります。

また、HTMLの断片をそのまま埋め込みたい場合もあるでしょう（リストや表を挿入したい時など）。そのために、最初の行が「<」で始まっている場合には次の空改行まで段落処理を無視することにします。

さて、まだ1行もRubyスクリプトが出てきていませんね。その日の日記を書いて、「追加」ボタンを押したあと、

▼リスト2　更新用フォームリスト（update.html）

```
<html>
<head><title>更新</title></head>
<body>
<h1>更新</h1>

<form method="post" action="./">
<p>
<input name="year" size="4">年
<input name="month" size="2">月
<input name="day" size="2">日
</p>
<p>タイトル: <input name="title" size="40"></p>
<p>本文:<br>
<textarea name="body"cols="60" rows="15">
</textarea></p>
<p><input type="submit" value="追加"></p>
</form>
</body>
</html>
```

やっとRubyスクリプトの出番がやってきます。

日記を管理するクラス

入力された日記を構造化して保存するために、まずこのシステムのデータを管理するクラスを作りましょう。ある日の「日記」は複数の「段落」で表現され、その日記が複数集まってなんらかの保存単位になるはずです。そこで「段落」をParagraphクラス、「日記」をDiaryクラスとして表現します（**リスト4**、**リスト5**）。

Paragraphクラスは単純に、段落本文とサブタイトル（あれば）を保持するクラスです。initializeでは、空改行で分割された日記本文の断片（fragment）を受けとり、最初の行が空白文字かHTMLタグの開始を表す「<」で始まっていなければサブタイトルと見なし、インスタンス変数@subtitleに保存します。その他の部分はそのままの形で@bodyというインスタンス変数に保存します。これらはattr_readerを使って、それぞれsubtitle、bodyというメソッドで取り出せるようにしました。

Diaryクラスは、フォームから入力するデータである、日付、タイトル、本文を管理するクラスで、initializeでそれらのデータを受けとっています。日付はTimeクラスのインスタンスを前提としています。日付とタイトルはそれぞれインスタンス変数@date、@titleに保管され、Paragraphクラスと同じように、date、titleメソッドで取り出せるようになっています。@titleについては、title=メソッドで置き換えも可能です。

Diary#replaceメソッドは日記の内容をごっそり置き換えるためのものです。@date、@titleを置き換え、段落を保持する@paragraphs配列を初期化します。appendメソッドは与えられた日記本文を段落に分けて、@paragraphsに追加するメソッドです。本文を空改行（"\n\n"）で分割し、Paragraphインスタンスを生成しています。

Diary#each_paragraphメソッドは、この日の日記が保持している段落を数え上げるイテレータです。ブロックパラメータには、Paragraphインスタンスが渡ります。@paragraphsをそのまま見せてしまってもよかったのですが、実装を隠すためにこのようにしてみました。

日記をファイルに保存する

これで「ある日の日記」を保持することができるわけですが、これをファイルに保存すると、たとえば1カ月分の日記を表示するためには30個前後のファイルを開かなければなりません。これではあまりに非効率なので、1カ月分の日記を1つのファイルに保存することにします。保存形式にはRubyのデータ構造をそのまま読み書きできるPStoreを使うことにしました。

PStoreはRubyに標準添付されているため別途インストールの手間がなく、またRubyだけで書かれているのでISP上で使う時に面倒が少ないのも利点です（DBMのような拡張ライブラリは利用できないことがあります）。PStoreはファイルをいったんすべてメモリに読み込むので、大量のデータ処理には向きませんが、1カ月分の日記はせいぜい100Kバイト程度のサイズですから、大丈夫でしょう。

この「1カ月分」の日記を管理して、ファイルへの入出力を行ったり、CGIスクリプトへのさまざまなサービスを行うクラスとして、TDiaryクラスを作ります（**リスト6**）。最初にpstoreをrequireして、PStoreクラスを使えるようにしてあります。

initializeのパラメータcgiは、CGIクラスのインスタン

▼リスト3　日記フォーマット

```
サブタイトル1
  ここに本文を書きます。行頭に空白のある文章は、改行までをひとつのブロックとして認識されることにします。
  改行が入ると、次のブロックになります。段落アンカーは、サブタイトルの先頭にのみ入ります。

サブタイトル2
  空改行を挟むと、新たなアンカーが埋め込まれた段落が始まります。

  サブタイトル抜きで書きたいときは、このように空改行の後の行を空白から始めればそのまま段落の開始になります。サブタイトルとは、太字で表示されるかどうか程度の違いしかありません。同様に行頭にはアンカーが埋め込まれます。

<pre>
このように、段落の最初の行がHTMLタグで始まっていた場合、なにもせずにそのまま表示することにします。もちろんアンカーも付けません。
</pre>
```

スです。これは@cgiに保存してあとで利用します。Hashのインスタンスとして@diariesを用意し、日付単位に日記を保持できるようにします。Hashのキーには、日付を「YYYYMMDD」の形式に文字列化したものを使います。

続いて@diariesから特定の日付の日記を取り出すための[]メソッド、追加するための<<メソッド、削除するためのdeleteメソッドが用意されています。それぞれHashである@diariesを操作する単純なメソッドですが、追加もしくは削除された場合には@dirtyというインスタンス変数をtrueにし、データに変更があったことを記録しています。

最後のtransactionが、PStoreを使ってファイルからの日記の読み出しと保存を行うメソッドです。与えられた日付からファイル名を生成して（「data/YYYYMM」、つまりdataディレクトリの下に置きます）、「diary」というキーの付いたオブジェクトを取り出して@diariesに設定しています。PStoreはこのキーがなかった場合、PStore::Errorという例外を発生しますから、まだ日記がひとつも書かれていない場合にそれをrescueで捕捉して、@diariesを初期化しています。

これで既存の日記データの読み込みが終わりましたから、yieldを使って呼び出し元のブロックに制御を渡します。transactionメソッドをこのようにブロック付きメソッドにすることで、PStore#transactionメソッドの呼び出しを1回で済ますことができます。transactionメソッドの呼び出し元では[]や<<、deleteメソッドを使って日記データを操作します。この時、@dirtyが操作されますから、yieldの次の行では変更があった時のみPStoreに対して更新処理を行っています。

▼リスト4　Paragraphクラス

```
class Paragraph
        attr_reader :subtitle, :body

        def initialize( fragment )
                lines = fragment.split( /\n+/ )
                if lines.size > 1 then
                        if /^[   <]/ !~ lines[0]
                                @subtitle =
lines.shift.chomp
                        end
                end
                @body = lines.join( "\n" )
        end
end
```

▼リスト5　Diaryクラス

```
class Diary
        attr_accessor :title
        attr_reader :date

        def initialize( date, title, body )
                replace( date, title, body )
        end

        def replace( date, title, body )
                @date, @title = date, title
                @paragraphs = []
                append( body )
        end

        def append( body )
                body.gsub( "\r", '' ).split( /\n\n+/ ).each do |fragment|
                        paragraph = Paragraph::new( fragment )
                        @paragraphs << paragraph if paragraph
                end
                self
        end

        def each_paragraph
                @paragraphs.each do |paragraph|
                        yield paragraph
                end
        end
end
```

日記の更新を行うクラス

日記を読み出し、保存するコードが準備できたので、続いてようやくCGI本体にとりかかることができます。さきほど作ったフォームに日記を入力して「追加」ボタンが押すと、ブラウザはWebサーバにその情報を送ります。Webサーバは指定されたCGIスクリプトを子プロセスとして実行し、フォームに入力された情報を渡します。CGIスクリプトはその情報を受けとって解釈しなければならないわけですが、基本的な処理はRubyに付属のcgi.rbを使います。

cgi.rbが定義しているCGIクラスは、CGIプログラミングに便利な非常に充実したメソッドを多く持っています。今回はこのうち、入力処理の省力化をしてくれる機能を使います。まず、日記の更新処理専用のクラスTDiaryAppendを、Tdiaryを継承して作りました（**リスト7**）。

initializeではパラメータとして与えられたCGIクラスのインスタンス（これは呼び出し元で生成します）を使って、年、月、日を取り出し、Timeインスタンスを生成しています。この@dateは、dateメソッドで外部から取り出せるようにattr_readerで指定してあります。また、タイトル、本文も@cgiから取り出しています。

CGIクラスを使うとこのように、以下の形式でフォームに入力されたデータを取り出すことができます。

```
cgi［HTMLのname属性値］[0]
```

▼リスト6　TDiaryクラス

```
require 'pstore'

class TDiary
        def initialize( cgi )
                @cgi = cgi
                @diaries = {}
        end

        def []( date )
                @diaries[date.strftime( '%Y%m%d' )]
        end

        def <<( diary )
                @diaries[diary.date.strftime( '%Y%m%d' )] = diary
                @dirty = true
        end

        def delete( date )
                @diaries.delete( date.strftime( '%Y%m%d' ) )
                @dirty = true
        end

protected
        def transaction( date )
                PStore::new( date.strftime( 'data/%Y%m' ) ).transaction do |db|
                        @dirty = false
                        begin
                                @diaries = db['diary']
                        rescue PStore::Error
                                @diaries = {}
                        end
                        yield
                        db['diary'] = @diaries if @dirty
                end
                File::delete( filename ) if @diaries.empty?
        end
end
```

最後の［0］は、CGI#［］の返り値がArrayとして返ってくるため、その最初の要素を取り出しているからです。HTMLでは同一フォーム中に同じname属性値を持ったフィールドを複数持たせられるので、CGI#［］だけでは単一のフィールドを特定できません。そのためにこのようになっているわけですが、今回使うフォームではすべてのフィールドに別々のname属性を与えていますから、［0］を決め打ちで使っています。

続くtransactionメソッド呼び出しは、先ほど作ったTDiaryクラスのものです。ブロックの中では、まずself［@date］で同じ日付の日記がすでにあればそれを取り出し、なければ新しいDiaryインスタンスを生成して、@diaryに代入します。続いてその@diaryに対して新しい日記本文を追加しています。タイトルは空でない場合のみ置き換えるようにしました。これで、日に何度も日記を更新するような場合には、本文だけが追加されていくようにできます。

最後に、これらの処理全体をbegin～rescue～endで囲み、日付の生成処理がうまくいかなかった場合（年月日に誤った値が指定された場合など）に、@dateをnilにしています。本当は@bodyが空でないかどうかなどもチェックすべきでしょう。

日記更新CGI本体

ここまで作ってきたParagraph、Diary、TDiary、TDiaryAppendクラスを1つのファイルtdiary.rbに入れてしまいましょう。これらのクラスは今後、複数のCGIスクリプトから共通に利用されるものです。その上で、Webサーバから起動されるCGI本体、index.rbを作ります（**リスト8**）。

最初のほうでcgi、tdiaryをrequireして、今まで書いてきたコードを利用可能にしています。続いてCGIインスタンスを生成し、それを与えてTDiaryAppendインスタンスを生成しています。この時点で、フォームからの入力がすべて処理され、日記のデータの更新も終わっているはずです。あとは結果を表示すれば、日記の更新処理は終わりです。

CGIでは、結果となるHTMLを返す前に、HTTPヘッダを返さなければいけません。このヘッダ生成の処理も、CGIクラスを使うことで行えます。以下の行がそれです。

```
print @cgi.header( 'type' => 'text/html', 'charset' => 'EUC-JP' )
```

ここでは、Content-Typeヘッダに「text/html; charset=EUC-JP」を設定しています。CGI#headerメソッドは、パラメータを追加することでContent-LengthやCookieなど、HTTPヘッダを自由に設定することが可能です。このメソッドの結果をprintすることで、HTTPヘッダの処理は完了です。

あとは結果をWebブラウザに返すだけです。今回は更

▼リスト7　TDiaryAppendクラス

```
class TDiaryAppend < TDiary
        attr_reader :date

        def initialize( cgi )
                super
                begin
                        @date = Time::local( @cgi['year'][0].to_i, @cgi['month'][0].to_i, @cgi['day'][0].to_i )
                        @title = @cgi['title'][0]
                        @body = @cgi['body'][0]

                        transaction( @date ) do
                                @diary = self[@date] || Diary::new( @date, @title, '' )
                                self << @diary.append( @body )
                                @diary.title = @title if not @title.empty?
                        end
                rescue NameError, ArgumentError
                        @date = nil
                end
        end
end
```

▼リスト8　index.rb

```
#!/usr/bin/ruby -Ke

require 'cgi'
require 'tdiary'

@cgi = CGI::new
tdiary = TDiaryAppend::new( @cgi )

print @cgi.header( 'type' => 'text/html', 'charset' => 'EUC-JP' )

if tdiary.date then
        diary = tdiary[tdiary.date]
        print <<-PART1
                <html>
                <head><title>更新完了</title></head>
                <body>
                <h1>更新完了</h1>
                <h2>#{diary.date.strftime( '%Y年%m月%d日' )} #{diary.title}</h2>
        PART1
        diary.each_paragraph do |p|
                print "<h3>#{p.subtitle}</h3>" if p.subtitle
                print "<p>#{p.body.sub( /\n/, '<br>' )}</p>"
        end
        print <<-PART2
                </body>
                </html>
        PART2
else
        print "error"
end
```

画面2　更新完了画面

新した日付の日記データをすべて表示して、更新結果が正しいことを確認できるようにしましょう。CGIクラスにはHTMLを生成するメソッドが用意されていますが、今回はヒアドキュメントを使ってHTMLを生成することにします。

TDiaryAppend#dateがnilでない場合という、日記の更新に成功した条件のもとで、tdiaryから更新のあった日付のDiaryインスタンスを取り出します。続いて「PART1」で挟まされたヒアドキュメントをprintします。このパートでは、diaryから取り出した日付とタイトルを表示しています。

次の部分はdiaryが持つすべてのParagraphインスタンスを表示しています。この部分はイテレータを使ったループで、複雑になるのでヒアドキュメントは使っていません。サブタイトルがある場合のみ<h3>タグで出力し、本文は<p>タグで囲んで出力します。

最後に「PART2」で囲んだ2つめのヒアドキュメントを出力して、完了です（**画面2**）。

次回は……

日記を更新できたところで紙数が尽きてしまいました。あとは日記の読者のために、適当なフォーマットで日記を表示するだけです。この仕組みにはeRubyを使ってみようと思います。

また、tDiary独自の機能として、読者が日記にツッコミを入れられる簡易掲示板機能など、次号では今回作ったCGIの機能を拡張していくことにしましょう。

Perlスクリプト入門
すぐにできる実践Perl

今回は文字列を操作する方法について解説します。また、CSVファイルの読み書きについても解説しましょう。さらに、文字列や数値を加工して表示する方法などについても解説します。

第5回　文字列をマスターする

文：おもてじゅんいち／かざぐるま
Text : Junich Omote/Kazaguruma

Perlが特に文字列処理に向いているということは、これまでにも何度か触れてきました。今回は、その「文字列」についてじっくり学んでみましょう。

文字列とクオート

文字列とは、正確にいうと「0文字以上の文字の並び」ということになります。「1文字以上」ではなく、0文字も含まれているのは、0文字のときにも「空文字列」という文字列の一種として扱ったほうが都合がよいからです。

以下のようなものが、文字列データになります。

```
"a"
"あいう"
"あいう漢字123"
```

このように、スクリプト上でそれが文字列データであると指定するためには、' '（シングルクオート）や、" "（ダブルクオート）で囲みます。

正規表現のときにも少し触れましたが、' 'や" "は、

```
q{ }        （シングルクオートと同じ）
qq{ }       （ダブルクオートと同じ）
```

と書くこともできます（Perlバージョン5以降）。

これは、文字列中に' 'や" "を含める場合、いちいち \ でエスケープすると見にくくなったり、わかりにくくなりそうなときに使えば便利です。このような働きをするクオート演算子をまとめてみます（**表1**）。

変数展開の欄に○印があるものは、クオート内で$から始まる変数名が書かれていれば、その変数の内容に置き換えられ、さらに \ から始まるメタ文字が使えるということです。

メタ文字

メタ文字とは、通常の文字以外の特殊な機能を持つ表記のことでした。この連載の第1回にもいくつか紹介しましたが、" "などで変数展開されるメタ文字をひととおり挙げてみます（**表2**）。

表2のメタ文字は、 \ とほかの文字の組み合わせですが、 \ を単独で使うと、「エスケープ」といって、直後の文字が何らかの機能を持つ文字であっても、その機能を無効に

従来の表記	演算子表記	機能	変数展開
' '	q{}	文字列	×
" "	qq{}	文字列	○
` `	qx{}	コマンド実行	○
/ /	m{}	パターンマッチ	○
s/ / /	s{}{}	置換	○
tr/ / /	tr{}{}	変換	×

表1　クオート演算子

して、通常の文字として扱います。英数字は問題ありませんが、変数を表す$、ダブルクオート内での"や、シングルクオート内の'、パターンマッチ内での/、[、]、－など、記号類はいろいろな場面でそれぞれの機能を持っていたりするため、単にその記号を文字として扱いたくても、思わぬ機能が働いてしまうかもしれません。たとえば、

```
$a = "ダブルクオートは"です。";
```

と書いてしまうと、「ダブルクオートは"です。」という文字列を$aに代入したかったのに、代入されるのは「ダブルクオートは」だけになってしまいます。文中の"が、文字列の終わりと認識されてしまうのです（実際はその前にエラーになりますが）。

この場合正しくは、

```
$a = "ダブルクオートは\"です。";
```

と、文字として扱いたい"の前に \ を挿入します。これを「\ でエスケープする」といって、"の本来の機能（文字列を囲む働き）を無効にします。

パターンマッチのときにも、

```
/http:¥/¥/www/;
```

という例がありましたが、この場合の \ もエスケープです。

このとき、\ によるエスケープを使いたくないのであれば、それぞれ、

メタ文字	意味
\a	ベルをならす
\b	バックスペース
\cM	コントロール文字
\e	エスケープ（ESC）
\f	改ページ
\n	改行（LF）
\r	復帰（CR）
\t	タブ
\033	8進数で指定した文字
\x5a	16進数で指定した文字
\l	次の文字を小文字にする
\u	次の文字を大文字にする
\L	以降\Eまでを小文字にする
\U	以降\Eまでを大文字にする
\Q	\Eまでの英数字以外の文字の前に\を挿入する
\E	\L、\U、\Qの効果の終わり

表2　メタ文字

```
$a = qq{ダブルクオートは"です。};
m{http://www};
```

と、クオート演算子を使って書くことにより、前者では"が、後者では/が特殊な文字ではなくなります。

ただ、文字列の場合で変数が含まれないのなら、

```
$a = 'ダブルクオートは"です。';
```

と、シングルクオートで囲む方法もあります。

文字列をつなぐ

文字列と文字列をつなぎたいことがよくあります。たとえば、「定価 5000 円」と出力したいとき、普通は数字部分が変数に記憶されていて、出力するときに、

```
print "定価 $kakaku 円";
```

とすれば、変数展開されて「定価 5000 円」というように出力されます。

直接出力するときだけでなく、「定価」と「円」を付け加えた結果を、再び変数に記憶させる場合でも、

```
$a = "定価 $kakaku 円";
```

と書くことができます。

ただし、変数展開させるために" "を使っていますから、これを、

```
$a = "定価は¥$kakakuです。";
```

と書いてしまうとうまくいきません。本当は「定価は\5000です」という結果になってほしいのですが、 実際には、「定価は$kakakuです。」と出力されるでしょう。

これは、\ のエスケープ機能が働いたために$の変数を表すという機能が無効になり、「$」という文字として出力されてしまったのです。そのため、「kakaku」も変数名ではなく単なる文字としてそのまま出力されました。\ でのエスケープやメタ文字を無効にするには、" "を' 'に変えてしまうと前述しましたが、

```
$a = '定価は¥$kakakuです。';
```

にすると、今度は「定価は\$kakakuです。」と表示されます。\はうまく文字として出力されたのですが、肝心の$kakakuも変数展開されていません。これもダメですね。

これを解決する方法として、「.」があります。「.」は、文字列と文字列をつなぐ演算子です。「.」はいわば足し算のように、次々と文字列をつないでいってくれます。たとえば、

```
$a = "価格";
$b = "円";
$kakaku = "5000";
```

のとき、

```
$c = $a . $kakaku . $b;
```

とすると、$cは「価格5000円」となります。また、

```
$c = "価格" . $kakaku . "円";
```

というように、変数と文字列をつなぐことも可能です。

Perlでは、変数そのものに数値か文字列かという区別はありませんから、数値として計算するのか文字列として扱うかは演算子で決まります。たとえば、

```
$kakaku = 5000;
$zei = 250;
```

のとき、

```
$kakaku + $zei
```

は、「5250」というたし算の結果になりますし、

```
$kakaku . $zei
```

は、「5000250」というように文字列として両者をつないだものが結果になります。

では、さきほどの\を含んだ文字列はどのように扱えばよいでしょうか。「定価は\」と「です。」の部分は、そのまま出力すればよく、$kakakuは変数展開させたいのですから、それぞれを「.」でつないで、

```
$a = '定価は¥' . $kakaku . 'です。';
```

とすればよいことになります。「定価は\」と「です。」の部分は' 'で囲まれていますから、\やメタ文字の機能は無効になり、文字がそのまま出力されます。2番目の$kakakuは変数を直に書いていますので、変数展開されて内容が表示されます。

文字列を扱う場合は、たいてい" "で大丈夫なのですが、「\」「"」「$」「@」などの文字が含まれる場合は気をつけてください。それぞれが特別な機能を持ちますから、" "の中では変数展開されたり、メタ文字として扱われてしまいます。そういう場合は、うまく" "と' 'やqw{ }を使い分けながら、「.」でつないでいくという工夫が必要です。

文字の並び

それでは文字列に関する便利な関数と使い方を紹介しましょう。

■length

ある文字列が何文字なのかを知りたいときがあります。もちろん自分で直接書いた「あいうえお」などの文字列の文字数はすぐにわかりますが、いろいろな処理をした結果、変数に記憶されている文字列やユーザーに入力してもらった文字列などについては文字数を調べる必要があります。

文字列の文字数を調べるにはPerlに用意されているlength()という関数を使います。たとえば、$a = "ABCDEFG"のとき、

```
$n = length($a);
```

とすれば$nは7になります。つまり、$aに記憶されている文字列の文字数を調べてくれます。

■substr

文字列の中の一部分を取り出すときにはsubstr()という関数を使います。先ほどと同じ$aの3文字目が何かを知りたいとき、もしくは取り出したいときには、

```
$s = substr($a,2,1);
```

とすれば、$sには「C」という文字が入ります。しかし、ここで3文字目なのに「2,1」と指定しているのはなぜなの

でしょう。

文字列の構造を考えてみましょう（**図1**）。文字列は文字が順に並んでできています。substrはこの文字列の一部分を取り出すのですから、どの部分かということを指定してやらなければなりません。そのために何文字目からかという位置を数字で表すのですが、位置番号は**図1**のように文字と文字のあいだに、先頭が0として割り振られています。つまり1文字目は0から、2文字目は1から、3文字目は2から始まります。

上記の例では3文字目の1文字だけを取り出しましたが、substrが取り出せるのは文字列の「一部分」ですから、取り出す文字数も指定できます。それが3番目に指定している数字です。**図1**には2文字目から3文字取り出すときの位置とsubstrの式があります。substr関数の使い方をまとめると、

```
substr(変数名, 位置番号, 取り出す文字数)
```

となります。

■ substrによる文字の変更

substrは文字列の中の一部分を取り出すだけでなく、その部分を変更することができます。その場合は、substrが左辺に来ます。**図1**で取り出した部分を変更してみましょう。$a = "ABCDEFG"のとき、

```
substr($a,1,3) = "123";
```

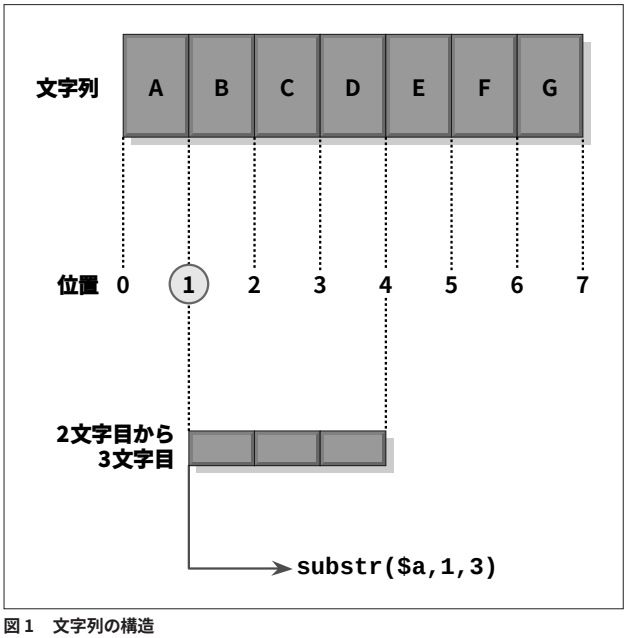


図1　文字列の構造

とすると、$aの内容は「A123EFG」になり、「BCD」と「123」が入れ替わったことになります。

また、元の部分の文字数と新しく代入する文字列の文字数を変えるとおもしろいことになります。今の例では元の文字列が3文字ですから、新しい3文字と入れ換えると、文字の置換ということになります。これを2文字にしてみましょう。

```
substr($a,1,3) = "12";
```

結果は、「A12EFG」となり全体の文字数が1文字減りました。

これをいろいろな文字数で試したものが**表3**です。結果を確認してください。0文字で入れかえたときは、元の部分がなくなっています。つまり文字列の一部分を削除することができるのです。4文字、5文字のときなどは元の部分の文字数よりも多い文字が挿入され、文字列全体も長くなります。

このように、文字の位置を指定して行う処理のような場合に重宝するsubstrですが、応用として、7桁の郵便番号に「ー」を挿入することを考えてみましょう。正規表現による置換でもできますが、郵便番号などのように桁数が決まっている場合はsubstrのほうが適しています。

$a = "1518024"の場合に、郵便番号として3桁と4桁に分けるための「ー」を挿入するスクリプトは、

```
substr($a,3,0) = "ー";
```

になるのです。どうです、わかりましたか。元の部分の文字数には0も使えるのですね。この場合は文字列を挿入する働きをするのです。

■ index／rindex

文字の位置に関する関数に、index／rindexがあります。これは、文字列中に指定した文字が含まれているかどうかを調べ、もし含まれるならその位置を返してくれる関

文字数	式	結果($aの内容)
0	substr($a,1,3) = ""	AEFG
1	substr($a,1,3) = "1"	A1EFG
2	substr($a,1,3) = "12"	A12EFG
3	substr($a,1,3) = "123"	A123EFG
4	substr($a,1,3) = "1234"	A1234EFG
5	substr($a,1,3) = "12345"	A12345EFG

表3　substrで文字数を変えた場合の処理結果

数です。つまり、$a = "ABCDEFG"のとき、

```
$n = index($a,"E");
```

とすると、$nには4が入ります。「E」は5文字目ですが、ここでの位置番号も**図1**と同じものですから、「E」の位置として4が返ってくるのです。1文字だけを探すのなら、「5文字目」と考えたほうがよさそうですが、index／rindexでは1文字だけでなく、部分としての文字列を探せます。たとえば、

```
$n = index($a,"CDE");
```

という使い方もできるのです。結果は2ですね。

もし、その文字列が含まれていないときには結果として－1が返ります。

rindexはindexと同じ機能ですが、探す方向が文字列の末尾のほうからになります。たとえば、$a = "ABCDE ABCDE"のとき、

```
$n = index($a,"C");
```

とindexを使えば結果は2になり、

```
$r = rindex($a,"C");
```

とrindexを使えば結果は7になります。つまり$aの末尾から先頭に向けて探すので、後ろの「C」のほうを探し当てることになります。

では、郵便番号処理の応用として、今度は「－が含まれていれば、その－を削除して7桁の数字だけにする」というスクリプトを、パターンマッチを使わずに考えてみましょう。

$zipに郵便番号が入力されているものとして、

```
$n = index($zip,"ー");
if ( $n >= 0 ) {
   substr($zip,$n,1) = "";
}
```

となりますね。単に削除するだけならパターンマッチのほうが簡単ですが、「－」の位置を覚えておいて、あとで使う場合などはindexを使うことになります。

CSVデータの処理

データを記録したり管理するときに、CSV形式というのがよく使われます。これは、ある項目順にデータを「,（カンマ）」で区切って並べ、改行で終わる1行が1件のデータとして扱われるテキストデータの形式です。データベースソフトのデータなどをテキストデータに変換するときによく用いられますし、データベースソフトでなくても、項目の決まったデータを保管しておくのに便利です。

PerlではCSVデータを簡単に扱える関数が用意されています。CSV（カンマ区切り）の状態の文字列をカンマごとに区切って配列に格納するのがsplit()で、その逆を行うのがjoin()です（**図2**）。

■ split()

$lineにCSVデータ1行が格納されているとき、カンマごとに区切って配列@dataに格納する場合は、

```
@data = split(/,/ ,$line);
```

と書きます。カンマが2つもあってちょっとわかりにくいのですが、()内の最初の/,/はパターンマッチで、これとマ

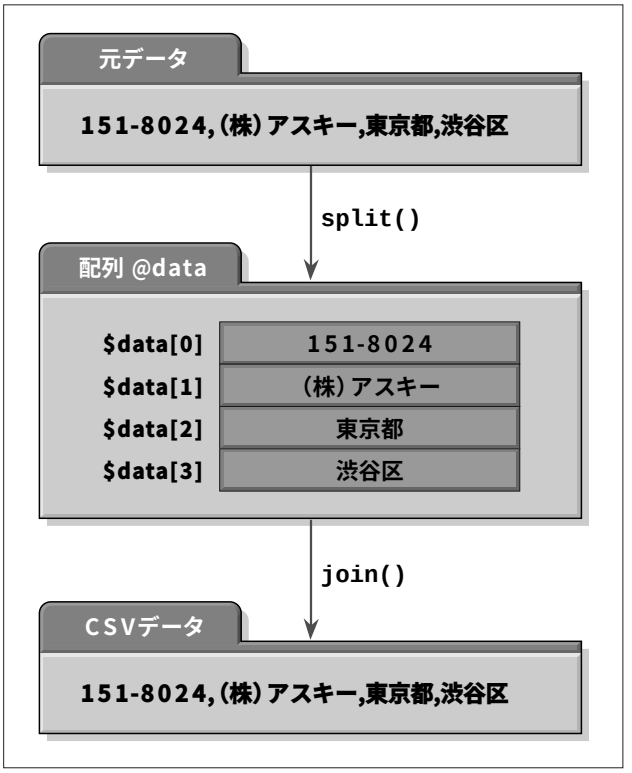


図2　split／join関数とCSVデータ

ッチする文字（列）が区切り文字になります。CSVデータの場合は区切り文字が「,」ですから、パターンは/,/となるのです。これがもしタブ区切りならばパターンは、

```
/¥t/
```

となります。つまり、区切り文字はどんなものにでも対応できるのです。ときにはタブや；などが使われることもありますが、ようは区切り文字を正規表現パターンで表せばよいということです。

パターンの後にはカンマに続いて元データの格納されている変数名を書きます。

split()は区切り文字で分割された個々のデータをリストとして返してきますので、それを配列変数に代入すればそれぞれのデータが**図2**のように配列の各要素になります。

■join()

join()はsplit()の逆を行うもので、

```
$list = join("," , @data);
```

のように使います。配列から1行の文字列が作成されるので、左辺と右辺の変数名が入れ替わっていますね。ただし、split()のときと違って、区切り文字はパターンではなく文字列として" "または' 'で囲んでください。

CSVファイルを1行ずつ読み込んでsplit()を使えば、データベースのように各項目を操作することができますし、配列のデータをjoin()で連結して1行ずつファイルに書き出せば、CSVデータとして保管することができます。

図3　CSVデータを加工

■CSVデータの加工

例として、CSVデータの並び順を変えてみましょう。リストは、**図3**の元データのようなCSVファイルを1行ずつ読み込んで、会社名と郵便番号のデータ順を逆にして、今度は/で区切って並べたデータを表示するスクリプトです。もちろんこのデータをファイルに書き出せばCSVファイルを作ることができます。

$data[0]と$data[1]の内容を入れ替える部分が慣れないとややこしく感じるかもしれませんが、じっくり考えてみればわかると思います。

ただし、split()の部分はパターンの後ろの変数名が省略されています。これは、例のごとくデフォルト変数$_ですね。つまり、split()のパターンの後ろの変数名を省略すると変数$_を使うということになっています。ここでは、その前のwhile(<IN>)によって、ファイルから1行ずつ変数$_に読み込まれていますので、split()でも変数$_を使うようにすればよいのです。なお、join()ではデフォルト変数$_の使用はできません。

このように、split()とjoin()を使ってCSVデータを処理すれば、ちょっとしたデータベース代わりになります。10フィールドくらいの項目数で、1000件程度のデータの処理なら、処理速度もそれほど遅くなく実用になります。PostgreSQLやMySQL、Oracleなどのデータベースをわざわざ使わなくても、このようなCSVベースのデータ処理が可能なのです。

文字列の書式付き出力

文字列や数値をさまざまに加工して出力したい場合などは、書式付き出力関数を使うと便利です。ここでは、printfとsprintfを解説します。

▼リスト1　CSVデータの加工

```
open(IN,"test.csv");
while(<IN>) {
        @data = split(/,/);
        $a = $data[0];
        $data[0] = $data[1];
        $data[1] = $a;

        print join("/",@data);
}
close(IN);
```

■printf

数字などを出力するときに、桁を合わせて右揃えにしたい場合があります。そんなときはprintの代わりにprintfを使うと便利です。

```
printf 書式指定文字列, 変数名, 変数名,……
```

printfは、書式を指定して文字列を出力する関数で、書式指定文字列というものが必要になります。書式指定文字列は%から始まる文字列で、**図4**のような構成になっています。

%の次には、−、＋、0または空白を指定することができ、それぞれ**表4**のような意味を持ちます。そのあとには桁数を数字で指定します。数字を指定すると、強制的にその桁数で表示されます。この数字を省略すると桁数の強制は行われません。

桁数を指定しても、出力されるデータがその桁数に満たない場合、**表4**にあるとおり、%の後に0を付けると数字の前の部分に0が埋められます。

数字のあとの英文字は、どのような型のデータとして出力するかを指定します。**表5**はよく使われる型の指定方法です。実際には、

```
printf "%4d",$n;
printf "個数%3d  金額%4d",$kosu,$kin;
```

というように、書式指定文字列の中の%がそれぞれ後ろの変数と対応していますから、%と変数名との数は同じでなければなりません。

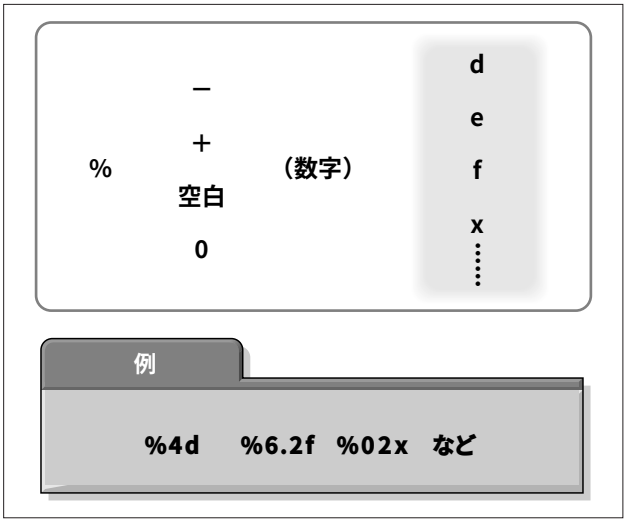


図4　書式指定文字列

これだけではなかなかわかりにくいと思われるので、実例を見ていきましょう。

$a = "12"、$b = "340"、$c = "5600" のとき、

```
printf "%4d\n",$a;
printf "%4d\n",$b;
printf "%4d\n",$c;
```

というスクリプトを実行すると、表示は、

```
  12
 340
5600
```

のように、すべて4桁として右に揃えて表示されます。このとき、「%4d」を「%04d」とすると、

```
0012
0340
5600
```

と、数字の前が0で埋められます。

小数の場合は、整数部分と小数部分の桁数を指定することができ、整数部分4桁、小数部分2桁の場合は、

```
%6.2f
```

と指定します。つまり、「(全桁数).(小数桁数)」という指定になりますので注意してください。

表記	意味
−	左詰めにする（通常は右詰め）
＋	正の数値に＋符号を表示する
（空白）	正の数値のとき空白を付ける
0	桁数が満たないとき0で埋める

表4　%直後の記号

表記	意味
d	10進数
e	指数形式の浮動小数点
f	浮動小数点
x	16進数
X	16進数（大文字のA〜F）
o	8進数
c	文字
s	文字列

表5　データ型の指定

■ sprintf

printfはprintの代わり、すなわち画面に表示（出力）するときに使いますが、前の桁を0で埋める機能などはスクリプト内の処理としても使いたい場合があります。たとえば8桁のID番号を作成する場合などは、数値としては8桁に満たなくても、8桁分の数字が並んでいなければなりません。

printfと同じ処理を施した結果を、文字列として変数に代入するときは、sprintfを使います。sprintfも書式はprintfと同じですが、左辺に変数名を書きます。

```
$id = sprintf "%08d", $a;
```

決まった書式にデータを加工する場合、正規表現だけでなく、特に桁数や文字数が決まっている場合はprintfやsprintfが役に立ちます。

ヒアドキュメント

行数のまとまった固定のテキストデータを出力する場合、これまでのようにprintを使うと、

```
Perlスクリプト入門
第1回　テキストデータ加工
第2回　ループとファイル入力
第3回　正規表現
第4回　配列とリスト
```

という文章を出力するのに、

```
print "Perlスクリプト入門¥n";
print "第1回　テキストデータ加工¥n";
print "第2回　ループとファイル入力¥n";
print "第3回　正規表現¥n";
print "第4回　配列とリスト¥n";
```

と書かなければなりませんでした。毎行、printや\nを書かなければならないのも面倒ですし、スクリプト自体も決して見やすいものではありません。

単に文字列をそのまま出力するだけなのですから、もう少し楽に処理できればと思うでしょう。そんなときに便利なのがヒアドキュメントです。

ヒアドキュメントとは、スクリプト中に出力したい状態そのままの文字列を埋め込んでしまう機能です。今の例をヒアドキュメントで書いてみると、

```
print <<"EOD";
Perlスクリプト入門
第1回　テキストデータ加工
第2回　ループとファイル入力
第3回　正規表現
第4回　配列とリスト
EOD
```

となります。1行目は、ちょっと変わった書き方ですが、このとおり覚えておいてください。つまり、「EODというマークのところまでprint」という意味になります。ただし、<<と"EOD"の間には空白を入れないようにしてください。文字列をマークだと思ってしまいます。

最終行が「EOD」だけの行になっています。この行にも余計な空白や；を書いてはいけません。必ずこのマークである「EOD」だけを書いておきます。

「EOD」はEnd Of Documentの頭文字を取ったものですが、ここには自由な名前をつけることができます。文字データの終わりまでということで、慣習的に「EOD」や「EOS」がよく使われるというだけです。別にこれにこだわる必要もありませんし、スクリプト内でヒアドキュメントが2カ所以上使われる場合などは、それぞれ違った名前にしなければなりません。

■ヒアドキュメントの扱い

先ほどの例の1行目の"EOD"はダブルクオートで囲まれていますが、この場合は実際のデータ部分（2行目から5行目）がダブルクオートで囲まれているものとみなされます。つまり、ここに変数やメタ文字を使えば展開されます。

展開されて困る場合は、文字列と同様にシングルクオートを使います。

```
print <<'EOD';
```

これで、データ部分は変数展開やエスケープされずに、そのまま出力されることになります。

今回は文字列とその処理を徹底的に学びました。Perlの処理の中心は何といっても文字列ですから、前回の正規表現と今回の文字列処理をマスターすれば、あなたもけっこうな「Perl使い」といえるのではないでしょうか。

[超] 入門シェルスクリプト

bashのシェルスクリプトについて学ぶ本連載。今回は、スクリプトの中のスクリプトとも言えるシェル関数の定義方法や呼び出し方、引数やローカル変数の利用方法などについて説明し、あらかじめ登録したディレクトリの一部を指定するだけでカレントディレクトリを変更できるシェル関数を作成する。

第9回　シェル関数を使う

文：大池浩一
Text:Koichi Oike

この連載でこれまでに作成したシェルスクリプトは、分岐や繰り返し処理が含まれているにしても、基本的には先頭の行から末尾の行へ順番に実行されるというものだった。数行から十数行の短いスクリプトを作っているだけなら、これでも問題はほとんど生じない。

しかし、何十、何百行もあるような長いスクリプトを書いていると、いくつかの部分で似たような処理が必要になることがある。こうした場合、C言語やPerlなどのプログラミング言語では、同じような処理を何度も書く代わりに、「関数」や「手続き」「サブルーチン」などと呼ばれる部分的なコードを定義して、必要な場面でそれを呼び出すという手法が使われる。

Linuxの標準シェルであるbashや、そのモデルとなったBourneシェル（ただしSystem V以降の新しいものに限る）にも、これとよく似た「シェル関数」という仕組みがが用意されており、スクリプト内の似たような処理をまとめてモジュール化できる。作成した関数は、通常のコマンドと同じように引数を付けて実行可能だ。

また、スクリプト自体の代替物として、コマンドラインでシェル関数を実行することもできる。実行時にファイルを検索して内容を読み込む必要のあるスクリプトと違って、シェル関数は最初からメモリに保存されているので素早く実行できる。また、コマンドラインを制御しているシェルの環境変数を変更するなど、スクリプトでは難しい処理を行うために利用することも可能だ。

シェル関数の定義と利用法

シェルはスクリプトの内容を1行ずつ読みながら逐次的に処理を進めるインタプリタなので、シェル関数を利用する際には注意しなければならない点がある。それは、「シェル関数を呼び出す（実行する）より前の時点で、関数の定義をしなければならない」ということだ。

また、シェル関数を呼び出す際の引数の指定方法は、C言語やPerlなどとはかなり異なるし、シェル関数で得られた結果を「関数値」として呼び出し元に返す仕組みも用意されていない。C言語やPerlなどのプログラミング経験のある人は、こうした点にも注意が必要だ。

なお、bashの場合は、シェル関数内でのみ有効なローカル変数を利用できる。大規模なシェルスクリプトを作成するには、ほかの関数や関数外部のスクリプトに影響を与えないローカル変数は必須といえる。

以下では、

- **シェル関数の定義と呼び出し方**
- **引数（位置パラメータ）とローカル変数**
- **スクリプトの代替物としてシェル関数を使う方法**

などの説明を行った後、カレントディレクトリを変更するシェル関数の作成を行う。

●シェル関数の定義と呼び出し方

シェル関数を定義する書式には、

```
function 関数名
{
  関数で実行したいコマンド
}
```

というものと、

```
関数名 ()
{
  関数で実行したいコマンド
}
```

という2通りがある。機能的な違いはないので好きなほうを使おう。ここでは前者の書式を利用する。

まず、シェル関数であることを示す「function」というキーワードに続いて、関数名を指定する。関数名には、アルファベットと数字、一部の記号（「_」など）を利用できる。ただし、数字は関数名の先頭には使えない。

続いて、中カッコ「{」「}」に囲まれた部分に、関数を呼び出した際に実行してほしいコマンドを記述する。複数のコマンドをパイプで接続したり、1行ずつ（あるいは「;」で区切って）並べることも可能だ。もちろん、ifやcase、for、whileといった制御構造を利用できる。

たとえば、端末画面に「ほげほげ」と表示する関数hogeは、以下のように定義する。

```
function hoge
{
  echo ほげほげ
}
```

シェル関数をスクリプトで利用するには、スクリプト中で関数を定義し、通常のコマンドと同じように関数名を行頭に書けばいい。たとえば、

```
#!/bin/sh
function hoge
{
  echo ほげほげ
}
hoge
hoge
```

というスクリプトを実行すると、シェル関数hogeが2回実行され、2行の「ほげほげ」が表示される。

なお、定義したシェル関数と同名のコマンド（実行ファイル、シェルの組み込みコマンド、スクリプト）が存在する場合、シェル関数はコマンドよりも優先的に実行される。このため、シェル関数を利用して既存のコマンドを置換する（あるいは改良する）ことができる。具体的な例はのちほど説明しよう。

注意すべき点として、シェル関数を実行する前に関数の定義を済ませておくことが挙げられる。先ほどのスクリプトが次のように書かれていたらどうなるだろうか。

```
#!/bin/sh
hoge
function hoge
{
  echo ほげほげ
}
hoge
```

最初にhogeを実行する段階では、シェル関数hogeはまだ定義されていないので、シェルには関数の内容がまったくわからない。そのため、「hoge: command not found」とエラーメッセージが表示され、関数hogeは実行されない。もし、同名のコマンドhogeが存在すると、エラーメッセージは表示されずにそちらが実行されてしまう。一方、2つめのhogeを実行する段階では、シェル関数hogeが定義されているため、たとえ同名のコマンドがあったとしても、関数のほうのhogeが実行される。

要点をまとめると、

- **シェル関数の定義は「function 関数名 { コマンド }」あるいは「関数名 ( ) { コマンド }」で行う。**
- **スクリプトで定義したシェル関数は、通常のコマンドのように実行できる。ただし、実行する前に関数の定義を済ませておく必要がある。**
- **コマンドと同名のシェル関数を定義した場合、コマンドよりも関数が優先的に実行される。**

ということになる。

●引数（位置パラメータ）とローカル変数

本連載でこれまで説明してきたように、シェルスクリプト実行時にコマンドラインで指定した引数の内容は、「位置パラメータ」と呼ばれる特殊な組み込み変数に格納される。たとえば、最初の引数は位置パラメータ「1」に格納され、スクリプト中では「$1」で参照可能だ。

スクリプト中のスクリプトと言えるシェル関数にも、まったく同じ仕組みが用意されている。たとえば、

```
say hoge ほげほげ
```

とコマンド入力すると、

```
hoge:「ほげほげ」
```

と表示する関数sayは、次のように定義すればいい。

```
function say
{
  echo "$1:「$2」"
}
```

シェル関数を実行する際に、そのコマンドラインで指定された引数は、位置パラメータに格納され、$1、$2、…で参照できる。先の例でいえば、実行時の最初の引数「hoge」は$1、次の引数「ほげほげ」は$2で参照可能だ。ここでは使っていないが、位置パラメータの数を「$#」で参照したり、それぞれの位置パラメータを「"」で囲んでスペース区切りで並べる「"$@"」なども利用できる。

それでは、スクリプト実行時の引数が格納された位置パラメータのほうはどうなってしまったのだろうか。実はちゃんと保存されている。ただし、シェル関数内では直接参照することはできないのだ（**図**）。もっとも、「$0」のみは例外で、関数の内外を問わず、常にスクリプトのファイル名を参照できる。

もし、シェル関数内でスクリプトの引数を利用したいなら、あらかじめ適当なシェル変数に引数の内容を格納しておき、呼び出した関数内ではシェル変数の内容を参照すればいい。シェル変数の内容は、関数の内外で変化しないからだ（次に説明するローカル変数を除く）。

また、シェル関数での処理結果（文字列や数値など）を、呼び出した側に「関数値」として返す機能は用意されていない。こうした場合も、シェル変数を利用して処理結果を格納する必要がある。

通常のシェル変数に格納されている内容は、位置パラメータとは違って、シェル関数の内外で変化しないグローバ

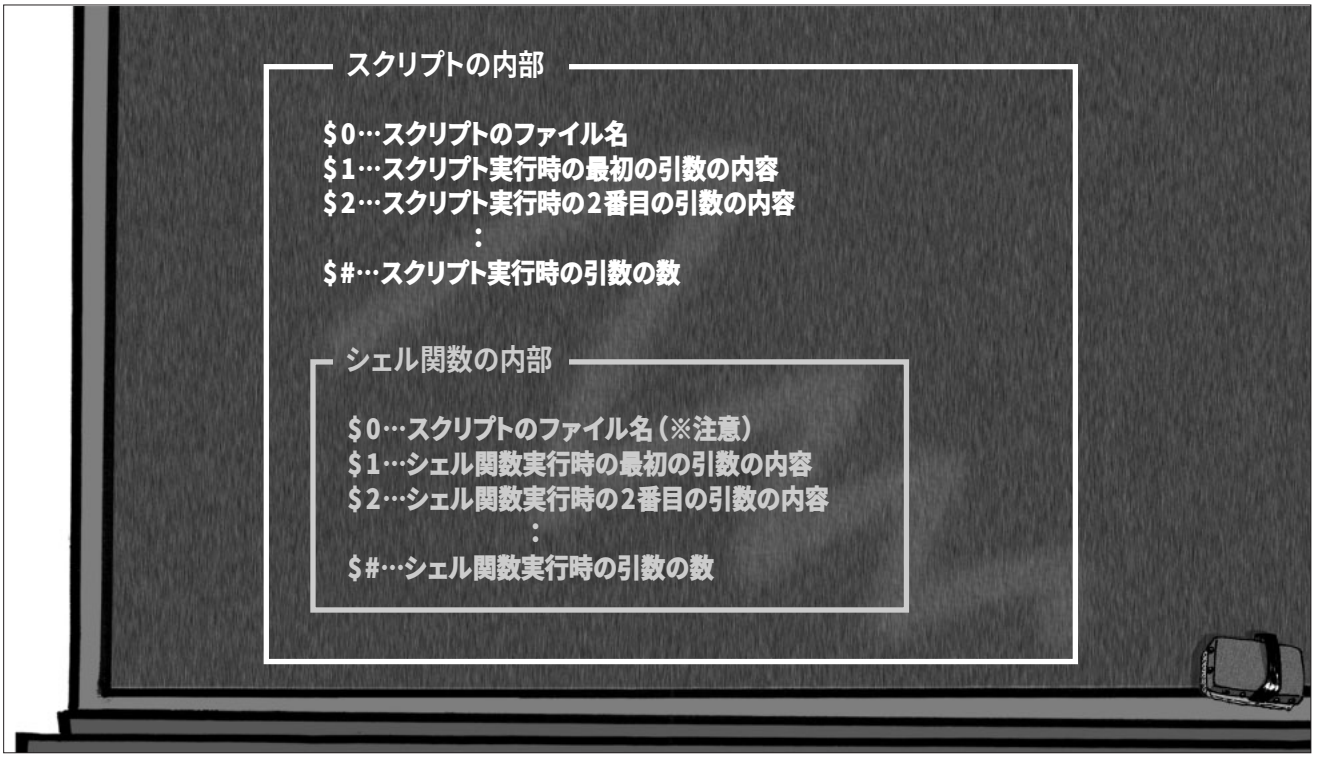


図　シェル関数は独自の位置パラメータを持つ

ル変数だ。たとえ、ある関数内でしかシェル変数の内容を設定・参照しないとしても、一度関数を実行した後は、スクリプトのどの部分でもそのシェル変数の内容を参照したり、書き換えたりできる。

このようなグローバル変数だけでは、大規模なプログラムの作成は難しい。常にあらゆる関数で使われているシェル変数の名前を把握していないと、複数の関数でシェル変数名の衝突が起こる。その結果、ほかの関数で設定されたシェル変数の内容を別の関数内で上書きしたりして、思わぬバグの原因になるのだ。

そこで、たいていのプログラミング言語には、関数内でしか利用できないローカルな変数を定義する機能が用意されている。Bourneシェルでは、それほど大規模なスクリプトを作成することを想定していなかったので、ローカル変数を定義することはできない。しかし、Linuxで使われているbashでは、組み込みコマンドのlocalを利用して、シェル関数内でしか利用できないローカルなシェル変数を定義する機能が用意されている。

localの書式は以下のようになる。

```
local 変数名
```

変数名には、関数内でのみ有効なシェル変数の名前を指定する。複数の変数をスペースで区切って並べることも可能だ。たとえば、

```
function hoge
{
  local foo bar
  コマンド...
}
```

という関数hogeでは、シェル変数fooとbarがローカル変数として定義される。fooとbarは、この関数内では通常のシェル変数と同様に、「foo="ほげほげ"」などとして内容を設定したり、「$bar」として内容を参照したりできる。ただし、ローカル変数の内容は関数を終了した際に失われ、関数の外部（呼び出した側）では利用できない。

それでは、シェル関数の外部に同名の（グローバルな）シェル変数が使われていたらどうなるだろうか。この場合、関数の外部ではグローバル変数、内部ではローカル変数が有効になるため、関数内でのグローバル変数の内容の変更や参照はできなくなる。

要点をまとめると、

- **関数実行時の引数の内容は、位置パラメータに格納され、$1、$2、…で参照できる。**
- **スクリプト実行時の引数の内容は、シェル関数内では直接参照できないため、あらかじめシェル変数に格納しておく必要がある。**
- **シェル関数の処理結果を呼び出し側に返す場合もシェル変数を利用する。**
- **組み込みコマンドのlocalを使うと、シェル関数内でのみ利用可能なローカル変数を定義できる。**
- **ローカル変数と同名の通常のシェル変数の内容は、シェル関数内では変更・参照できない。**

ということになる。

●シェル関数をスクリプトの代わりに利用する

これまでの例では、シェルスクリプトの中でシェル関数の定義と実行を行っていた。こうした方法で定義されたシェル関数の内容は、スクリプトが終了すると自動的に失われてしまう。というのも、シェル関数はそれを定義したシェルでのみ有効だからだ。

シェルのコマンドラインからスクリプトを実行すると、最初に「サブシェル」と呼ばれるシェルのコピーが起動される。サブシェルは指定されたスクリプトを読み込み、順次コマンドを取り出しながら実行する。すべてのコマンドが終了すると、サブシェルは終了して元のシェルに制御を戻す。この時点で、スクリプトで定義したシェル関数の内容は失われてしまう。

ところで、サブシェルには、元のシェルの環境変数の内容がそのままコピーされる。環境変数には、コマンド検索パス（ディレクトリ）が格納されたPATHや、カレントディレクトリが格納されたPWDなどが含まれる。しかし、サブシェルの環境変数の内容を変更しても、その内容は元のシェルには反映されない。

次のスクリプトspecsがよい例になる。

```
#!/bin/sh
cd /usr/src/redhat/SPECS
```

説明するまでもないだろうが、これはカレントディレクトリを「/usr/src/redhat/SPECS」に変更するスクリプトだ。実行すると、

```
$ specs
```

と正常に終了する。しかし、カレントディレクトリをpwdで確認するとspecs実行前と変わっておらず、/usr/src/redhat/SPECSにはなっていない。というのも、specsが変更したのはサブシェルのカレントディレクトリで、コマンドラインを制御しているシェルのカレントディレクトリは、まったく影響を受けないのだ。

それでは、コマンドラインを制御しているシェルの環境変数を変更するにはどうすればよいのだろうか。この問題を解決する方法のひとつは、コマンドラインを制御しているシェルで関数を定義することだ。以下のように、シェル関数の内容をコマンドラインに入力してみよう。

```
$ function specs
> {
>     cd /usr/src/redhat/SPECS
> }
```

「> 」はbashのプロンプトのひとつで、入力が完了していないことを示す。関数の定義が終了する「}」を入力した時点で通常のプロンプトに戻る。

これで、コマンドラインを制御しているシェルに対して、シェル関数specsが定義された。こうしたシェル関数は、通常のコマンドと同様に関数名をコマンドラインで指定すれば実行できる。試しに、

```
$ specs
$ pwd
/usr/src/redhat/SPECS
```

として、カレントディレクトリが/usr/src/redhat/SPECSに変更されていることを確認しよう。

コマンドラインで定義されたシェル関数の内容はメモリ上に保存されているため、いちいちファイルをPATHから検索して内容を読み込む必要のあるスクリプトよりも素早く実行できる。もっとも、最近の高速なCPUとハードディスクでは、あまり効果を実感できないかもしれない。

もうひとつの利点は、シェル関数が同名のコマンドより優先的に実行されることを利用して、既存のコマンドの動作を置換する（あるいは改良する）「ラッパー」関数を作成できることだ。

たとえば、次のシェル関数cdは、カレントディレクトリを変更する際に、変更後のディレクトリを画面に表示するためのラッパー関数だ。定義後は、組み込みコマンドcdよりもこの関数が優先的に実行される。

```
function cd
{
  builtin cd "$1"
  pwd
}
```

関数内では、組み込みコマンドcdとpwdを順番に実行しているだけだ。ただし、そのまま「cd "$1"」とすると、定義中のシェル関数cdが再帰的に実行されてしまう（最終的にはシェルが異常終了する）ため、組み込みコマンドbuiltinを利用して、組み込みコマンドのcdを明示的に実行している。通常のコマンドのラッパー関数では、組み込みコマンドcommandを使えばいい。

```
$ function ps
> {
>     command ps l
> }
```

ところで、現在のシェルで定義されているシェル関数の一覧は、組み込みコマンドdeclareを-fオプション付きで実行すると確認できる。

```
$ declare -f
```

このとき、関数名だけでなく、定義内容もあわせて表示される。-fの後ろに関数名を指定して、特定の関数だけ表示させることも可能だ。

一方、現在のシェルで定義されているシェル関数を削除したい場合は、組み込みコマンドunsetの引数に-fオプションと関数名を指定する。たとえば、さきほど定義した関数cdを削除するには、次のようにすればいい。

```
$ unset -f cd
```

以上の要点をまとめると、

- **シェル関数は、それを定義したシェル内でのみ有効。**
- **スクリプトで環境変数を変更しても、元のシェルの環境**

変数には反映されない。代わりに、コマンドラインで直接定義したシェル関数を利用する。

- **コマンドラインで直接シェル関数を定義すると、通常のコマンドのように実行できる。実行速度はスクリプトよりも高速。**
- **コマンドと同名のシェル関数を定義することで、既存のコマンドの動作を置換する「ラッパー」を作成できる。**
- **シェル関数の一覧表示は「declare -f」、関数の削除は「unset -f 関数名」で行う。**

ということになる。

今月のスクリプト

後半は、与えられたテーマを実現するスクリプトの作成手順を説明する。今月は、スクリプトではなく次のようなシェル関数を作成する。

- **指定したディレクトリをディレクトリリストに追加する「addir」**
- **ディレクトリの一部を指定するだけでカレントディレクトリを変更する「ccd」**

前半で説明したように、コマンドラインを制御しているシェルのカレントディレクトリを変更するには、シェルスクリプトではなくシェル関数を利用する必要がある。そこで、あらかじめカレントディレクトリの候補となるリストをファイルに保存するシェル関数addirと、そのリストを検索して見つかったディレクトリをカレントディレクトリとするシェル関数ccdを作成する。

●ディレクトリリストを作成する

カレントディレクトリの候補となるディレクトリのリストは、各ユーザーのホームディレクトリ（「~」と表記する）の「.ccd」というファイルに保存することにする。こうした「.」で始まる名前は、lsでは-aオプションを付けない限り表示されないので、設定ファイルやデータファイルによく使われている。

まずは、いくつかのコマンドを組み合わせてリストを作成してみよう。たとえば、ディレクトリ「/usr/src/redhat/SPECS」を、「~/.ccd」に追加するには、

```
$ echo /usr/src/redhat/SPS >> ~/.ccd
```

とすればいい。なお、追加する場合はリダイレクトに「>>」を使うことに注意されたい。「>」を使うと、現在の~/.ccdの内容がすべて失われてしまう。

こうして~/.ccdにディレクトリを追加していると、同じディレクトリが複数登録されてしまう可能性がある。そこで、内容が同じ行を1つにまとめることを考えよう。それには、行単位でソートするコマンドsortを、-uオプション付きで実行すればいい。

たとえば、~/.ccdに含まれる重複するディレクトリを1つにまとめるには、

```
$ sort -u -o ~/.ccd ~/.ccd
```

とする。-oは、出力ファイルを指定するオプションで、ここでは入力ファイルと同じ「~/.ccd」を指定している。「>」でリダイレクトする場合と違って、-oオプションでは一時ファイルを利用して入力ファイルの内容が失われるのを防いでくれるのだ（コラム参照）。

上記の2つのコマンドラインは、パイプ（|）を使って接続できる。echoの出力をパイプでsortに入力し、既存の~/.ccdの内容と合わせてソートすればいい。

```
$ echo /usr/src/redhat/SPECS | sort -u -o ~/.ccd -
~/.ccd
```

Column

リダイレクト処理のタイミングに注意

UNIX系OSの初心者が犯しがちなミスのひとつに、「入力ファイルと同じファイルに出力をリダイレクトしてしまう」というものがある。

たとえば、本文のsortを実行する場合に、

```
$ sort -u ~/.ccd > ~/.ccd
```

としてしまうわけだ。

入出力のリダイレクト処理は、コマンドの実行に先だって行われる。「>」の出力先に指定されたファイル（上の例では「~/.ccd」）は、このリダイレクト処理の時点で初期化され、内容が空になってしまうのだ。

このため、いざsortが実行されて「~/.ccd」の内容を読み込もうとした時には、「~/.ccd」の元の内容はすでに失われている。本文では、出力をリダイレクトする代わりに、sort自身にファイルへの出力処理を行わせることで、ファイルの内容が失われるのを防いでいるわけだ。

ここで、sortの入力ファイル名として、「~/.ccd」だけでなく「-」が指定されていることに注意されたい。これは「標準入力」を意味する特別なファイル名で、パイプ経由で入力されるコマンド（ここではecho）の出力を、他のファイルと同様に扱うことを意味する。この「-」を指定しなかった場合は、ソートの際にパイプ経由で入力された内容がいっさい無視されてしまうのだ。

●相対パスを絶対パスに変換する

シェル関数addirの実行時に、「/」で始まるフルパスを引数に指定するのは面倒なので、カレントディレクトリを基準とした相対パスで指定できることが望ましい。引数が指定されなかった場合は、カレントディレクトリ「.」を対象とすればいい。

一方、「カレントディレクトリの変更」という目的からすると、~/.ccdの内容は相対パスよりもフルパスのほうが都合がいい。処理を行う際に、変更前のカレントディレクトリを気にする必要がないからだ。

そこで、相対パスで指定されたディレクトリをフルパスに変換することにしよう。ここでは、もっとも簡単な「実際にカレントディレクトリを変更して、pwdでフルパスを取得する」という方法を利用する。

具体的には、コマンド置換を使って、

```
dir=`cd "$1"; pwd`
```

とすると、シェル変数dirに、最初の引数（$1で参照）のフルパスが格納される。

まず、「`」で囲まれたコマンド置換内の「cd "$1"」と「pwd」が順番に実行され、カレントディレクトリの変更とフルパスの出力が行われる。続いて、コマンド置換によりフルパスがシェル変数dirに格納される。なお、コマンド置換もスクリプトと同様にサブシェルによって実行されるため、このようにカレントディレクトリを変更しても、元のシェルには一切影響しない。

最初の引数（$1で参照）が指定されなかった場合に、シェル変数dirにカレントディレクトリ「.」を格納するには、シェルの置換演算子「:-」を使って、

```
dir=`cd "${1:-.}"; pwd`
```

とすればいい。置換演算子「:-」は、対象となるシェル変数（ここでは位置パラメータ「1」）の内容が空文字列の場合に、直後の文字列（ここでは「.」）で展開結果を置換する。つまり、引数を指定しなかった場合は、「cd .」によりカレントディレクトリは今のままで、pwdによってフルパスが得られる。

最後に、変更後のディレクトリが存在しなかった場合のエラー処理を追加すればフルパスへの変換部分は完成だ。ディレクトリが存在しない場合はcdが異常終了するので、if制御構文を使った分岐処理で対処できる。

```
dir=`if cd "${1:-.}" 2> /dev/null; then
       pwd
     fi`
```

cdが正常に終了した場合の処理は前と同じだ。一方、cdが異常終了した場合はpwdの実行はスキップされ、シェル変数dirは空文字列になる。その際、「No such file or directory」というエラーメッセージが表示されるため、標準エラー出力を「/dev/null」にリダイレクトしてメッセージを隠蔽している。

●ディレクトリリストを作成する関数addir

これでほぼ材料がそろったので、ディレクトリリストにディレクトリを追加する関数addirを作成できる（リスト1）。最初に一度だけ、

```
$ touch ~/.ccd
```

として、~/.ccdを作成すれば準備OKだ。

使い方は簡単で、変更先の候補としたいディレクトリを引数で指定すればいい。たとえば、

```
$ addir /usr/doc/fileutils-4.0
```

▼リスト1　関数addir

```
 1: function addir
 2: {
 3:   local dir
 4:   dir=`if cd "${1:-.}" 2>/dev/null; then
 5:           pwd
 6:         fi`
 7:   if [ -z "$dir" ]; then
 8:     echo "addir: $1 not found" > /dev/stderr
 9:     return 1;
10:   fi
11:   echo "$dir" | sort -u -o ~/.ccd - ~/.ccd
12: }
```

とすると、/usr/doc/fileutiles-4.0がディレクトリリスト（~/.ccd）に追加される。相対パスを利用することも可能だ。また、引数を省略して、

```
$ addir
```

とするとカレントディレクトリが追加される。

それでは、シェル関数の内容について解説しよう。まず、3行目では、シェル関数内で利用するシェル変数dirをローカル変数として定義している。続く4～6行目は、すでに説明したように、最初の引数で指定したディレクトリをフルパスに変換してシェル変数dirに格納する処理だ。

7～10行目では、ディレクトリが存在しない場合のエラー処理を行う。dirが空文字列の場合は、エラーメッセージを標準エラー出力（/dev/stderr）に表示し、終了ステータス1で終了する。終了ステータスを指定して関数を終了する場合、returnを使うことに注意されたい。

最後に、11行目のechoによりディレクトリのフルパスが出力され、現在のディレクトリリスト（~/.ccd）の内容とあわせてソートと重複行の処理が行われた後、新たなディレクトリリストとして保存される。

●カレントディレクトリを変更する関数ccd

addirによって作成されたディレクトリリスト（~/.ccd）を検索し、カレントディレクトリを変更するシェル関数ccdはリスト2のようになる。

使い方は簡単で、引数として変更先のディレクトリ名の一部だけを指定すればいい。

たとえば、

```
$ ccd fileutils
```

とすると、ディレクトリリストから「fileutils」を含むディレクトリを検索し、最初に見つかったディレクトリが新たなカレントディレクトリとなる。なお、「./」や「../」で始まる場合や絶対パスを指定した場合、あるいは引数を指定しなかった場合は、ディレクトリリストを検索せずにcdと同じ振る舞いをする。

それでは関数の内容を説明しよう。まず、4行目では、シェル変数dirに最初の引数（$1で参照）を格納する。なお、引数を指定しなかった（$1が空文字列の）場合は、置換演算子「:-」を使ってホームディレクトリ（$HOMEで参照）を格納するようにしている。

5～13行目では、case制御構造を使って、通常のcdと同じ振る舞いをさせる場合と、ディレクトリリストを検索する場合に分けている。6行目の「.|..|./*|../*|/*」は、通常のcdと同じ振る舞いをさせたいディレクトリのパターンで、dirがこれにマッチする場合はcase制御構造を抜けて14行目まで飛ぶ。

それ以外の場合は、8行目のパターン「*」（任意の文字列）が必ずマッチするので、9～12行目が実行される。まず、「`」で囲まれた「egrep "$dir" ~/.ccd | head -1」がサブシェルで実行され、dirの内容を含むディレクトリを~/.ccdから検索し、見つかったディレクトリの最初の1つをコマンド置換によりシェル変数foundに格納する。ディレクトリが見つかった場合は、10行目でdirの内容をfoundの内容で上書きする。見つからなかった場合はdirの内容は変化しない。

最後に、14行目で現在のdirの内容が「-->」とともに表示され、15行目でカレントディレクトリが変更される。

なお、今回作成したシェル関数addir/ccdを、コマンドラインを制御しているシェルに登録するには、適当なファイル（たとえばccdfunc）に両方の内容を記述し、組み込みコマンドsource（あるいは「.」）を使って、

```
$ source ccdfunc
```

とするといい。sourceや「.」は、指定したファイルの内容を、現在のシェルでそのまま実行する組み込みコマンドだ。

※今回紹介したリストは、Linux magazineのホームページ（http://www.ascii.co.jp/linuxmag/）からダウンロードできます。

▼リスト2　関数ccd

```
 1: function ccd
 2: {
 3:   local dir found
 4:   dir="${1:-$HOME}"
 5:   case "$dir" in
 6:   .|..|./*|../*|/* )
 7:     ;;
 8:   * )
 9:     if found=`egrep "$dir" ~/.ccd | head -1`; then
10:       dir=$found
11:     fi
12:     ;;
13:   esac
14:   echo "--> $dir"
15:   cd "$dir"
16: }
```

Linux日記

第22回　メール配送（9）

メールは、ヘッダとボディ（本文）から成り立っています。ヘッダは、MUA（メールクライアント）が付けるもの、sendmailなどのMTAが付けるものなどがあります。今回は、sendmailがヘッダをどのように扱うかを解説します。

illustration ; Aki

文：榊 正憲
Text : Masanori Sakaki

前回、シンセサイザの話が、ページ数の都合で尻切れトンボになってしまった。というわけで、筆者の手元にある2台のアナログシンセサイザを紹介しよう。

■KORG 800DV

1975年頃に購入した製品である（**写真1**、**2**）。これは、当時の国産機では唯一の2ボイスモデルだったと思う。アナログシンセサイザは基本的に単音しか出せないのだが、800DVは2ユニット内蔵し、同時に音程、音色の異なる2音を出力することができる。

アナログシンセサイザとしての構成はかなり固定的だ。ADSR（包絡線発生回路）も略式だし、外部入力もない。しかし、フィルタ構成がLPF、HPFが直列になっているとか、VCOにオクターブ違いの鋸波を重ねられるといった構成により、一般的な1ボイスシンセサイザよりは厚みのある音が出せる。自由な音作りを楽しむというよりは、ステージで扱う楽器としての側面が重視されていたのだろう。

■Roland SH-5

これは最近インターネットオークションで手に入れたものだ。800DVよりちょっと遅れて登場した製品で、思いっきりmini MOOGを意識した製品である（**写真3**）。1ボイスモデルであるが、完全に独立した2つのVCOを持ち、またLFOも2系統あり、パッチケーブル式ではない割には、各モジュールを自由に組み合わせていろいろな効果を得ることができる。当時の教科書通りの作り方という感じだ。また、外部からの信号入力や制御入力も持っているので、ステージ用シンセサイザでありながら、多重録音して遊ぶという用途にも十分に使える。

■比べてみると

SH-5は1ボイスではあるものの、800DVに比べて構成要素が多く、制御の自由度も高いぶん、いろいろな音が作れる。しかし、ステージで使う楽器という点では、2ボイスという面もあり、800DVのほうがおもしろい音かなという気がする（800DVをつい最近まで使っていたミュージシャンもいるらしい）。この手の楽器は、とにかくオシレータ音源の数が多いほうが音に厚みが出るのだ。

でも、どちらもそれなりの設計ポリシーと音の違いがあって楽しい。エレクトーンの外部入力端子につないで、いじって遊んでいる（2台接続するために、安いミキサまで買い込んでしまった）。いつかmini MOOGも欲しいなぁと思う今日この頃である（実は、トーンホイール／管球式のハモンドオルガンも欲しいと思っている）。

■いまさらISDNの話

久々に我が家のISDN機器を更新した。別に更新したかったわけではないのだが、情けない理由で更新せざるを得なかったのである。我が家はインターネット接続にOCNエコノミーを使っているので、基本的にデータ通信用に

写真1　KORG 800DVの外観

写真3　Roland SH-5の外観

ISDNは使っていない。ふだんは電話／FAX用に使っていて、たまにISDNルータを使ってプロバイダに接続し、実験などを行う程度だった。

ところが問題が発生した。ことの起こりは、ふと思い立って小さなWindows CEマシンを買ったことだ。メインの目的は、出先でメールをやり取りすることだ（かつて同じ理由でA4ノートを買ったのだが、ついに一度も目的を果たすことはなかった）。ふだんデスクトップ機でメールをやり取りし、たまにCE機を使って出先でメールを読もうとすると、メールサーバ上でIMAPサーバを稼働させる必要がある。我が家は、自宅でメールサーバを運用しているので、プロバイダにつなぐのではなく、自宅にダイヤルアップすることになる。もちろん、プロバイダ経由で自宅につないでもいいのだが、セキュリティのことなどを考えると、自宅に直接ダイヤルアップしたほうが簡単だ。

気合いの入ったユーザーであれば、PHSの通信端末をつなぐところだろうが、筆者のように自宅でくすぶっている時間が長い生活（外出は週に1〜2日である）の場合、そのためにPHSカード端末の料金を払うのは経済的ではない（そもそも、CE機を買ったこと自体経済的ではないのだが）。そこでモデムを使って、日頃使っている携帯電話でトロトロとつなぐことにした。とりあえず、Windows 2000 Serverにダイヤルアップ接続用のモデムをつなぎ込んだ（BSDマシンは近日更新予定なので、更新したらつなぎ換えるかもしれない）。ここで問題が起こる。自宅に電話をかけると、留守番電話／FAXが受けてしまうのである。モデムが先に受けるようにすると、電話が取れなくなる。この問題を解決するために、電話番号を追加するiナンバーというサービスの契約をした。

写真2　KORG 800DVの内部

昔からISDNを使っている人は知っていると思うが、iナンバーは比較的新しいサービスである。これを使うためには、iナンバー対応の機器を使わなければならない（iナンバーではなく、ダイヤルイン契約にすれば確実に動作するのだが、料金が高い）。ところが、我が家の電話／FAXがつながっているのは、骨董品的な、アナログポート専用TAなのである。このようなiナンバー非対応の古い機器が混ざっている場合に、iナンバーが正常に動作するかどうかをNTTに問い合わせたのだが、向こうでもわからないという（新しい番

号は、iナンバー対応の機器で使うのだが)。だめなら工事費無料で解約してくれるというので、とりあえず申し込んだのだが、案の定だめだった。iナンバーで追加した番号に電話をかけると、古いTA側も着信してしまうのである。

結局、iナンバーに対応した機器を新規に購入することにした。せっかく買うならということで、TAではなくルータにしたので、結構な出費になってしまった。素直にPHS通信カードを買って毎月料金を払うほうが安かったかもしれない。

新しく買ったヤマハのルータをアナログ回線専用にするのはもったいないので、永らく実験用として使っていたMN-128 SOHOを電話／FAX用に格下げした。これはパルスダイヤルに対応していないので、補助電話として使っていた電電公社の601電話機は、当分お役御免である。ヤマハのRT52iは一応ルータとしてネットワークに接続はしているが、本来のダイヤルアップ接続という用途には、今のところ使っていない。そのうち、何か実験でもするときに使うことになるだろう。こちらはパルスを駆動できるので、場所を整理したら601をつなごうと思っている。

さて、本題だ。前回までに、とりあえずルールセットの話をした。しかし、あれだけの記事では、ルールやルールセットの働きはわかっても、具体的にどのようにルールセットを書けばいいのかということはわからなかったと思う。実際にルールセットを自分で書いてみたいのであれば、コウモリの本を買おう。

sendmail.cfの中身で、まだ触れていない部分の解説を続けよう。今回はヘッダの話である。メールに含まれるヘッダの内容については、今までいろいろ解説してきたので、ここではsendmailが生成するヘッダの定義について説明する。

■ヘッダ

メールにはヘッダが含まれている。ヘッダには、必須のもの、オプションのものがある。また、MUAが作成するものもあるし、配信処理中にMTAが付加するものもある。MUAは、Subject：やTo：など、基本的なヘッダを生成する。これに加えて、メールの内容に関するヘッダも作成する。内容のデータ形式（プレーンテキスト形式やHTML形式など)、添付ファイルのエンコード形式やファイル名を記述するヘッダ類である。こういった内容に関するヘッダは、基本的に受信側MUAで使われる情報であり、配信するMTAにとっては関係のない情報だ(ごく一部、関与するものもある)。

sendmailは、メールを受信すると、ヘッダ部分をばらばらに分解し、メール本文とは別に保存する。そして再送信する際に（適当な配信エージェントに渡すときに)、再びヘッダと本文を組み合わせる。この処理の途中で、必要に応じてヘッダの書き換え、追加などが行われるのである。

ここでは、MTAに関連するヘッダの定義について説明しよう。

メールには、最初の作成時点から存在しなければならない、あるいは作成時点でのみ作成し得るヘッダがある。たとえば、Subject：ヘッダの内容は、最初にユーザーが指定することになる。To：やCc：なども最初から必要なヘッダである。しかしこれらは必須のヘッダではない。Bcc：で送信すれば、To：もCc：もないメールとなることを思い出そう。

Date：やMessage-Id：は必須のヘッダである。中継するMTAは、これらが存在すればそれを保存し、なければ新規に作成して付加する。また、中継を行うたびに、Received：ヘッダを追加する。From：は、メール作成時にMUAで作成することもできるし、MUAから実行される最初のMTAに作成させることもできる。

sendmailは、基本的に、すでに存在しているヘッダはそのまま保存する。必須のヘッダについては、すでに存在していればそれを保存し、なければ新規に作成して追加する（Date：や

Column

アナログ専用TA

今ではほとんど見ることはないが、かつては、電話やG3 FAX、モデムなどのアナログ機器を接続するための専用TA（デジタルデータ通信機能はまったく持っていない）が販売されていた。うちで使っていたのはこれである。さすがにアナログ「専用」TAだけあって、いろいろな機器を接続することができる。ベルを鳴らす昔のダイヤル電話も駆動できるし、モジュラジャックではないネジ止め式の機器にも対応している。また、設定のためのプッシュキーが付いていて、これがそこらの電話より高級なスイッチを使っているのだ。さすがに7万円くらいしただけのことはある（ちなみに、同じ頃買ったデータ通信用TAは15万円くらいした)。

写真　INSメイトD1-Sという古いアナログ専用TA

Message-Id：など）。それとは別に、単に追加するだけのヘッダもある（Received：ヘッダ）。

■ヘッダの定義

Date：やMessage-Id：のように存在しなければ生成するヘッダ、From：のように状況に応じて生成するヘッダ、Received：のように必ず付加するヘッダなど、sendmailはヘッダの生成と付加を行うことができる。このようなヘッダを定義するのが、sendmail.cf中のHコマンドである。

まずは、実際のHコマンドの例を見てみよう（**リスト1**）。

Hコマンドには、2つのパラメータを指定する。ひとつはフラグパラメータ、もうひとつは実際に付加するヘッダ行である。フラグパラメータは、Hに続く2つの？文字で囲まれた部分である。フラグパラメータはオプション項目で、指定しなくてもよい。フラグパラメータのあとは、実際のヘッダを記述する部分だ。sendmail.cfのほかの部分と同じく、空白やタブで始まる行は、前の行の継続行である。**リスト1**の例では、Received：ヘッダの定義が継続行を使っている。

■フラグ

フラグは、メールの配信時に、ヘッダを付加する条件を指定する。2つの？に囲まれた文字があるが、これがフラグ指定である。このフラグ指定条件が真になった場合に、H行で指定されるヘッダがメールに追加されるのである。ここで指定されるフラグは、配信エージェント定義のF＝パラメータで指定されたフラグに関連している。

前に説明したように、sendmailが配信を行う際は、ルールセット0で実際に使う配信エージェントを選択する。配信エージェントはMコマンドで定義されるが、ここでは、実際に使うプログラム名などとともに、フラグが指定される。これがMコマンドのF＝パラメータで、さまざまな文字によって配信エージェントの動作を指定する。Hコマンドのフラグパラメータで指定するフラグは、MコマンドのF＝パラメータを参照するものだ。

sendmailがHコマンドを評価する時点では、すでに配信エージェントが選択されている。そして、Hコマンドのフラグパラメータで指定されているフラグが、使用する配信エージェントのF＝パラメータでも指定されていれば、そのHコマンドが定義しているヘッダが付加されるのである。

たとえば、Date：ヘッダは、「H？D？Date： $a」という形で定義されている。？で囲まれたDフラグは、**リスト2**にあるように各配信エージェントのF＝パラメータに含まれているので、どの配信エージェントが選択された場合でも、このH行は評価されることになる。この例にはないが、もしDフラグを含まない配信エージェントが選択された場合は、Date：ヘッダは付加されないことになる。

？で囲んだフラグ指定がない場合は、選択された配信エージェントに関らず、そのヘッダが無条件に追加される。どのような形でヘッダの付加処理が行われるかは、少しあとで解説する。

■ヘッダ名

実際に付加するヘッダの内容は、フラグパラメータに続く文字列により指定される。ヘッダは、コロンで終わるヘッダの名前に続けて、各種文字列が続くという形式になる。Hコマンドの

▼リスト1　ヘッダ定義の例

```
H?P?Return-Path: <$g>
HReceived: $?sfrom $s $.$?_($?s$|from $.$_)
        $.by $j ($v/$Z)$?r with $r$. id $i$?u
        for $u; $|;
        $.$b
H?D?Date: $a
H?F?From: $q
HSubject:
H?M?Message-Id: <$t.$i@$j>
```

▼リスト2　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

Mprog,    P=/bin/sh,    F=lsDFMoqeuP, S=10, R=20/21,
          T=X-Unix, D=$z:/,
          A=sh -c $u

Msmtp,    P=[IPC],      F=mDFMuX,     S=31/11, R=41/21,
          T=DNS/RFC822/SMTP, E=\r\n, L=990,
          A=IPC $h

Mesmtp,   P=[IPC],      F=mDFMuXa,    S=31/11, R=41/21,
          T=DNS/RFC822/SMTP, E=\r\n, L=990,
          A=IPC $h
```

▼リスト3　MLサーバから配信されたメールのヘッダ（抜粋）

```
Date: Sat, 12 May 2001 14:16:05 +0900
From: Masanori Sakaki <masa@ascii.co.jp>
Reply-To: junk@takobeya.com
Subject: [Junk-ML 12345] VAX 11/780を探しています
To: junk@takobeya.com
Message-Id: <3AFCC715.62536D57@ml-svr.takobeya.com>
X-ML-Name: junk
X-Mail-Count: 12345
X-MLServer: fml [fml 4.0 STABLE (20010225)]; post only (only members can post)
X-ML-Info: If you have a question, send e-mail with the body
        "help" (without quotes) to the address junk-ctl@takobeya.com;
        help=<mailto:junk-ctl@takobeya.com?body=help>
```

書式を見ると、任意のヘッダを自由に定義できるように思うかもしれないが、ヘッダとして指定できる文字列（ヘッダ名）はRFCで規定されている。まともに配信しようと思ったら、正しいヘッダを指定しなければならないということだ。

しかし、X－で始まる名前のヘッダは例外で、ユーザーや管理者が自由に定義し、使用することができる。もちろん、そのヘッダが意味することが、相手側で正当に評価されるかどうかは別問題である。特殊なメールサービスなどによって配信されるメール、たとえばメーリングリスト（ML）サーバなどから送られるメールには、MLサーバが付加したX－で始まるヘッダが含まれていることが多い（**リスト3**）。これらは、MLサーバがユーザーに対し、何らかの情報を伝えるために使われている。

このメールは、ml-svr.takobeya.comというサーバ上で動いているfmlというMLサーバプログラムから送られたメールのヘッダである。後半のX－？？？というヘッダは、MLサーバが付加したヘッダであり、MTAによる配信や、MUAには何も関与しない。メーリングリストの参加者に向けて、メーリングリストに関する情報を知らせているだけだ。

■ヘッダのパラメータ

ヘッダ文字列が固定的なものであることはまれで、実際のメールでは、ヘッダ名に続けて、メールアドレス、ユーザー名、日付、ホストのドメイン名など、各種の配信時の情報が示されることになる。ヘッダに表示するために必要なこの種の情報は、sendmailの実行中に、マクロで参照することができる。したがって、ヘッダを定義するHコマンドは、各種マクロを参照してヘッダを作成することになる。

Received：ヘッダを生成するHコマンドを見てみよう（**リスト4**）。やたらに$が使われ、マクロが多数参照されていることがわかる。

この定義により実際に作成されるReceived：ヘッダは**リスト5**のようになる。ヘッダ定義の継続行の指定は、実際のヘッダにその通りに反映されている。

$に続けて名前を指定する形式は、sendmail.cfのほかの部分と同様に、マクロを参照するものである。メールアドレス（$u）、自身のホスト名（$j）などはそのまま展開されるだけなのでわかりやすいだろう。

通常の$によるマクロ参照のほかに、$？という表記がいくつかあるが、これは条件付きの展開処理である。$？に続くマクロに値が定義されている場合に、$.までの部分がヘッダ中に展開される。たとえば、「$？sfrom $s $.」という条件つきの参照では、sというマクロに値が定義されている場合に、「from $s 」がヘッダ中に出力される（もちろん、$sは実際の値に展開される）。sに値が定義されていなければ、何も出力されない。

▼リスト4　Received：ヘッダを生成するHコマンド

```
HReceived: $?sfrom $s $.$?_($?s$|from $.$_)
        $.by $j ($v/$Z)$?r with $r$. id $i$?u
        for $u; $|;
        $.$b
```

▼リスト5　生成されたReceived：ヘッダの例

```
Received: from mail-svr.ascii.co.jp (mail-svr.ascii.co.jp [192.168.1.17])
        by mail.takobeya.com (8.9.3/3.7W-2.8compat) with ESMTP id QAA01334
        for <masa@takobeya.com>; Mon, 7 May 2001 16:11:16 +0900 (JST)
```

■ヘッダの付加

フラグパラメータで指定したフラグが配信エージェントで指定されている場合、あるいはフラグパラメータが指定されていない場合は、そのヘッダは無条件に追加される。**リスト1**の例では、Received：とSubject：ヘッダは、フラグパラメータがないので、常に付加されることになる。一方、Date：ヘッダは、フラグパラメータでDが指定されているので、F＝でDが指定されている配信エージェントが選択された場合にのみ付加される。しかし、**リスト2**に示したsendmail.cfの例を見てみると、どの配信エージェントもDを指定しているので、結局、常に付加されることになる。

sendmailによるヘッダの付加処理には、いくつかのパターンがある。たとえば、Message-Id：は、なければ追加され、あればオリジナルのヘッダを残す。Subject：も同じように処理される。それに対してReceived：は、有無に関らず追加する。すでにReceived：ヘッダを含んでいるメールを処理した場合には、最初からあるヘッダに加えて、新しいReceived：ヘッダが追加される。Received：は、常にメールの先頭に付けられることになっている。複数のMTAで中継されたメールには、複数のReceived：ヘッダが含まれているが、これは新しい順に上から並んでいることになる。

また、sendmail内で特殊な扱いを受けるヘッダがいくつかある。たとえば、sendmailに−tというオプションを指定すると、宛て先情報（エンベロープ）をコマンドラインパラメータからではなく、ヘッダ中のTo：、Cc：から取得する。このような処理を行うためには、sendmailがTo：やCc：を特別な機能を持つヘッダと認識できなければならない。

個々のヘッダに関する、このような付加モードの違い、特殊な意味付けは、すべてsendmailプログラム中の構造体にハードコードされている。sendmail.cf中で指定できるパラメータではない。もしこれらの設定を変更したければ（通常は変更する必要はないが、方式の異なるメールシステムと相互乗り入れする場合などに変更することがあるかもしれない）、ソースコードを書き換えて再コンパイルする必要がある。

ヘッダについてどのようなモードが存在するのか、各ヘッダはどのようなモードが設定されているのかについては、sendmailのソース、コウモリの本などを参照してほしい。

■Received：ヘッダ

ここまで、いろいろなヘッダを紹介してきた。To：やFrom：といったヘッダは重要なものであるが、その内容は単純である。それに対して、やたら長くてよくわからないヘッダに、Received：がある。**リスト5**を例に解説しよう。

じっくり見れば、さほど複雑なものではない。fromのあとにあるのは、メールを送信、あるいは中継したメールサーバのドメイン名である。カッコ内にも同じような情報があるが、これは参照しているマクロが異なる。

byに続くのは、このメールを受信したメールサーバのドメイン名である。カッコ内は、動作しているMTAのバージョン情報である。つまり、このメールは、mail-svr.ascii.co.jpからmail.takobeya.comに送られたという情報がわかる。withに続くのは、その配信に使われたプロトコルだ。通常のSMTP配信であれば、SMTPかESMTPになる。idは各MTA上でメールを識別するIDである。Message-Idと異なり、各MTA上で異なるIDとなる。

forは、誰宛のメールであるかを示す。そのあとは、処理を行った日時である。これは、このReceived：ヘッダを生成したMTAが中継／受信処理を行った日時である。したがって、一連のReceived：ヘッダの時刻を見れば、メール配信にどれだけ時間がかかったかがわかる（もちろん、各MTAの動作するマシンの時計が正確に合っていればだが）。

要するに、mail.takobeya.comが、QAA01334というIDで、Mon, 7 May 2001 16:11:16 +0900 (JST)に、<masa@takobeya.com>宛のメールを、ESMTPを使い、mail-svr.ascii.co.jpから受け取ったということである。

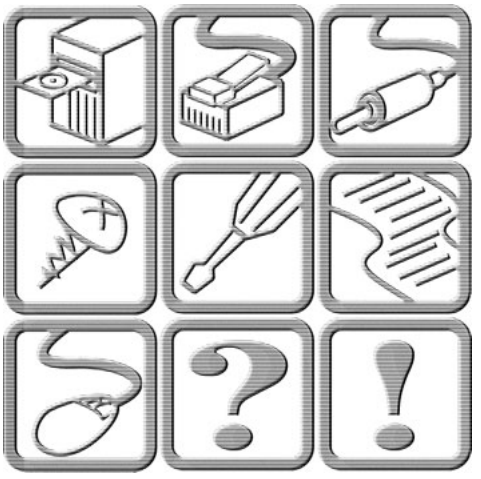
Received：を定義しているHコマンドを眺め、参照しているマクロをドキュメントで調べれば、似たように見える情報が、sendmailのどのような内部情報であるかがわかるだろう。

■次回は

ずっとやらねばやらねばと思っている我が家のインターネット用サーバマシンのリプレイスをいよいよやらねばなるまい。理由は明らかだ。梅雨入り前の快適な季節ゆえ、気温が高くなってきたからである。そうこうしているうちに、Pentium Ⅲ×2のWindows 2000マシンがくたばってしまった。スタンドバイモードのまま、永眠してしまったのである。これもどうにかしなければ。

sendmailについては、sendmail.cfは一段落ということで、エイリアスの解説でもしようかと思っている。

Try & [trái] Try [trái]



まだまだネットワークの話

文：政久忠由
Text : Tadayoshi Masahisa

底辺はともかく、ニュースなどで漏れ聞こえているように、ネットワークの上位部では、IPv4からIPv6への移行に向けた準備や実験やらが着々と行われている。最近ではどこだかは忘れたけど、IPv6の商用サービスも始められたようだ。

しかし本当にIPv6への移行が必要で、それがベターな選択であるかどうかは、僕自身かなり懐疑的であるし、似たような感想を持っている人も少なくないと思う。

IPv6

IPアドレスが足りない、場当たり的なアドレス割り当てによるルーティング情報の爆発、あとその他もろもろのIPv4の問題点を解消するために新たに開発されたIPv6の特徴を簡単にまとめておくと（詳細はRFC2460のほか関連RFCを参照のこと）、

①Expanded Addressing Capabilities

アドレッシングの大幅な拡大、IPv4は32ビット、IPv6は128ビット、加えてこの拡張により、アドレスの階層構造、自動アドレス構成機能、anyキャストアドレスの追加を含むマルチキャスト配信の効率化などを実現

②Header Format Simplification

ヘッダフォーマットの簡略化、IPv4での不要なフィールドを削除したり、オプションに移すことでハンドリングの処理効率やヘッダーサイズコストを軽減（アドレスフィールドが巨大なので標準的なヘッダサイズはIPv4の20オクテットに対して、IPv6は40オクテットとすでに2倍なんだけどね）

③Improved Support for Extensions and Options

拡張可能なIPヘッダ、従来のIPv4にあったフラグメント情報などのフィールドや新たな拡張情報を自由に追加できるようにヘッダの拡張をサポート

④Flow Labeling Capability

フローラベルフィールドを装備、QoS、つまりトラフィックの性質による帯域制御を効率よく行えるようにIPv6では標準ヘッダにその情報フィールドを配置している

⑤Authentication and Privacy Capabilities

セキュリティ機能、いわゆるIPSecのサポート

といった感じになる。これらは当該のRFCのイントロ部分に書いてあることで、僕が仕様を見てどうこう言っているものではない。

さてここからは1ユーザーとしての僕個人の感想。

■アドレスは足りないか？

まず①のアドレスの枯渇とルーティング情報の肥大化問題なんだけど、世界のすべてのネットワーク機器にグローバルIPアドレスは必要ないし、既存の不必要に割り当てたアドレスを回収して適切に割り当て直せば解決できると考

えている。

すべての電化製品がネットワークに接続されるし、NAT（ここではNAPT、IPマスカレードと区別せず総称としてNATと表現する）は双方向通信に対して適していないからという意見もあるけれど、セキュリティの面からもグローバル空間とプライベート空間は明確に分けるべきだし、各家庭にしろ、企業にしろ、そのプライベート空間とインターネットとの接続ポイントは集約されるべきである。そう考えるとプライベート空間のネットワーク機器の数に関わらず、必要なアドレスは各拠点に1個、多くて4個で十分だ。数万人の企業でも必要なグローバルアドレスは拠点の数でしかない。また個人ユーザーの多くは、ISP経由で動的にグローバルIPアドレスが割り当てられるようになっているのが現状だが、プル的な利用であれば、そのISP全体がプライベート空間で構築されていても何の問題もない。常時接続環境が一般化しようとしている昨今、むしろこのことによるセキュリティ対策を謳ったサービスを望むユーザーは少なくないと思う。すなわちISPでもアクセスユーザーの数だけグローバルアドレスが必要なんてことはどこにもないのである。まぁグローバルアドレスを望むユーザーがいないわけではないので、それなりに用意する必要があるものの、必要とされるアドレスはかなり少なくて済むだろう。

こんな風に考えてみると、本当に必要なIPアドレスはかなり少ないということがわかると思う。

一応この場合の問題点としては、H.323のようにIPヘッダ以外、要はパケット本体に発信元のIPアドレスを埋め込む仕様のプロトコルが少なからず存在し、いくつかのネットワークゲームのように外から内への通信などがあるため、標準的なNATでは特殊な処理が必要となることが挙げられる。あとポートの数には64K個の制限があるし、コネクションのトラッキングも大変といえば大変だ。でもハードウェアの性能を考えると1つのNAT装置に数千から1万のユーザーを想定してやってやれないことはない。それに問題のプロトコルは改善したほうがいいと思うし、ネットワークゲームもTCPでFTPのパッシブモードのように極力内側からのコネクションで対処して欲しいのだけれど、ダメかな？

これからはNATなしのピアツーピア双方向通信が必要だという意見もあるけれども、迷惑メールと同様にあまり歓迎できるものではないと思う。ほとんどの場合、定期的なスケジューリングによるプルで目的は果たせる。たぶん今後システムのユーザーサポートの一環として、サポートセンターからのリモートコントロール機能を利用する際に問題となるのだろうけど、これもプロトコルの実装レベルでどうにかしてもらいたいところである（全部パッシブモードを実装してね）。最近はゲートキーパーだったかな、H.323用の変換機能も整備されつつある。

サービスを公開する側も、サーバ群すべてにグローバルアドレスを与え、DMZに配置したりせず、すべてをフロントエンドシステムの内側のプライベート空間に隠すのが、定着しつつある状況だ。もちろんNAT技術を利用してである。これはアドレスが足りないからではなく、セキュリティや管理効率を向上させるための措置だ。つまりここでも公開するサーバの数だけグローバルアドレスが必要という状況ではなくなってきているのだ。

ルーティング情報については、アドレスの再割り当てをして、ルーティングの経路、つまり地域的な分配が重要になってくる。きちんとインターネットエクスチェンジへの接続レベルで、トップはアドレスの上位10ビット程度を使って1000の地域に分けられたりするととってもすっきりするのだが、なかなかそうはいかない。全部取り上げて再割り当てができれば容易なのだけれども。そういった意味ではIPv4だと実質的に難しい。ちなみにIPv6だと128ビットのうち、上位は地域やネットワークのまとまりをベースにした数階層を表すものとして、すでに決められている。この点に関してはIPv6のほうが優れているわけだけど、そもそもアドレス空間が爆発的に広がるのだから、本当にルーティングテーブルが現状より小さくなるかどうかは疑問。

マルチキャストは今後、動画のストリーミングなどで非常に重要になってくる。一般的なユニキャストだと300Kビットの動画を1000人に配信すると、100Mビットの帯域が消費される。マルチキャストだとこれが、300Kビットで済むのだから当然だ。IPv4でも定義されているのだが、ルータのサポートが遅れているのがネックとなっている。IPv6はマルチキャストに対してより進んだ機能定義をしているが、IPv4のレベルでダメってものでもない。

②のベーシックヘッダ情報の簡略化や③のオプション／拡張ヘッダ定義に関しては、ごもっともだと思う。④のQoS系については、現状1つ下位のレベル、イーサネットのレイヤレベルのQoSが実装されているのはよく見かけるようになったのだけれど（通常ルータを超えると効力がなくなる）、IPヘッダレベル（ルーティングレベルで効力を発揮する）では、IPv4だとTOSフィールドが該当する程度で何の対策も成されていない。まぁ今後必要な場面も多くなると思う。⑤のIPパケットレベルのセキュリティは、

IPv4にもIPSecという形で反映されているので、IPv6を推進する決定打にはならない。

結局IPv6は必要なわけ？

さて、IPv6は本当に必要なのだろうか？

インターネットの爆発的な広がりでIPアドレスの数、そしてこれまで紹介したようなさまざまな問題をIPプロトコルが抱えていて、早急な対策が必要だったことは確かだ。そのためいくつかの次世代IP候補が開発され、すったもんだの末、IPv6という形でまとまったわけだ。

しかしその間、セキュリティの点からもメリットがあるプライベートアドレスの利用が促進され、NATという技術がそれを後押しし、一気に広まってしまった。結果としてすべてのネットワーク機器にグローバルアドレスをなんてことをいうユーザーは希で、NAT越しにすべてのネットワークサービスが行えるようにという要望が大勢を占めようとしているのだ。

グローバルアドレスだと、サービスを公開できるというメリットがあるが、自前でサーバを用意してまでやりたいユーザーは越えるごく少数で、ISPなどのサービスを利用したほうがコストを含め何かと便利な状況だ。

つまるところインターネットの今後の展開など全体を勘案すると、IPv6の必要性はそれほど高くないのである。1人が100程度のネットワーク機器を有するとして、世界の人口を100億とすると10000億のグローバルアドレスが必要になるといった話は、詭弁としかいいようがない。管理のためにそれぞれのデバイスにユニークなIDを付けるといったことは、意味があるけれど、それはグローバルなIPアドレスであって欲しくない。家庭では、パソコンとなると操作や設定が面倒な部分もあるので、セットトップボックスという形で集約サーバがネットワーク機器を管理することになると思う。現在のモデム、ターミナルアダプタ、アクセスルータといった機器がその役割を担うハズだ。インターネット経由でも通常のダイヤルコールでも対応できて、接続されているすべてのネットワーク機器をコントロールできる環境である。この環境でグローバルアドレスが必要なわけ？　直接インターネット経由でそれぞれのネットワーク機器に個別に接続して操作するわけ？　セキュリティのためにIPSecったって認証サーバは？　それはちょっと無謀だね。

先のIPv6の特徴②から⑤、アドレス空間以外は確かに改良されているのだけれど、既存のIPv4環境にまったく存在しない機能ではないため一般に表立って強調されないようにいまひとつ押しに欠けるのが残念だ。

■NATを前提にした上位プロトコルの改良のほうが先でしょ

個人的には、IPレイヤよりもNATを前提にした上位レベルのプロトコルの改良が望まれていると思うんだけど、どうかなぁ。あとTCPのポート数もね。

64ビット、128ビット、256ビットCPUは、一部のサーバアプリケーションなどで本当に必要とされているが、いやぁIPv6はどんなもんでしょうねぇ。一部の必要な部分で使えばいいってものではないから。2台のLinuxマシンでIPv6を有効にして、あとWindows 2000にIPv6のプロトコルスタックをインストールして通信してみたんだけど、そもそもIPv6はスケールメリットプロトコルなわけで、netfilterとかの動作を確認したものの、もっとHopルーティングする巨大なネットワークで試さないと、やっぱちっとも面白くないね。

IPv6を否定する気はさらさらないんだけど、手放しに受け入れる気にもならない、それは次世代ネットワークプロトコルじゃなくて、IPレイヤだけの改良だからかもしれない。OSI参照モデル云々といった仰々しいだけで実現の見込みのないプロトコル群じゃなくて、物理レイヤはどんどん性能が向上しているわけだから、それに合った上位レイヤでないと、ねぇ。とは言っても当分はTCP/IP自体を引退に追い込むことはできそうにないのだけれど。

まぁ、何だかんだ言ってもそのうち企業の思惑でIPv6に移行してしまうと思う。たぶん、きっとね。

余談だけど、NATはIPフラグメントパケットを必ずバッファリングして組み立て直してから、変換、通過処理を行うようになっている。その理由はコネクションのトラッキングにある。アドレスとポートなどの関係を記録しておいて通過処理をするのだから、その情報が欠落しているIPフラグメントパケットのファーストパケット以外は対処に困るのだ。そのため必ず組み立て直すのである。その分、通常のルータよりオーバーヘッドが大きくなるがメリットもある。経路の途中でフラグメントが生じても、最終的に通過したパケットは、そのインターフェイスのMTUにもよるが、フラグメントが解消されるというわけだ。また同様の理由で不正なフラグメントパケットは、NATを通過することもできない。外部を発信元とする通信以外に、特別なフィルタリングを追加しなくても、いくつかの不正パケットを防ぐことができるのである。とにかく今後NAT抜きに考えるほうが問題だと思うんだけど。

ゲートウェイのネットワーク設定の前提

ちょっとIPv6の話が長くなってしまったけど、前回に引き続きADSLの導入に際して立ち上げたゲートウェイマシンの設定に移ろう。

前々回と前回は、ネットワークの基礎部分、MTUとかLinuxのネットワークオプションの説明をしたわけだけれども、我が家の環境では特にどこも調整しなくても最適な状態であったため、ちょっと手持ち無沙汰感が漂ってしまった。喜ばしいことなのだが、心持ち無念である。よく何かにつけてトラブルに見舞われる人がいるけど、実はそれがとても羨ましい。まぁ当事者の立場だと、楽しんではいられないと思うけれども。

■ここではxDSLモデムの設定はしないけど

ISPとの契約にもよるが、ほとんどのユーザーは動的にグローバルIPアドレスが1つ割り当てられるサービスだと思う。我が家もそうだ。

またDSLモデムとマシンとの接続形態には、大きく分けてブリッジタイプとルータタイプがある。僕はルータタイプを選択した。ブリッジタイプだと、デバイスとして扱い、通常PPPoEソフトウェアパッケージを接続するコンピュータに導入して設定する必要があるが、ルータタイプはひとつのホストとして見えるので特に接続ソフトウェアはいらない。コンピュータのネットワークアダプタと直接ケーブルで接続するか、ハブを介して接続するだけだ。我が家はハブに接続して利用している。

我が家の場合、ルータタイプでPPPoAが利用されていることは前回紹介したとおりなんだけど、実はこのDSLモデムは、設定を変更すればブリッジとしても機能するようになっている。しかしそれに変更するメリットは何ひとつ存在しない。DSLモデムのNATを無効にして完全にLinuxマシンでNAT、ポートマッピング、フォワード、ルーティングフィルタ系のコントロールを行うと便利そうではあるが、インターネット接続が完全にあるコンピュータに依存してしまうのはリスクが大き過ぎる。専属のゲートウェイマシンとしてならそれでもいいけど、我が家のゲートウェイとして機能するLinuxマシンは、実験場も兼ねていて構成の変更も著しく行われる性質を持っているので、最悪の場合を想定するとDSLモデムはルータタイプで独立運用できるようになっていないと非常に困るのである。そもそも初期費用がちょっと割高なルータタイプをわざわざ選択した理由はそこにある。

この前、DSLモデムの設定変更ツールを入手したので、早速変更してみようと思ったのだけれど、これがまた現在の我が家のネットワーク構成、内部ネットワークをゲートウェイマシン経由でDSLモデムに接続する、結果としてNATを2回通過する構成が意外としっくり僕に合ってしまったものだから、当初のDHCPを無効にして、内部ネットワークのアドレス範囲も調整して、といった目論見が不要となってしまった。使用していくうちに何か不便さを感じたら構成を変更しようと思っているが、今のところそれが特に見当たらない。

というわけで、DSLモデム側では、NATが有効になっているだけで、IPフィルタやポートマッピングはこれといってやっていない。外部からDSLモデム自体にtelnetとかでアクセスすることも制限されていたし、フィルタリング機能もチープなレベルでしかないので、積極的に使うことはたぶんないと思う。NATによって得られる外部を発信元とする接続のフィルタというセキュリティ効果だけでよしとしたい。まぁ一応フィルタを追加したいという場合などの方針は、次のゲートウェイマシンと基本的に同じであるので省略する。

少しPPPoEの話

そういえば、ちまたではADSL-PPPoE環境での該当インターフェイスのMTUサイズの設定で、さまざまな値が飛び交っているようだ。まぁ実装によって異なるのかもしれないが、規格の基本的なところをとりあえず押えておくとしよう。あくまで理論的な部分だけだが。

PPPoEヘッダは6オクテット（厳密さに欠けるけど、これ以下はなじみのバイトで表す）、それにPPPoEペイロードが続く。PPPoEペイロードに通常のデータ、この場合はIPパケットを収めるわけだけど、PPPoEペイロードの先頭2バイトはPPPプロトコルフィールドなので、全体としては8バイトのオーバーヘッドが生じることになる。一般的なMTU1500の環境なら、ADSLへの接続インターフェイスのMTUは1492バイトに設定する必要がある、ということになっているらしい。本来はこれだけで問題ないハズだ。IPヘッダ、TCP／UDPヘッダがオプションの利用で拡張されたとしてもそのインターフェイスを持つマシンで生成されたパケットなら、それらは自動的に考慮される。

僕はここに少々疑問点がある。PPPoEヘッダはどの段階で追加されるのだろうか？　申し訳ないが、この環境が

ないので推測しかできない。もし該当するインターフェイスを有するマシン上で生成されるのであれば、わざわざインターフェイスのMTUを変更する理由はないように思われる。通常PPPならPPPヘッダのオーバーヘッドを考慮してIPパケットを生成するからだ。ということは、DSLモデム側で追加されている可能性が高い。ブリッジ接続で特にPPPoEヘッダの処理をしないでマシン側のインターフェイスに入ってくるのではなく、その梱包を解いた状態のパケットがモデムとはやり取りされる。そうでないとわざわざマシン側のインターフェイスのMTUを調整する意味がない、という風に考えられる。たぶんそうなのだろう。

ではどうしてMTU1454とかMTU1412といった値が飛び交っているのだろうか？　真実はわからないが、多少ネットワークでの利用を考慮してのことかもしれない。

DSLモデムを接続したマシンを経由、ネットワークでゲートウェイ的な利用をする場合、そのマシン以外は、先のPPPoEを使ったMTU1492のインターフェイスを持っているわけではないので、通常のMTU1500のインターフェイスとしてIPパケットを生成してしまう。これをPPPoEに通すとなるとフラグメント化が生じることになる。すべてのマシンのMTUを調整すれば問題ないのだが、かなり面倒なので、ちょっとしたトリックが利用される。インターネットに対する通信のほとんどはTCPで、MTUが問題になるのはこのプロトコルがほとんどである。UDPは大きなデータのやりとりにはあまり使用されないし、ICMPもデータサイズは小さい。つまりTCP通信をどうにかすれば問題は解決したも同然だ。

MSSの書き換えによる問題解決方法

前々回紹介したと思うが、TCP通信は、ハンドシェイクによって約束事を取り交わし、セッションを確立してから送受信を行う。いわゆるTCP SYNパケットだ。そこではMSSがやり取りされていたことを思い出して欲しい。MSSは、IPパケットのヘッダ部を除いた実質データ部の最大サイズである。

いくつかのPPPoEデーモンは、このTCPハンドシェイク時のMSSを書き換えられるようになっているのだ。ネットワーク内のあるマシンがMSS1460でネゴシエーションを図ろうとしたとしても、ゲートウェイマシンを通過する際にMSS1452に置き換えるのである。この機能を利用すれば、ネットワーク内のマシンのMTUを調整することなく、少なくともTCP通信は問題なく行えるようになる。先の1412という値は、IPとTCPのベーシックヘッダ、拡張オプションヘッダの計、最大80バイトとPPPoEヘッダの8バイトを考慮した値だ。本来これはMSSでMTUサイズではないのだけれど、伝言ゲームをやっていくとそういう伝わり方をしちゃうことがある。まさにそのケースだ。1412をMTUとMSS書き換えの両方で利用するのはNG。あと1454は何を考慮したものかはようわからん（それぞれのベーシックヘッダのみ？かな）。

まぁ何が正しい最適値かは実際にチェックしてみないとね、ということだ。

なおnetfilterでもMSSの調整を行うTCPMSSモジュールが用意されている。FOWARD、OUTPUT、ROSTROUTINGで指定できる。IPマスカレードを利用している場合は、ROSTROUTINGのMASQUERADE設定の前か、FOWARDで次のようなエントリを追加するとよい。

```
iptables -A POSTROUTING -p tcp -syn -j TCPMSS -
clamp-mss-to-pmtu
```

最後の-clamp-mss-to-pmtuオプションは、検出したPath MTUから40（IPとTCPのベーシックヘッダサイズ）を引いた値がセットされる。このオプション以外に--set-mssがあり、これは数値を直接指定できる。たとえば、--set-mss 1412といった感じだ。

PINGを利用したIPフラグメントのチェック

前々回、前回はtcpdumpを利用したパケットのチェックを紹介した。ほとんどのLinuxディストリビューションに含まれているし、実際このツールに尽きるのだが、環境によっては存在しないかもしれないので、pingコマンド（TCP/IPスタックを装備したシステムなら必ずある）利用したチェックを紹介しておこう（どちらかというとWindows系OS向けなんだけどね。だからpingの発行元はWindows 2000でそこのpingコマンドを使っている。あとオプションが環境によってことなるので注意）。

pingはICMPのEcho requestを発行して、ターゲットからのEcho replyを受け取るピンポンツールで、ネットワーク状態のチェックに利用される。ここではIPフラグメントのチェックを目的とした使用を行ってみるとしよう。

これには、pingコマンドのパケットサイズの指定オプションとDF（Don't Fragment）オプションを併用する。

pingは、ICMPヘッダが8バイト、あとIPヘッダが20

バイト、計28バイトが基本ヘッダサイズとなっている。パケットサイズオプションで指定する値には、これらの値は含まれていないので、MTU1500の環境でパケットサイズ1500を指定すると、IPフラグメントが発生し、IPとICMP両方のヘッダ28バイトと1472バイトのデータを収めた1500バイトのファーストパケットと、ICMPヘッダのないIPヘッダ20バイトと残りのデータ28バイト分を収めた48バイトのセカンドパケットに分割して送信される。

```
>ping -l 1500 lin
cer > lin: icmp: echo request (frag 5:1480@0+) (len 1500)
cer > lin: (frag 5:28@1480) (len 48)
```

逆にpingのパケットサイズオプションで1472を指定すれば、MTU1500環境でフラグメントの生じない最大のICMPパケットが生成できるのである。実際、Windows 2000のpingコマンドには、パケットにDFフラグをセットするオプション（-f）が用意されていて、それと併用した場合、1472以上のサイズオプションを指定するとエラーになる。また経路のどこかでパケットの分割が必要な場合、DFフラグをセットしていれば、その趣旨のメッセージが送られるか、リクエストが返ってこない。

つまり、DFフラグをセットしたICMPパケットのサイズを変えながらチェックを行うことで、経路の最大転送ユニット（MTU）サイズがわかるというわけだ。ポイントは、pingのパケットサイズオプションで指定した値に28を足したサイズがMTUであるということだ。

たとえば、MTUが1492の経路であれば、pingのパケットサイズオプションは1464以下でないとリクエストは返ってこないハズだ。

DSLモデムを接続したLinuxマシンをゲートウェイ的に利用し、ネットワーク内の複数のマシンから使用する際、適当なマシンからインターネット上のpingに応答してくれる適当なサーバに対してチェックを行ってみるとよいだろう。先ほどのMSS書き換え機能は、ICMPパケットには通用しないので、実際の通過可能なMTU値が導き出せる。ただ経路によっては、DSLの部分以外で引っかかることもありうるので、リジェクトメッセージの送信元アドレスをよく確かめるようにしたい。

くれぐれもpingのパケットサイズオプションがMTUサイズではないことを頭におくこと。pingのオプション指定で1472が通れば、そのMTUは1500だ。MTUを導き出すには必ずpingのデータサイズに28を足すように。

ゲートウェイマシンのフィルタ設定

我が家のゲートウェイマシンは、ネットワーク的にもプライベート空間にあるわけだが、NAT（IPマスカレード）といくつかのフィルタリングを設定している。

まずPOSTROUTINGには、内部ネットワークを外側のインターフェイスに向けてマスカレードする設定。

```
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o eth1 -j
MASQUERADE
```

次にINPUTで外側からの不正な通信を排除する設定。

```
iptables -t filter -A INPUT -m state --state INVALID -i
eth1 -j DROP
iptables -t filter -A INPUT -m unclean -i eth1 -j DROP
```

FORWARDはポリシーをDROPに変更して、内部ネットワークの通信だけを通過させる設定。

```
iptables -t filter -P FORWARD DROP
iptables -t filter -A FORWARD -s 192.168.0.0/24 -j ACCEPT
iptables -t filter -A FORWARD -d 192.168.0.0/24 -m state
--state ESTABLISHED,RELATED -j ACCEPT
```

取り急ぎこれだけを設定している。実際にはDROPしている部分をLOGに渡してからDROPしている。そのため上記のDROPはLOGDROPで、次のchainの作成設定を最初に行っている。

```
iptables -N LOGDROP
iptables -A LOGDROP -j LOG
iptables -A LOGDROP -j DROP
```

もう少しlimitを使って細かくチェックしたほうがいいかなと思っているんだけど、今のところDROPされるパケットがほとんどないので放置している。次回までに考えておくとしよう。それとiptables-saveとiptables-restoreコマンドが今ひとつ信用ならないので、上記のコマンドはスクリプト/etc/sysconfig/netfilterとして作成し、rc.localで実行している。あとスクリプトの最後に“ echo 1 > /proc/sys/net/ipv4/ip_forward ”を忘れずに。

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15回
【～本】（～ぼん）
【コウモリ本】（こうもり・ぼん）
【バッタ本】（ばった・ぼん）
【ラクダ本】（らくだ・ぼん）
【フクロウ本】（ふくろう・ぼん）
【アルマジロ本】（あるまじろ・ぼん）
【マッチョ本】（まっちょ・ぼん）
【256本】（にごろ・ぼん）

～本

【～ぼん】

蒙昧なLinuxerを啓蒙するのに役立つとして一目置かれている書籍は、敬意をもって「～本」と呼ばれる。～部分には伝統的に動物名が入ることになっており、これはLinuxer向けオタク本を多数擁するO'Reilly社“ NutShell Handbook ”シリーズが、表紙に動物のイラストを使用しているため。現在では、「手間がかからない」、「版権が安い」、「なんとなく」、「自然保護に理解があると勘違いしてもらえる」、「気の迷い」などから、他の出版社もLinuxer向け技術書については“ 動物の表紙 ”を採用するのが一般的だ。これに反し、新潮流として注目されているのがFree Software Foundation関連本の「GNUシリーズ」で、『GNU Emacsマニュアル』では牛にまたがる麻原彰晃をみごとに描き出し、一転社会派路線を打ち出した。ちなみに編者は中学生くらいのとき、オライリーをオー、リアリイと読んでいたものである。あのころは私も若かった。それから、この際なので豆知識として教えておくが、「おおブレネリ」を替え歌にするとオライリーのテーマソングを簡単に作ることができる。わたしのお家はアメリカなのよ、ジェフ・ベゾスとは仲が悪いのよ～。

もっとも、作ったテーマソングを普通の人が歌う機会があるとは思えない。

コウモリ本

【こうもり・ぼん】

邦題『sendmailシステム管理』、『sendmailリファレンス』。どうせ読んだってsendmail.cfなんて書けない。そんな、飾るために買われる本の代表格（編集部注：編者の実体験に基づいた偏見に満ちあふれています）。めったに手に取られないという状況を見越して版元が採用したのがコウモリの表紙だった。これを本棚の天井近くに飾っておくと、コウモリが天井からぶらさがっているように見えてインテリアになる。しかも、いざというときには技術書としても使える。部屋が汚くなりがちなLinuxerにとってはピッタリのアイテムだろう。天井からコウモリがぶらさがっているような家には人も寄りつかないので、コミュニケーションの下手なLinuxerには人除けとして、一石三鳥にもなるという。コウモリ本の成功に気をよくした版元では、ゴキブリ本、ネズミ本の出版も検討している（編集部注：ゴキブリはありませんが、ネズミ、クモ、ムカデ、マダニはすでにあります）。

バッタ本

【ばった・ぼん】


バッタ飼う。食われる。

バッタというと安そうに聞こえるくせに、5000円もの大枚をふんだくる謎の本のこと。邦題『DNS & BIND』。かつて食費を削ってやっとの思いでBIND 4対応の旧版を買ったら、じきにBIND 8が出て泣きそうになった。そうか。人の財布の中身を貪り食うからバッタが描いてあるのか。あと、BINDは不正アクセスに遭うとバッタバッタと倒れていくので、そのさまをイメージしているのかもしれない。ぶ厚い装丁なので、アンチBINDポリシーのD.J.Bernsteinファンも枕として実用的に使える

一冊である。ただし、人前でやると「やっぱりdjbコミュニティの人は……」などと言われるので注意。

ラクダ本
【らくだ・ぼん】

邦題『プログラミングPerl』。同じラクダでも赤いものと青いものがあり、旧校舎2階トイレのいちばん奥の個室に入ると、突然どこからともなく「赤いラクダと青いラクダ、どっちがいい〜？」という声が聞こえてくる。赤いラクダと答えると、リファレンスなしの環境で複雑なデータ構造を表現するスクリプトを書かされ、赤い血を吐きながら死ぬ。青いラクダと答えると、難解なオブジェクト指向プログラミングを強制され、顔面蒼白になって死ぬといわれている。

ラクダ本の内容は、主にLarry Wallのダジャレと回想録（私はいかにしてこのグチャグチャな文法の言語を作ったか）に索引からなり、おまけとして、読んでも理解できないリファレンスページがついている。

フクロウ本
【ふくろう・ぼん】

ガン飛ばし……

邦題『詳説 正規表現』。表紙では、目つきの悪い2匹のフクロウ（ミミズクかもしれない）がこちらをにらみつけている。書店で平積みにしておくと客が避けて通るので、棚差しにする必要に迫られる一冊として有名。この本の中で紹介しているようなひねくれた正規表現を使うプログラマーは、フクロウ（コノハズクだったらどうしよう）のように目つきが悪くなって、しかも昼夜逆転の破綻した人生を歩みかねませんよという警告が込められたデザインとなっている。だいたい、メールアドレスの正規表現にあんなに文字数を割いてどうする。日本語ドメインも導入されることだし、[^\s]+\@[^\s]+でいいじゃないか。

アルマジロ本
【あるまじろ・ぼん】

邦題『UNIXシステム管理』。この本を読めば、あなたのLinux boxも全身をよろいでつつまれたアルマジロのように強固になりますよ、という意味の込められた表紙となっている。しかし、ひっくり返して腹を攻撃すれば一発でやられる動物をチョイスしたことが、はからずもシステム管理の実態を如実に表したかっこうとなってしまった。版元では現在表紙の再デザインを検討しており、次期採用候補として腹部も比較的強固なカメや、厚い脂肪で攻撃を無力化するハート様などが挙がっているという。

マッチョ本
【まっちょ・ぼん】

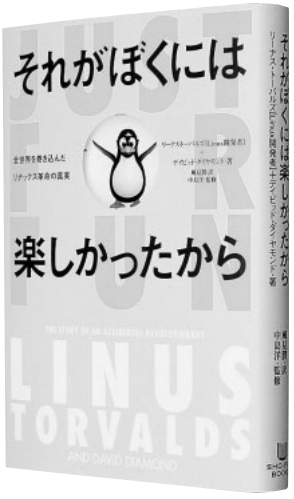
邦題『Flex入門』。ハダカの男が筋肉を見せびらかしているさまは確かに気持ち悪いが、性差別になるので「ホモ本」と呼んではいけない。

256本
【にごろ・ぼん】

アスキー社のペーパーバックスタイルシリーズのことで、『〜を256倍使うための本』の略称。他出版社の書籍のことばかり書いていると原稿全体がボツになる可能性があるので、バランスを取るため急遽追加されたのがこの項である。とかく大人というものは、まわりの人々のさまざまな事情を考慮して行動することが必要だ。編者は大人なので、こういった配慮ができるのである。

2001年5月号「Rubyで行こう！」では、この「256倍使うための本」が「256倍本」と略されていたが、正しい省略形は「256本」である。少なくとも東京・神田書店街界隈ではもっぱらこの略し方が通とされており、技術書のメッカ、書泉グランデののれんをくぐって「おやじ、256本入荷してる？」などと聞いたら、あなたももう立派な業界人（＝イヤなヤツ）。読者諸兄には、ひとつの雑誌内で見解が分かれていては混乱を招くと心配される向きもいらっしゃるだろうが、はたして信頼のおける本コーナーに書いてあることと、オブジェクトだのスレッドだの（編者には）理解不能なことばかり書いてあるコーナーとどちらが正しいと思われるか。などと適度に挑発的なことを書いておくと、またRubyコミュニティで「Linux Garbage Collectionは意味がわからない」とか発言されて反感を買う可能性が高い。だが、それもまたよし。こうなったらどっちが正しいか誌面で白黒つけようじゃないか！　と思ったら、赤松智也氏の連載は先月で終了していた。今回は私の不戦勝に終わったようだが、編者も男である。赤松氏が本誌に復帰し、正式な決着をつけられる日が来るのを待っている。読者のみなさんへのお知らせ：次号から赤松氏不在のあいだ、当コーナー編者によるRuby入門が始まります。乞うご期待！（編集部注：そんな予定はまったくありませんが、次号よりRubyの作者である、まつもとゆきひろ氏によるRuby入門の連載がスタートする予定です）。

Books



それがぼくには楽しかったから

リーナス・トーバルズ、デイビッド・ダイヤモンド著／風見潤訳／中島洋監修
小学館プロダクション
四六判／384ページ　　本体価格 1800円

Linus Torvaldsの手による初の著作である。子供時代のこと、コンピュータとの出会い、Torvalds一族について、フィンランドのお国柄、軍隊での体験（フィンランドには徴兵制がある）、プログラミングの持つ魅力、Linuxを開発した経緯と当時の生活、オペレーティングシステム論、Transmetaへの就職とアメリカ移住の顛末、Steve JobsやBill Joyといった業界の大立者との関係、ストックオプションでの一喜一憂、オープンソースと知的所有権についての考察、人生観などなど、とにかく自身について余すところなく語っている。また、共著者のDavid Diamondは、本人との会話や家族・友人への取材を通じて、Linusの人柄やその周りの人々が醸し出す雰囲気を伝えてくれる。

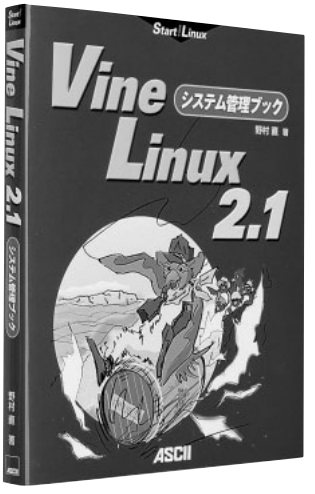
マスメディアを通じて広まった「偉大なことを成し遂げたにもかかわらず、とても謙虚でいい人」という通り一遍のイメージとはちょっと違うLinusに触れることができる。

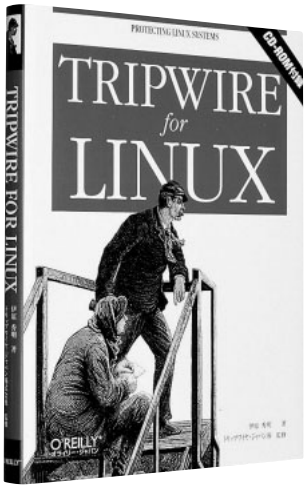
Vine Linux 2.1システム管理ブック

野村 直 著
アスキー
B5変形判／296ページ／CD-ROM2枚付き　　本体価格 2600円

唐突だが、Linuxをインストールしたときのことを思い返してほしい。マウントポイントを指定したり、rootユーザーのパスワードを入力したり、ユーザーアカウントを追加したり、いくつかの「システムの設定」をしたはずだ。つまり、意識するしないにかかわらず、この段階であなたはシステム管理者となったのである。そして、「インターネットに接続する」とか、「ほかのマシンとデータのやり取りをする」といった利用目的のレベルに合わせてシステム管理者としてグレードアップしてきたというわけだ。

本書は、これから「成長期」を迎えようとしているシステム管理者にお勧めしたい1冊。適切なシステム設定を行い、Linuxを有効に活用していくために必要な知識がギュッと詰め込まれている。付録CD-ROMには、Vine Linux 2.1.5のバイナリとソースが収録されている。巻末には、注目の新ツール「apt-rpm」の解説もアリ。

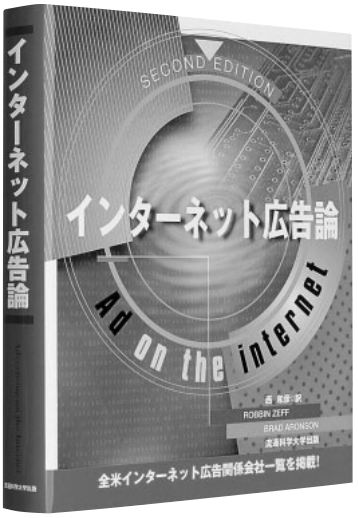




Tripwire for Linux

伊原秀明 著／トリップワイヤ・ジャパン株式会社 監修
オライリー・ジャパン／オーム社
B5変形判／160ページ／CD-ROM1枚付き　　本体価格 2000円

クラッカーは目的のコンピュータへの侵入に成功した際に、侵入の形跡を消そうとする。あるいは、次の侵入に備えてセキュリティに穴を開けておくこともある。さらには、別のサイトの攻撃のためにそのコンピュータを利用する準備をしておく場合もあるだろう。いずれにしろ、クラッカーは侵入したコンピュータ上のログファイルやシステム設定ファイルに変更を加える可能性が高いわけだが、管理者がそれに気づかないと、侵入されたこと自体が発覚しないということになりかねない。「Tripwire」は、ファイルに対する不正な変更を検出するためのセキュリティツールだ。不正アクセス自体を防ぐことはできないが、指定されたファイルやディレクトリが改竄されていないか検査することで、クラッキングの有無をチェックできる。よって、どのファイルをどのような基準でチェックするかが非常に重要になる。本書を読んで、しっかりとした対策を立てよう。



インターネット広告論

R・ゼフ、B・アロンソン 著／西 和彦 訳
流通科学大学出版
B5変形判／448ページ　　本体価格 7800円

Webページがバナー広告で埋め尽くされているのも、メールマガジンに広告が入るのも当たり前の世の中だ。ユーザーにとっては、広告は邪魔なものだと感じることも多いが、企業がインターネットを宣伝の場たりうるメディアとして認知したことによって、我々が享受できているサービスも少なくはないのだ。

本書は、企業がWebやメールなどを使い、効果的な広告・宣伝を行うにはどうすればよいかを細かく分析し、具体的な例を示しながら解説するものである。翻訳書であるため、日本国内での状況に即した内容だとは言い難い面もあるが、インターネット広告の特徴であるターゲティングや効果測定の重要性などは、少しでもインターネットに関わる宣伝・マーケティング担当者なら理解しておきたいポイントだ。オンライン広告で無駄な費用を出したくない企業には必携の一冊であろう。

Linux管理トラブル解決Q&A

Brian Ward 著
田郷 明／武藤健志 監訳
トップスタジオ 訳

オーム社

B5変形判／296ページ

本体価格 3200円

例題で学ぶLinuxプログラミング

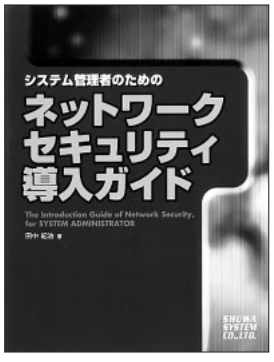
カート・オール 著
野村純子 訳
桑村 潤 監修

ピアソン・エデュケーション

B5変形判／508ページ

本体価格 4000円

システム管理者のための ネットワークセキュリティ導入ガイド



田中 紀治 著

秀和システム

B5変形判／302ページ

本体価格 2600円

フロッピー1枚から使えるPC-UNIX入門

後野 隆 著

広文社

B5変形判／280ページ／CD-ROM1枚付き

本体価格 3800円

読者の声

俺にもいわせろ！

そろそろ梅雨入りの時期ですね。担当の家はおかしな構造になっているうえ、換気設備がきわめて乏しいため、ふだんからとても湿っぽいのです。今年の冬には、カーテンなど数点がカビてだめになってしまいました。これ以上湿気を帯びたら、我が家はどうなってしまうんでしょう……心配で眠れません。

当たりますように！

Linux機のオール自作をしたいので、ぜひ6月号のプレゼントに当たりますように！！！！

（マックさん）

仕事が忙しく、久しくLinuxと格闘していません。近いうちに、ノートPCを購入する予定なので、ぜひともLinuxを搭載し、格闘したいです。ですから、6月号のプレゼントのHAND TRACKが当たると、モバイルが楽になるのですが……お願いします天の神様……。

（Sho－Shoさん）

6月号のプレゼントアイテムの中でも、VIA Cyrix IIIとマザーボードのセットは、ダントツの一番人気です。当選するといいですね＞マックさん。

「HAND TRACK」は、手のひらにすっぽりと収まるトラックボールです。これはコンパクトなうえに使う場所を選ばないので、モバイラーにとって魅力的なアイテムですね。ただ、ひとつだけ忠告しておかなければなりません。HAND TRACKはマウスとは操作感が大きく異なるため、手放しで「モバイルが楽になる！」とはいい切れません。うまく使えるようになるまでには、鍛錬と根気が必要となることでしょう。

ところで、ノートパソコンでLinuxといえば、日立からWindows 98がプレインストールされたPCと、「Do Office」をセットにした「はじめてみようLinuxパック」が販売されています。

Do Officeは、Vine LinuxとHancom Officeなどをワンパッケージにした、Linuxのオフィススイートです。LinuxとWindowsのデュアルブートにするためのセットアップガイドやサポートもついているので、心強いですね。

また、ターボリナックス ジャパンからも、IBM ThinkPad i Seriesに Windows MeとLinux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」が販売されています。インストールではなく、Linuxを使いこなすための格闘に時間を費やしたい方には、良い選択肢のひとつではないでしょうか？

社内イントラネット！

業務の合間を縫って細々とLinuxサーバを立ち上げ、ようやく社内でのイントラネット用Webサーバを構成。興味のない人にはなかなか理解はしてもらえないため、「趣味でやっている」とかの中傷もありました。しかし、最近ようやく有効性を理解してくれる人が増えてきました。次のステップはデータベースサーバとの連携。まだまだ時間がかかりそうですが、地道に進めていくしかありません。本職は別にあり、そちらも忙しいので……家族の理解が何よりも大切です。

（Hirorinさん）

学生時代、UNIXを少し扱っていたので、この知識を利用して社内イントラで利用できるLinuxサーバでも構築しようと思い、まもなく1年。設定をすれば誰でも動かせるファイルサーバやその他のデーモンは動作しているものの、当初から実現したいと思っている、Webベースで利用できるDBシステムの構築は滞ったままです。今回、6月号の「読者の声」に掲載された和田さんの記事を見て、病院勤めでお忙しいのに、ご自分でシステムを構築され、すごい人だと思いながら、自分も努力しなければ！　と思いました。

（うめうめさん）

皆さん、お仕事の合間を縫って、仕事で使うサーバを構築されているとは、ご苦労も一入のことでしょう。お疲れ様です。趣味でのLinuxも楽しいのですが、お仕事Linuxはカッコイイですね。

成長期

昔、Linuxをさわり始めた頃はslackwareくらいしかなかったディストリビューションが、今や数えるのも面倒なくらい増えている。いいことなのか、悪いことなのかは今の時点ではわからないが、個人的には、もっと絞られてほしいと思う。

（太田孝弘さん）

6月号特集1のディストリビューション乗り換え術は参考になりました。といっても、メインマシンの乗り換えはTurbolinux Workstationの7.0が出てからの予定ですので、もうちょっと先になりそうですが。まもなくカーネル2.4採用のメジャーディストリビューションが出揃いそうですので、一括紹介する特集を期待しています。

（森岡祐一さん）

㊗現在、Linuxに多くのディストリビューションが存在するのは、Linuxがまだまだ成長期である証拠でしょうか。多種多様なものが競争・成長を繰り返しているうちは、その成長過程に何かと無駄が多いものですが、見ていて楽しいものでもあります。それを直に感じることができるのは、喜びのひとつともいえるでしょう。

カーネル2.4採用ディストリビューションが出揃う時期が待ち遠しいですが、もうすこし時間がかかりそうですね。

楽しいことから順に

4月に社会人になり研修が始まったのですが、UNIX関係の仕事も多いので、個人でも何かしようと思い雑誌を購入しました。3年ほど前に一度Linuxに挑戦したことがあるのですが、派手な利用方法（TV自動録画サーバ等）が紹介されていて驚きました。

（野本大介さん）

テレビ自動録画サーバの記事をとても楽しみにしています。動画などのマルチメディア機能はMicrosoft Windowsのほうが先行しているように感じているのですが、最近はLinuxでも環境が整備されつつあって大変期待しています。興味あるところからLinuxを楽しむのに、Linux magazineはちょうどよくて気にいってます。

（遠藤 新さん）

㊗「自動録画サーバ」という言葉は、なんと魅力的な響きをしていることでしょう。やっぱり、楽しいことから始めることが、一番ですよね！？

4万円マシン

「超実践的　電脳お買い物ガイド」はとても良かったです。自作となると、どうしてもLinuxが動くどうか心配になりますが、今回の特集を見てみると案外OKだったのでちょっと驚きました。Cyrix IIIでLinuxが動くことは知ってましたが、ベンチマークを見る限り、ちょっと使う気にはなりませんね(^_^; それから、「箱の中のペンギンたち」は毎回感心しながら読ませてもらってます。一応、技術屋のたまごの端くれ(^_^;　なので、将来このような人たちのようになりたいなと思ってます。

（福重直行さん）

4万円マシンの記事ですけど、最近はホントにマシンを組みやすくなりましたよね。1年前の頃に比べて、グラフィックカードのサポートも充実してきているし。もう、最近じゃハードには考えが行かなくなって、もっぱらソフトウェアです。いかに使いこなすか！？　なんてね。そういうとき役にたつんだよなぁ、この雑誌！もうすっかり、生活の一部ですもん。また、来月もよろしく！

（小川恭生さん）

㊗こちらこそよろしくお願いします。4万円マシンを作るにあたっては、予算との戦いがそれはもう激しいものとなりました。数々のパーツが安価になった今となっては、フロッピーディスクドライブは、コストパフォーマンスの悪い記憶デバイスなんです。2.88Mバイトドライブや、スーパーディスク（LS-120）は普及しませんでしたし、このまま進化が止まってしまうのでしょうか。

Cyrix IIIも価格の割に性能がイマイチですが、発熱が少ないのは魅力的なポイントでしょう。チップに放熱ファンは要りませんって書いてあるんですよ。

Linuxの印刷環境整備

Linuxからエプソン製のWindows専用プリンタに印刷したいなぁ、といろいろ試行錯誤していたときに「Linux印刷環境徹底ガイド」が特集記事として掲載され、タイムリーでした。さっそくやってみました……が、うまくいかないのはなぜ〜？？

（杉本 崇さん）

㊗杉本さん、その後うまく印刷できましたでしょうか？　おかげさまで、「Linux印刷環境徹底ガイド」には、多くの反響をいただきました。担当は、Linuxマシンに Sambaをインストールして、Windowsから使うプリントサーバとして使っています。Linuxマシンからも印刷できるように設定しなければ。

付録CD-ROMに収録した

Red Hat Linux 7.1のインストール

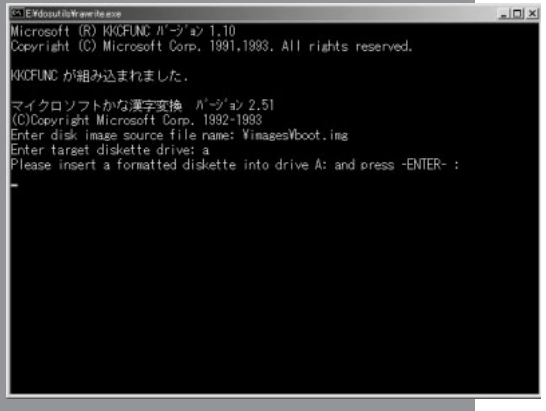
本誌付録CD-ROM収録のRed Hat Linux 7.1はFTP版です。非商用ソフトだけが含まれており、製品版を販売しているレッドハット株式会社からサポートを受けることはできません。
CD-ROMのメディアに不良があった場合は、お手数ですが（linux-cd@ml.ascii.co.jp）宛にご連絡くださるようお願いいたします。なお、Linuxの設定などについてのご質問にはお答えできませんので、あらかじめご了承ください。

インストールの前準備

これからRed Hat Linux 7.1のインストールを始めます。多くのネットワークカード、ビデオカード、モニタは自動認識されますが、自動認識されない場合に備えて、あらかじめハードウェアのマニュアルなどを用意しておくと、インストール中にあわてなくてすみます。

また、レスキュー用のフロッピーを作成するために、空のフロッピーを1枚用意しておいてください。CD-ROMブートができないマシンを使う場合は、別にインストーラ起動用のフロッピーも用意します。

さて、Linuxのインストールには、Linux専用に使えるディスク領域が必要です。ハードディスクでLinux専用に使える領域を作成するか、ディスクを増設してインストールに備えましょう。



ブートディスクの作成

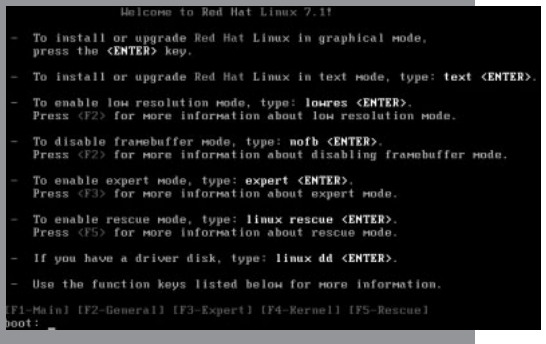
CD-ROMからインストーラを起動できない場合は、以下の手順でインストーラ起動用のフロッピーディスクを作成します（ここでは、フロッピーディスクドライブがA:であるとして解説します）。

（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。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どうかを確認して[Enter]を押すと、フロッピーの作成が始まります。



インストーラの起動

作成したフロッピーディスク、またはCD-ROMをドライブにセットしてマシンを再起動します。インストーラが起動して［boot:］というプロンプトが表示されたら、[Enter]を押します。しばらくすると、Xを使ったグラフィカルな画面が表示されます。

グラフィカルなインストール画面がうまく表示されない場合は、［boot:］の箇所で［text］とタイプして[Enter]を押します。こうすると、テキスト画面のインストーラが起動します。

使用言語の選択

4

インストーラを英語表示させるする場合は［English］を、日本語表示させる場合は［Japanese］を選択します。ここでは、［Japanese］を選択します。

キーボードとマウスの設定

5

デフォルトでは［モデル］に［Japanese 106-key］が、［レイアウト］に［Japanese］が選択されています。日本語キーボードを使用する場合は、このままの状態で［次］を押します。

次はマウスの設定です。PS/2タイプの2ボタンマウスを使う場合は［2 Button Mouse（PS/2)］をチェックします。［3ボタンマウスのエミュレート］は、2ボタンマウスの左右ボタンを同時に押すことで、3ボタンマウスの中ボタンを代用する機能です。これは便利な機能なので、チェックしたまま［次］を押します。

インストールタイプの選択

6

［ワークステーション］、［サーバシステム］、［ラップトップ］を選択すると、それぞれデスクトップ用、サーバ用、ノートPC用にLinuxがインストールされます。ただし、パーティションやブートローダLILOが自動で設定されるので、マシンをLinux専用に使う場合以外は、それらを柔軟に設定できる［カスタムシステム］を選択するのがよいでしょう。［アップグレード］は、すでにインストールされているRed Hat Linuxをアップグレードするための項目です。

ここでは［カスタムシステム］を選択したものとして、解説を進めます。

パーティション設定ツールの選択

7

［fdisk］は柔軟にパーティションを設定できますが、コマンドで操作するので初心者にはおすすめできません。

［Disk Druid］はやや柔軟性に欠けるものの、マウスを使ってパーティション設定できるので初心者におすすめです。

［パーティションの自動設定を実行し、データを削除］を選択すると、ハードディスクのデータやパーティション情報がすべて削除され、自動でLinux用のパーティションが作成されます。

ここでは［Disk Druid］を選択したものとして解説を進めます。

パーティションの作成

Linuxのインストールには最低限Linuxシステム用とSwap用の2つのパーティションが必要です。

まずはLinuxシステム用のパーティションを作成します。［追加］を押すと、小さいウィンドウが別に開かれるので、［マウントポイント］に［/］を、［パーティションタイプ］に［Linux native］を選択してます。Red Hat Linux 7.1をフルインストールすると、ディスクを約2.3Gバイト消費するので、フルインストールする場合はLinuxシステム領域を3～4Gバイト程度作成して［OK］を押します。

次にSwap用パーティションを作成します。［追加］を押して開かれるウィンドウで、［マウントポイント］にはなにも指定せずに、［パーティションタイプ］に［Linux swap］を選択して、マシンが搭載するメモリの1～2倍程度のサイズを指定します。

2種類のパーティションを作成したら、［次］を押します。

パーティションのフォーマットとLILOの設定

Linux用にフォーマットするパーティションを選択します。［フォーマット中に不良ブロックをチェック］をチェックすると、ハードディスクに不良箇所がないかどうかを調べながら、パーティションをフォーマットします。この欄はとくにチェックする必要はないでしょう。選択したら［次］を押します。

次にブートローダLILOのインストール先を設定します。まず、IDEタイプのハードディスクを使う場合は、［リニアモードを使用する（一部のSCSIドライブで必要）］のチェックをはずします。マシンをLinux専用に使う場合と、Windows 9x／MeとLinuxをデュアルブートする場合は、LILOのインストール先に［/dev/hda マスターブートレコード（MBR）］を指定します。Windows NT／2000とLinuxを共存させる場合は、LILOをインストールせずにフロッピーディスクからLinuxを起動します。

［ブートディスクを作成］は必ずチェックしておきましょう。

ネットワークとファイアウォールの設定

Red Hat LinuxをDHCPサーバからネットワーク情報を取得する環境で使う場合は、［DHCPを使用して設定］を、DHCPサーバのない環境で使う場合は［DHCPを使用して設定］のチェックをはずして、IPアドレスなどのネットワーク情報を入力します。なお、この画面ではダイヤルアップ接続の設定はできません。

次にファイアウォールの設定です。セキュリティの設定に詳しくないユーザーは、デフォルトのまま［次］を押してください。

使用言語とユーザー設定

Red Hatで使用する言語を選択します。Red Hat Linuxを日本語環境として使う場合は、デフォルトのまま［次］を押します。次の［タイムゾーンの選択］画面もそのまま［次］を押します。

さて、今度はユーザー設定です。Linuxには大きく分けて、ソフトウェアのインストールなどシステム管理を行う特権ユーザー［root］と、Webブラウズやメールの読み書きなど一般的な作業を行う一般ユーザーの2種類が存在します。

まず、Linux管理者のパスワードとして、［root パスワード］と［確認］に同じものを入力します。この管理者のログイン名は［root］です。

次に一般ユーザーのアカウントを設定します。［アカウント名］（＝ログイン名）、パスワードとして［パスワード］と［パスワード（確認）］に同じものを、［フルネーム］にユーザーのフルネームを入力して、［追加］を押します。

ユーザー設定を終えたら［次］を押します。［認証設定］もそのまま［次］を押します。

パッケージグループの選択

パッケージがいくつかのグループに分類されています。どのグループを選択してよいのかわからないユーザーは、最下段の［すべて］を選択して、すべてのパッケージをインストールしておくとよいでしょう。なお、［すべて］を選択した場合は、ハードディスクを2.3Gバイト程度消費します。

ここでは［すべて］を選択したものとして解説を進めます。

ビデオカードとモニタの設定

多くのビデオカードは自動で認識されます。インストーラが自動認識に失敗した場合は、リストの中から使用するビデオカードを選択します。

次はモニタの設定です。モニタの自動認識に失敗した場合は、モニタのマニュアルを参考にしながら、［Generic］の中からモニタのスペックを超えない解像度の［Generic Monitor］を選択します。なお、液晶モニタを使用する場合は、［Generic Laptop Display Panel］から無理のない解像度のものを選択します。

Xの設定とパッケージのインストール

Xを綺麗に表示させるには、［色深度］で［High Color(16Bit)］以上を選択する必要があります。［画面の解像度］には、モニタに合わせて適当な解像度を指定します。［色深度］と［画面の解像度］を選択したら、［設定のテスト］を押してXの表示テストを行います。GNOMEのデスクトップ画面が表示されれば、Xの設定は成功です。GNOMEのデスクトップ画面が表示されない場合は、［色深度］や［画面の解像度］の数字を低くしてテストしてください。

［デフォルトのデスクトップ環境を選択して下さい］には、ふだん使いたいほうのデスクトップ環境を選択します。［ログインの種類を選択して下さい］で、［テキスト］を選択するとテキストベースのログイン画面が、［グラフィカル］を選択するとグラフィカルなログイン画面が表示されます。ただし、Xの表示テストに成功してないユーザーは［グラフィカル］を選択してはいけません。

［次］を押すとパッケージのインストールが始まります。途中［続行するにはディスク2を挿入して下さい］と表示されたら、CD-ROMを入れ替えてインストールを続けます。

ブートディスクの作成

ハードディスクからOSを起動できなくなった場合にそなえて、緊急時用の起動ディスクを作成します。空のフロッピーディスクをセットして［次］を押すと、ディスクの作成が始まります。

起動ディスクの作成が終わるとインストール作業は終わりです。フロッピーディスクを抜いて、［終了］を押してください。CD-ROMは自動で排出されます。

お疲れさまでした。