J2EE対応Web アプリケーションサーバ製品 intra-martベースモジュール ver3.0

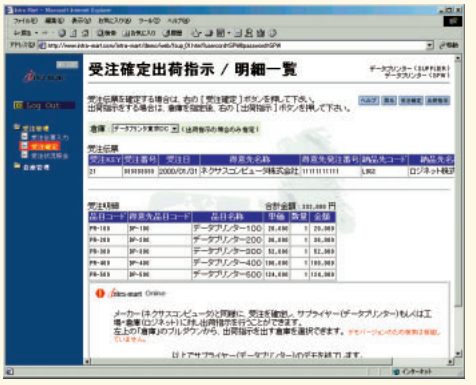
NTTデータ イントラマートは、J2EE対応Webアプリケーションサーバ製品「intra-martベースモジュール ver3.0」を7月末日より発売した。

従来のintra-martベースモジュールは、HTMLとJavaScriptを主体とした開発モデルをサポートしたWebアプリケーションサーバ製品だったが、新バージョンでJ2EE対応Webアプリケーション機能を追加、J2EEベースの開発モデルもサポートできるようになった。

J2EEベース開発モデルのサポートによってトランザクション処理が集中するようなWebシステムへの対応が改善でき、これまでより大きなECシステムおよびBtoB電子調達システム構築プラットフォームとしても活用できる。

また、intra-martベースモジュールがサポートする2つの開発モデルは、同一システム内で混在することもできるので、開発スピードを要求される部分はHTMLとJavaScriptによるページベースの開発モデルを使用し、トランザクションが集中する部分のみ切り出してJ2EEベースで開発するといった柔軟な対応がとれる。さらに、XMLにも対応した。

intra-martでは、このintra-martベースモジュールを利用して構築されたWebアプリケーションシリーズとして、Webイントラネット・ソリューション、Webフロントオフィス・ソリューション、Webエクストラネット・ソリューションなどが用意されており、これらをカスタマイズして独自のWebアプリケーションを構築することもできる。



発売日	2001年7月末日
発売	株式会社NTTデータ イントラマート
TEL	03-5549-2821
価格	80万円（1CPU）～
URL	http://www.intra-mart.com/

News Headline from 日刊アスキーLinux

www.Linux24.com

米Borland、「Kylix Open Edition」のダウンロードを開始

2001年7月26日

米Borlandは現地時間7月25日、「第12回 Annual Borland Developer Conference」において、「Borland Kylix Open Edition」英語版の提供を開始したと発表した。

「Borland Kylix」はLinuxネイティブのビジュアル開発環境。5月18日には日本語版も発表されている。

「Kylix Open Edition」には、Kylixで使われているコンポーネントクラスライブラリ“CLX”をGPLに準拠してオープンソース化した“FreeCLX”が搭載されている。

米BorlandのWebサイトよりダウンロード可能。なお、ボーランドによると、日本語版については現在開発中で、近日中に発表する予定があるとのことだ。

米Borland（http://www.borland.com/）

米Empower Technologies、LinuxベースのPalm IIIx用OSを開発

2001年7月20日

米Empower Technologies社は現地時間の7月17日、LinuxベースのPalm IIIx／IIIxe／Vx用オペレーティングシステム「Linux DA（DB v1.0）」をリリースすると発表した。Linux DAは、同社が組み込み用Linuxを米モトローラ社の「Dragonball」プロセッサ用に移植したもので、Palm OSと置き換えることができる。オンライン販売の価格はコンシューマー版が39.99USドル（約4900円）で、ソフトウェア開発キット（SDK）を同梱したプロフェッショナル版が59.99USドル（約7380円）。

Linux DAのデモ版は、同社のウェブサイトで無償でダウンロードでき、これにはマニュアルのほか、アドレス帳、スケジューラ、電卓、ゲームが含まれる。

Linux DA（http://www.linuxda.com/）
Empower Technologies
(http://www.empower-technologies.com/)

レッドハット、GNUProの最新バージョン「GNUPro 2001」を発表

2001年7月18日

レッドハットは7月18日、「GNUPro」の最新バージョン「GNUPro 2001」を発表した。「GNUPro」は、C／C++コンパイラやビジュアルデバッガ“Insight”、マクロアセンブラなどで構成されるソフトウェア開発ツールキット。デスクトップ／組み込み両方に対応している。

サポート対象となるプラットフォームは、Red Hat Linux 7.x、Solaris 2.5.1／2.6／2.7／2.8、HP-UX 10.20（PA 1.1）／HP-UX 11.0（PA 1.1）／HP-UX 11.0（PA 2.0w）、Windows NT 4.0、AIX 4.3.3である。

組み込み用途の対象プロセッサは、PowerPC、日立SH、富士通FR500、松下電器AM33、MIPS、ARMならびにStrongARM、Xscale、embedded Pentiumとなっており、SH-5、Pentium II／IIIにも対応予定だという。

「GNUPro 2001」のサポートは年間購入として販売され、Webベースのサポートや、最新バージョンへのアップグレード、パッチレベルのバグフィックスなどが含まれる。価格はホストがLinuxの場合は年間9995USドル、Linux以外では1万2500USドル。日本でのサポートの相談は、embedded-jp@redhat.comで受け付けている。

レッドハット（http://www.redhat.co.jp/）

ジャストシステム、「Choco」と「Muffin」のTechnology Preview 2を発表

2001年7月17日

ジャストシステムでは7月17日、Java表計算ツール「Choco Technology Preview 2」（以下、Choco TP2）とプレゼンテーションツール「Muffin Technology Preview 2」（以下、Muffin TP2）を公開した。

バージョンアップで加えられた主な新機能はそれぞれ以下のとおり。

「Choco TP2」

- 印刷、印刷プレビュー
- シート追加、削除などのシート操作
- 数式でセルを参照する際に、直接入力しなくてもセルをクリックすることで参照が可能になった
- 置換
- 行、列の幅を、入力された文字に合わせて最適な大きさに調整
- 手動での再計算
- ファイルのタイトルやキーワードの設定
- シートの表示倍率や行、列ヘッダなどの表示／非表示といった画面表示設定

「Muffin TP2」

- スライドショー
- 段落単位でのエフェクト
- 印刷
- ビューアとプレゼンテーションをまとめてJARファイルに保存することが可能になり、「Muffin」がなくてもJava環境があればプレゼンテーションが可能になった
- 枠飾り
- 目次の自動生成機能
- ハイパーリンク、ブックマークの挿入
- 指定したURLからファイルを開くことが可能になった

インストールに際しては「Java 2 SDK, Standard Edition, v 1.2」または「Java 2 Runtime Environment, Standard Edition v 1.2」が必要となる。また、「Choco TP2」、「Muffin TP2」ともに、それぞれの旧バージョンで作成したファイルを読み込ませることはできないそうだ。ダウンロードはArkサイトのダウンロードページから可能。

ジャストシステム
(http://www.justsystem.co.jp/)
Ark Site（http://www.justsystem.co.jp/ark/）

「Konqueror」がActiveXに対応

2001年7月12日

7月10日にKonqueror.orgに掲載されたリリースによると、「Konqueror」をActiveXに対応させる「Reaktivate」というモジュールが公開された。このモジュ

ールを使用することで、Shockwaveに対応した動画などのactiveXコンテンツをKonquerorで実行することが可能になるという。

KonquerorはKDE標準のWebブラウザ／ファイルマネージャ／ファイルビューア。HTML4.0やJavascript、CSS 1および2にも対応しており、Netscapeプラグインを使用することも可能になっている。

このReaktivateモジュールはWineライブラリに依存してActiveXコントロールを動作させるもの。すべてのActiveXコントロールに対応できているわけではなく、ほぼ確実に動作したのは、「Shockwave Flash 5」、「Shockwave Player 8」、「LivePics」に限られたようだ。プレスリリースには実際に動作している状態のスクリーンショットが添付されている。また将来的には「Quicktime」を含むさまざまなコントロールに対応させる方針だという。しかし、ActiveXコントロールはWineのレジストリのすべてにアクセス可能であったり、ActiveXをダウンロードする際にポップアップのメッセージは現われるものの、署名を確認できないといったセキュリティ上の問題があり、root権限では使用しないことや、信頼できるサイトのActiveX以外はダウンロードしないことを推奨している。

Konqueror（http://www.konqueror.org/）

サイバーソリューションズとミラクル・リナックス、ナレッジポータルシステム構築で提携

2001年7月10日

サイバーソリューションズは、ミラクル・リナックスと、販売／マーケティングにおいて提携したと発表した。今回の提携により、サイバーソリューションズのナレッジポータル構築ソフト「Cyber Finder」を「Miracle Linux」に対応させる。

「Cyber Finder」は、データベースや文書ファイル、Webサイトの検索／管理をWebブラウザ上で簡単に行なうことができるソフトウェア。すでに、「Cyber Finder」に対応した“Miracle Linux on Express5800”が、NECソリューションズより出荷されている。両社は、“コストパフォーマンスが高く安定したナレッジ・ポータルシステムの構築について”今後とも協業していくという。

サイバーソリューションズ
（http://www.cybersolutions.co.jp/）
ミラクル・リナックス
（http://www.miraclelinux.co.jp/）
NECソリューションズ
（http://www.sw.nec.co.jp/）

オープンソースの.NETプロジェクトが登場

2001年7月10日

7月9日にFree Software Foundation（以下、FSF）が発表したプレスリリースによると、Microsoft .NETフレームワークのコンポーネントをフリーソフトウェアとして開発するプロジェクト「Mono」と「DotGNU」が開始された。

Microsoft .NETは、XMLによるWebサービスのためのプラットフォームを提供するもの。データをW3C標準に基づいたXMLでやりとりすることで、プラットフォームに依存せずにデータを交換することができることを目指す。実際、Microsoftは.NETをWindowsだけでなくFreeBSDへの実装を進めており、これはMicrosoft独自の「共有ライセンス」に基づいて頒布される。

今回発表された「Mono」と「DotGNU」はどちらもオープンソースプロジェクトとして開発が進められている。「Mono」は米Ximianが、「DotGNU」はDavid Sugar氏が中心となって進めており、C#コンパイラやCLR（Common Language Runtime）コンパイラ、ライブラリ、分散サービスやユーザー認証を可能にするツールなど、Microsoft .NETフレームワーク互換のツールを実装していくという。これらのプロジェクトの成果は、GPLやLGPLで公開されるそうだ。

Microsoftでは、6月18日に「Visual Studio .NET Beta 2」をリリースしているほか、各社の開発ツールも.NETへの対応を意識し始めている。一部には.NETの完全な実装がまだない時点でオープンソース陣営が開発を始めることを危惧する声もあり、また、ライセンス上の問題が起こりうる可能性も否定できない。今後の展開を注意深く見守る必要があるだろう。

Mono（http://www.go-mono.net/）
GNU Project（http://www.gnu.org/）

VA Linux、ハードウェア事業から撤退 153人をレイオフ

2001年6月29日

米VA Linux Systemsは現地時間6月27日、2001年7月10日をもってハードウェア事業から完全に撤退すると発表した。また同時に、全社員の約35%にあたる153人をレイオフすると発表した。

今後は、企業の共同ソフトウェア開発プロジェクトを支援する「SourceForge OnSite」サービスを事業の柱とする。また、ストレージデバイスやリモートサーバ管理用のLinuxソフトウェア開発、コンサルティングサービスを行っていく。

「Slashdot」、「SourceForge」、「Freshmeat」など、同社が出資する一連のOpen Source Development Network（OSDN）サイトは継続する。

リリースによると、「ハードウェア事業からの撤退によって、売り上げは大幅に減少するが、資金の流出も減少すると期待している。当初は四半期で800万USドル、将来的にはそれ以上の節約を見込んでいる」とのことだ。

29日付けで、Web上でも日本法人としてのコメントを発表している。内容は、「ハードウェアとソフトウェア一体での供給／サポートに対する日本国内の多くの需要に対し、従来と同様に対応していく」というもの。

サーバ本体の供給については、複数のベンダーと協議中だという。当初、日本法人としての正式なコメントは、それがはっきりした時点で発表する予定だったが、ユーザーやパートナー各社からの問い合わせが殺到したため、急遽Web上に掲載することになったという。

日本法人としては、OSDNサイトも含め、従来どおりの体制を維持していくとのことである。

米VA Linux Systems
（http://www.valinux.com/）
VA Linux Systemsジャパン
（http://www.valinux.co.jp/）

Hardware　発売日　2001年6月7日

OSにLinuxを搭載したサーバパッケージ ExpressSelectionPack Linuxベースパック

URL　http://www.express.nec.co.jp/linux/

発売　日本電気株式会社
TEL　03-3455-5800
価格　18万8000円〜

NECは、サーバ本体にハードディスク、ディスプレイ、OSおよびミドルウェアを組み合わせ、導入を容易にしたパック商品「ExpressSelectionPack」を6月7日より発売した。

本商品でLinuxに対応しているのは、「Express5800/110Ee Linuxベースパック」と「Express5800/120Le Linux DBベースパック」の2種類。同社のExpressシリーズのサーバにLinuxを搭載して運用管理ソフトをパックにした。サーバの仕様は、110Ee LinuxベースパックがCPU Celeron 733MHz、メモリ64Mバイト、HDD 20Gバイト、インストールされるOSはTurbolinux Server 6.5。120Le Linux DBベースパックは、CPU Pentium III 1GHz、メモリ128Mバイト、HDD 9.3Gバイト×3（アレイ）、インストールされるOSはMiracle Linux Standard Edition Version 1.0、データベースとしてOracle8i Workgroup Server（5指名ユーザー）をバンドルしている。

Hardware　発売日　2001年6月20日

LinuxベースのDBアプライアンスサーバ SUN COBALT RaQ with Sybase ASE

URL　http://www.atckk.co.jp/

発売　エー・ティー・シー株式会社
TEL　045-593-7091
価格　70万円前後〜

エー・ティー・シーはサン・マイクロシステムズ社のSUN COBALT RaQ4Jにサイベース社のSybase Adaptive Server Enterprise（以下ASE）for LinuxをプリインストールしたLinuxベースのDBアプライアンスサーバ、「SUN COBALT RaQ with Sybase ASE」を6月20日より発売した。

Sybase ASEはOLTP、DSS処理、インターネットなどさまざまな環境に対応するエンタープライズ向けRDBMS製品。新バージョンでは動的SQLのパフォーマンスやデータ保護機能が強化された。管理はGUIツールSybase Centralで行う。8月末までの受注分に対してはNTTコムウェア社製のHAクラスタ・ソフトウェア「Progart／Local Cluster Enterprise」の試用版が同梱される。

Hardware　発売日　2001年7月27日

信頼性を高めたLinuxインターネットサーバ apricot LX230／LX630

URL　http://www.melco.jp/

発売　三菱電機株式会社
TEL　03-3218-9064
価格　27万8000円〜

三菱電機は、同社apricotシリーズに、ジャーナルファイルシステムと電源シャットダウン機能を追加したLinux搭載モデルLX230とLX630を7月27日より発売した。

主な仕様は、CPUはLX230がCeleron 766MHz、LX630がPentium III 1GHz、メモリは128Mバイト、HDDはLX230が20Gバイト、LX630が40Gバイト。

LX230はジャーナルファイルシステム「ext3」を採用、システムの異常終了後の再起動時のディスク修復時間が短縮される。LX630は2台のHDDでミラーリングすることで信頼性を確保する。さらに、電源ボタンを押すだけで自動的にシャットダウン（OS終了、電源遮断）を行える。サーバ管理にはHDE Linux Controller 2.4 Standard Editionを採用し、ブラウザによる管理ができる。

搭載しているOSはTurbolinux Server 6.5、価格はLX230が27万8000円、LX630が37万8000円。無停電電源装置付属モデルもある。

Hardware　発売日　2001年8月中旬

マルチOS対応ディスクアレイ hp surestore バーチャル・アレイ7400

URL　http://www.jpn.hp.com/storage/san/

発売　日本ヒューレット・パッカード株式会社
TEL　0120-352-239
価格　1620万円〜

日本ヒューレット・パッカードは、同社独自のバーチャル・アレイ技術を採用した新製品「hp surestoreバーチャル・アレイ7400」を7月1日より受注開始、8月中旬より出荷する。

hp surestoreバーチャル・アレイ7400は、ネットワークを活用した大規模ストーレージシステムを効率的に管理するためのソリューション・フレームワークFSAMへの対応機能を強化した。

7400は、2Gbpsのファイバ・チャネル・インターフェイスに対応、10000-15000rpmのディスクを採用、Linuxを含む複数のOSに対し1つの筐体内で1024の論理ボリューム、最大システム容量7.7テラバイトのストレージ管理が可能。ディスク増設用筐体hp surestore disk system 2400（別売）を用いて、ファイバ・チャネルディスクドライブを最大105個搭載できる。3Uの筐体に最大15個のディスクドライブを内蔵し、高さ2mの19インチラックに最大195ドライブを搭載できる。

Hardware

発売日 2001年8月1日

SANストレージ・ソリューション hpネットワーク・ストレージ・アプライアンス

URL http://www.jpn.hp.com/storage/san/

発売 日本ヒューレット・パッカード株式会社
TEL 0120-352-239
価格 3200万円〜

日本ヒューレット・パッカードは、ホスト接続後すぐに利用できるSANストレージ・ソリューション「hpネットワーク・ストレージ・アプライアンス（hp nsa）」を8月1日より発売した。

hp nsaは、ユーザーごとに最適化したSANストレージシステムを包括的なサービス・プログラムとともに提供するソリューション。製品は、hp surestoreディスク・アレイ、テープ・ライブラリ、ファイバ・チャネル・スイッチ、およびhp OpenViewストレージ管理ソフトウェアがインストールされた管理ステーションで構成される。ユーザーは、接続ホストの台数と各ホストごとに必要なストレージ容量の規模に合わせてストレージ装置を選択できる。すべてのハードウェア、ソフトウェアは工場で統合、構成、テスト済みで、ファイバ・チャネルで接続すればすぐにストレージを使用できる。

サポートするOSは現在Windows NT 4.0だけだが、年内にLinuxも対応する予定。最小構成の価格はテープ構成が3200万円、ディスク構成が3800万円、テープ＆ドライブ構成が4700万円。

Hardware

発売日 2001年7月2日

PCIバス対応RS-232C通信ボード COM-2／-4／-8（PCI）H

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

発売 株式会社コンテック
TEL 03-5628-9286
価格 2万3000円〜

コンテックは、PCI準拠のRS-233Cシリアル通信インターフェイスボードの新製品COM-2／-4／-8（PCI）Hを7月2日から発売した。

COM-2／-4／-8（PCI）Hは、同社従来製品のCOM-2／-4／-8（PCI）より通信速度を高速化、50〜921.6Kbpsをサポートした（従来製品は50〜230.4Kbps）。またFIFOメモリ容量も従来よりアップし、128Kバイトになった（従来製品COM-2／-4は16Kバイト、COM-8は64Kバイト）。

価格はCOM-2（PCI）Hが2万3000円、COM-4（PCI）Hが2万9000円、COM-8（PCI）Hが4万2000円。添付されている標準COMドライバCOM-DRV（W32）はWindows 2000／NT 4.0／Me／9x対応ドライバだが、同社ホームページよりLinux対応標準COMドライバCOM-DRV(LNX) for Linuxをダウンロードすることでも対応している。

Hardware

発売日 2001年7月3日

低価格マルチOS対応ストレージ Gateway GS400

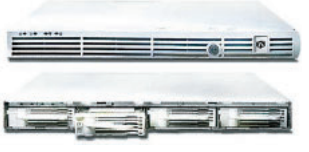
URL http://www.gateway.jp/

発売 日本ゲートウェイ株式会社
TEL 0120-54-2000
価格 19万9000円〜

日本ゲートウェイは、簡単にストレージを増設でき、ブラウザで設定やデータ管理ができるNAS製品「Gateway GS400」を7月3日より発売した。

Gateway GS400は、1Uラックマウント型で、最大240Gバイトのディスクを搭載可能なNAS製品。HDDの構成によりモデル140／260／460の3モデルが用意されている。

140はHDD 40Gバイト×2（40GバイトHDD最大2基増設可）、260はHDD 60Gバイト×2（60GバイトHDD最大2基増設可）、460はHDD 60Gバイト×4（HDD増設不可）を搭載。CPU Celeron 566MHz、100Base-TXは全モデルに共通している。RAID構成はJBOD、RAID 0、1、5をサポート（ただし、RAID 5にはHDD 3基以上が必要）している。



モデル140が19万9000円、モデル160が26万9000円、モデル460が39万8000円。

Hardware

発売日 2001年7月6日

Pentium III搭載デスクトップサーバ 4EXシリーズ Standardモデル

URL http://www.logicaleffect.com/

発売 ロジカルイフェクト株式会社
TEL
価格 8万7000円

ロジカルイフェクトは、デスクトップサーバ「4EX」のStandardモデルをモデルチェンジして7月6日より発売した。

今回変更された仕様は、CPUをCeleron 766MHzからPentium III 800MHzに変更。メモリバスをPC100からPC133に変更し、メモリ—CPU間の負荷を軽減した。また、HDDをUltraATA/66からUltraATA/100対応20Gバイト（最大100Gバイト）に変更、同社のラックマウントサーバやクラスタサーバで実績のあるSiS社の統合チップセットを採用し、部品点数を減らして価格を9万9700円から8万7000円に値下げした。そのほかの仕様はメモリ128Mバイト（最大1Gバイト）、CD-ROMドライブ、100Base-TXなど。各種Linux、FreeBSD、Windows NT 4.0／2000などのOSプリインストールモデルがある。OSをインストールしないオプションもある。

Software　発売日　2001年6月21日

Web上にデータベースを簡単作成 FTIS Webかんたん検索くん PRO

URL　http://www.fujitsu-ten.co.jp/FTIS/

発売　富士通テン情報システム株式会社
TEL　078-682-2017
価格　4万9800円

富士通テン情報システムは、データをCSVの形で活用して対話形式の操作でWeb上にデータベースを作成できるソフト「FTIS Webかんたん検索くん PRO」を6月21日より発売した。

FTIS Webかんたん検索くん PROは、Web上でデータの検索、修正、追加ができるデータベースソフト。メールと連携して関係者への一斉送信もできる。

サーバにインストールして複数で使用できるので、社内イントラネット上で共有すべきデータを公開して業務の効率化やペーパーレス化が図れる。

HTMLやPerlなどの知識は一切不要。クライアントフリーで約3000人程度の規模にまで対応する。データベースは作成時にパスワードを設定したり、IPアドレスでのアクセス制限ができる。

サーバ環境は、Red Hat系ディストリビューションでApache 1.3以上、Perl5以上が必要。

Software　発売日　2001年6月26日

知的資産の活用を加速する ConceptBase ナレッジソリューション・シリーズ

URL　http://www.justsystem.co.jp/ebiz/

発売　株式会社ジャストシステム
TEL　03-5412-3939
価格　780万円〜

ジャストシステムは、知的資産を有効活用する「ConceptBase ナレッジソリューション・シリーズ」を6月26日より発売した。

まず最初に販売されるのは、「Net-It Knowledge Search Plus」と「KnowledgeMarket with CB」をパッケージ化した製品。

Net-Itは、ファイルサーバに蓄積されたあらゆるファイル形式のドキュメントを自動的にWeb上に公開し、共有／活用するソリューション。KnowledgeMarketは、あらゆる情報や知識を人と関連付けてKnow Who（誰が何を知っているか）データベースを構築する人材活用型ソリューションで、業務ノウハウ、アイデアなどの人の頭の中にある情報の共有／活用を可能にするための製品。

価格はNet-Itが1パッケージ（クライアントフリー）で780万円。KnowledgeMarketが1パッケージ（100クライアントライセンス付き）で780万円。リースによる提供もある。

Software　発売日　2001年7月19日

Web型グループスケジューラ e schedule Linux版

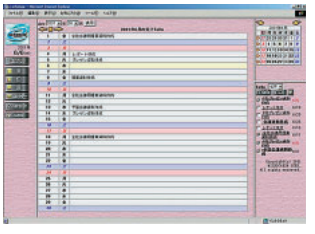
URL　http://www.mvi.co.jp/

発売　株式会社メディアヴィジョン
TEL　03-3222-3908
価格　3万4800円（20ユーザー）〜

メディアヴィジョンは、使いやすさを追求したWeb型グループスケジュール管理ソフト、「e schedule」を7月19日より発売した。

e scheduleは、機能をスケジュール管理とToDo管理に絞って「入力のしやすさ」、「見やすさ」と「携帯性」を追求した。主な機能は、①日、週、月単位の表示切替え、組織単位、グループ単位での表示切替え機能②個人のタスク管理からグループ単位での作業項目の整理ができるToDo機能③ToDo項目へのアラームを設定するアラーム機能④ZaurusやPalmとのリンク機能⑤i-modeからのアクセス機能など。

サーバ環境で必要な条件は、Red Hat Linux 5.x／6.x／7.xまたはTurbolinux 6.xとApache Webサーバ。1パッケージで20ユーザーまで使用可能。21ユーザー以上で使用する場合には5ユーザー追加ライセンスも8700円で用意されている。



Software　発売日　2001年6月28日

クロスプラットフォーム対応RDBMS Borland InterBase 6 日本語版

URL　http://www.borland.co.jp/interbase/

発売　ボーランド株式会社
TEL　03-5350-9380
価格　4万9800円〜

ボーランドは、Windows／Linuxに対応したRDBMS「Borland InterBase 6」日本語版を6月28日から発売した。

InterBase 6には、Javaによるアクセスをサポートする JDBCドライバInterClientや、Windows／Linuxアプリケーションからのアクセスをサポートする ODBCドライバが付属しており、C／C++、Java、Object Pascalなどでデータベースアプリケーションを開発できる。

今回、①CD-ROM、DVDなどの読み込み専用メディアでのデータベース配布②SQL適合のDATE、TIME、TIMESTAMPや高精度数値型のサポート③InterBaseをアプリケーションに組み込むためのインストールコンポーネント④ライセンシングサポートなどが追加された。対応するディストリビューションは、Red Hat Linux 6.2／7.0、Turbolinux 6.0／6.5、SuSE 7.0、 Meister Linux Mandrake 7.2、 Miracle Linux SE 1.1、LASER5 Linux 6.4。

Software

Java用RADツールのスタンダード Borland JBuilder 5

発売日 2001年7月12日

発売 ボーランド株式会社
TEL 03-5350-9380
価格 6万8000円〜

URL http://www.borland.co.jp/jbuilder/

ボーランドは、同社のJavaビジュアル開発ツール最新バージョン「Borland JBuilder 5」日本語版を7月12日から発売した。

JBuilder 5では、新たにIBMのWebSphereをサポートし、BEA WebLogic 6、Borland AppServer 4.5などの最新バージョンのアプリケーション・サーバに対応。Microsoft Visual SourceSafeとの統合機能の追加、CVSなどのバージョン管理機能の統合により大規模開発支援機能を強化した。また、XML対応アプリケーションの開発機能も追加された。

製品は、Javaの学習に適したJBuilder 5 Personal（4800円）、Webアプリケーション開発に適したJBuilder 5 Professional（6万8000円）、大規模分散アプリケーション開発用JBuilder 5 Enterprise（36万円）の3形態からなる。

動作環境はCPU Pentium II 233MHz以上、メモリ128Mバイト以上、HDD 150Mバイト以上、Red Hat Linux 6.2。

Software

Javaベースのプレゼンテーションツール Espress Report

発売日 2001年7月30日

発売 株式会社クライム
TEL 0471-67-9345
価格 137万5000円（1ユーザー）

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

クライムは、Javaを活用したレポート作成可能なビジネスプレゼンテーションツール「EspressReport」を7月30日から発売した。

EspressReportはビジネスプレゼンテーションで要求されるグラフィックス、レポート機能をブラウザ上に作成できるソフトウェア。オリジナルデータを自由にカスタマイズして、チャート変換する。テキストデータ、ODBC、JDBC、XMLなどのデータソースからリポートを作成し、HTML、DHTML、PDFなどのフォーマットで直接書き出すことができる。WebLogic、WebSphere、iPlanet、Dynamo、SilverStreamなどのアプリケーションサーバでの使用が可能なうえ、スタンドアロンのアプリケーションとして利用することもできる。

13種の基本チャートから29種類のコンビネーション・チャートを作成できる。

Software

複合ネットワーク対応のディレクトリサービス Novell Account Management 2.1

発売日 2001年7月19日

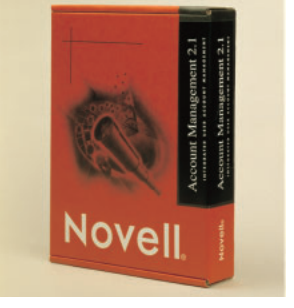
発売 ノベル株式会社
TEL 03-5481-1294
価格 オープンプライス

URL http://www.novell.co.jp/ncc/

ノベルはWindows 2000のサポートを追加したディレクトリ製品の新バージョン「Novell Account Management 2.1」を7月19日より発売した。

Novell Accont Management 2.1は、Windows NT／2000、Solaris、LinuxなどOSごとに管理されているユーザー／グループアカウントをNDS eDirectory 8.5を用いて、ディレクトリに統合し、組織内での異動や出入りにおける複数OSのユーザーアカウント管理を一元化する。複数のOSが混在するネットワーク環境をシームレスなネットワークとして統一することで、企業ネットワークの総合的な運用管理とコスト削減を実現するソフトウェア。

最小ユーザーライセンス250ユーザーからのライセンス販売で、追加ライセンスは1ユーザー単位。対応するOSはWindows 2000 Server／Advanced Sever、Windows NTサーバ4.0 SP 4以上、Solaris 2.6、Solaris 7／8（Sparc Platform Edition）、Linux（カーネル2.2.x、glibc 2.1.3）。



Software

メールによる情報漏えい防止ツール GUARDIAN WALL V3.1 for Linux

発売日 2001年7月2日

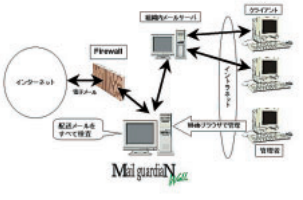
発売 NECソフト株式会社
TEL 03-5569-3399
価格 120万円（50ユーザー）〜

URL http://www.necsoft.com/solution/

NECソフトは、メールの内容を検査し、運用ルールに従っていない情報の外部発信を防止することで、メールによる機密情報漏えいの危険性を低減するソフトウェア「GUARDIAN WALL V3.1 for Linux」を7月2日から発売した。

主な機能は、①万一の事故に備えた発信記録の保存と内容閲覧機能②メール内容を指定した語句の組み合わせやタイプで検査し、機密情報や不適切な情報が外部に送信されるのを防ぐ、コンテンツ検査機能③メールの差出人アドレス、宛先アドレス、メールサイズ等の条件で制御する配送制御機能④メールの処理総数、コンテンツ検査ルールの適用状況、発信・受信状況を示す統計情報閲覧機能など。

V3.1では、ZIP、LHA形式の圧縮ファイルの展開、検査機能を追加。また、メールアドレスでグループ化して、管理をグループ単位で行える。



日刊アスキー Linux Business Report

――産官学民プロジェクトの「リナックスカフェ」

http://www.linux24.com/

7月23日、Linuxベンチャーの集合拠点「リナックスカフェ」の設立発表会が行われた。オープンは12月上旬だという。リナックスカフェは、東京・秋葉原の一角にある下島ビル（DOS/Vパラダイスの向かい。新名称はリナックスビルになる）に位置し、地下1階から地上5階までの全フロアに、インターネットカフェ、スクール、リナックスカーネル研究所、情報交流サロンなどを設置するという。その目的は、「いつでも誰でも気軽に立ち寄ることができ、技術を学ぶことができ、情報を交換したり、注文をしたり、機材を購入したりすることができる」場所を創設することにある。リナックスカフェ参加企業は、共同受注、共同研究、共同開発などを行っていく。

運営主体はインキュベーターのビジネスカフェジャパンをはじめ、リナックスカフェ参加各社が出資したジョイントベンチャー、株式会社リナックスカフェ。全フロアLinuxづくしという、リナックスビルが出現するだけでおもしろい事態だが、その運営形態もなかなかユニークだ。今回は、このリナックスカフェをレポートしたい。

一般ユーザーから企業まで

リナックスカフェは、前述の通り個人向けのインターネットカフェから企業向けのスペースまでがそろっているため、ビジネスの場でもあり、個人レベルでの情報交換の場としても機能する。リナックスカフェにおける具体的な事業内容は以下の5つである。

1. **Linux関連技術の基礎研究**
2. **オープンソース技術コンサルタントおよび技術者の育成と就職斡旋**
3. **Linuxベンチャーの育成**
4. **Linuxのサポートセンター／検証センターの設置**
5. **一般ユーザーがLinuxを体験するためのインターネットカフェ運営**

この5項目を実現するため、フロア構成は**表1**のようになっている。

寄付から始まったリナックスカフェ

リナックスカフェは、他のインキュベーターにはない特徴がある。もちろんLinux（オープンソース含まれる）に特化したということはもちろんだが、たとえば運営団体のひとつに法政大学の名があるように、「産官学民共同のプロジェクト」という性格も持っているのだ。逆に言えば、産官学民のプロジェクトという点が、このリナックスカフェの出発点ともいえそうだ。その設立までの経緯がなかなかおもしろいので紹介しよう。

7月23日の記者発表時に配布された、東京都千代田区街づくり推進公社発行の「千代田区街づくりニュース」によると、まず最初に、故下島美代氏が下島ビルを千代田区に寄付したことから始まったようだ。「公共に役立ててほしい」という下島氏の遺志を受けて、千代田区は同ビルを千代田区街づくり推進公社に無償貸与、さらに同公社は地域活性化の街づくり事業拠点として、ベンチャー企業育成センターとしての利用を企画し公募を行う。公募した企画の要件として、「IT関連ベンチャーの育成」、「地域活性化の街づくり」といった点が盛り込まれていた。公募には7法人からの提案が届き、ビジネスカフェジャパンの提案であるリナックスカフェが採用されたのだという。

	フロア名	概要	運営団体
5階	Developer Cafe	リナックスビルの情報通信環境整備やサポートを行う。リナックスプロフェッショナル協会事務局も入居予定	サイバーテック、ビジネスカフェジャパン
4階	Incubation Cafe	各ベンチャー企業が入居する小部屋の集合。Linuxベンチャーの立ち上げ支援を行う	ビジネスカフェジャパン
3階	College Cafe（リナックスカーネル研究所）	情報家電にフォーカスしたLinuxの教育・研究施設。エンジニアの育成と就職斡旋も行う	ビジネスカフェジャパン、法政大学
2階	Wired Library	交流の場。イベントやセミナーをひらく。ダウンロードコーナー、プレゼンテーションスタジオの3コーナーで構成される多目的ホール。コミュニティスペースも設置し、各ユーザーグループに解放して情報交換の場を提供する	リナックスカフェ
1階	Linux Cafe de Pront	サントリーのカフェレストラン「プロント」がインターネットカフェを運営	プロントコーポレーション
地下1階	ストリーミングスタジオ	スタジオ、映像編集室、ストリーミング配信システムを設置し、遠隔授業のためのコンテンツを作成、配信する	法政大学、ビジネスカフェジャパン

表1　リナックスビルの各フロア

こうした経緯から、リナックスカフェは単なるベンチャー企業育成にとどまらず、地域活性化や経済の活性化モデルといった、もう少し大きな目的も含んでいるのである。ビジネスカフェジャパンが配布した「リナックスカフェ・マニュフェスト」と題された文書では、

①行政、産業界、学会、地域住民間の障壁
②大手企業、中堅企業、ベンチャー企業間の障壁
③国家、民族間にある障壁

の3つの障壁を取り除いていくとしている。①や②の克服の一端としては、千代田区や秋葉原地域商工団体、法政大学の参加などが挙げられるだろう。また、③については、米国、インド、カナダ、韓国の企業がリナックスカフェに参加する基盤が整いつつある。すでにインドのエーセントジャパンや、カナダのビットフラッシュジャパンなどが、リナックスカフェ内のシステムインテグレーションやソフトウェア販売事業に名乗りをあげている。

つまりリナックスカフェは、秋葉原という地域と、Linuxという技術を中心として、あらゆる人たちの交流の場を作ってビジネスを発展させていこうというものらしい。実際にどのような雰囲気になるのかはオープン後の運営体制などにもかかってくると思うが、現在入居予定の顔ぶれを見る限りでは、なかなかおもしろそうだ。ちなみに、リナックスカフェ自体は、家賃やサービス手数料によって収益を得るという。また、ビジネスカフェジャパンは、こうした「カフェ」のスキームを他の分野に応用し、たとえば、XMLカフェやバイオカフェなどを設立する際に今回のノウハウを活かしていくことを考えているという。また、リナックスカフェに関しても、リナックスカフェのブランド力を高め、リナックスカフェとしてのビジネスによるコミッションなどでも収入を得ていく。

千代田区街づくり推進公社のWebページ（http://www2.odn.ne.jp/citystation/）に企画提案書抜粋が掲載されているので、興味のある方はご覧いただきたい。

Interview

株式会社リナックスカフェ
代表取締役社長
林 正博氏

――なぜカフェの題材をLinuxにしたのか？たとえば、もっと広く「インターネット」をターゲットとすることも可能だったのでは？

林氏：Linux（オープンソース）については、ビジネスカフェジャパンとしてはかなり前の段階から視野に入っていた。我々はリナックスカフェでベンチャーの最適チームを作り、大手に営業していく。

我々のビジネスとして、大企業の新規ビジネス、新規プロジェクトをオーガナイズし、そこにベンチャーの技術をマッチングして大きな力にするということがある。日本では、やはり大企業を相手にしなければビジネスは難しい。しかし、ひとつひとつのベンチャーが個別に大企業を相手にするのは困難だ。そこで、技術を持っているベンチャーを集め、大企業のプロジェクトとマッチングする。

――リナックスカーネル研究所の説明を見ると、情報家電分野に注力しているようだが

林氏：サーバ関連でLinuxが伸びていくのは間違いない。クライアントに関してはWindowsだろう。そのあとに何があるかというと、アプライアンス＝組み込みだろう。情報家電が本格化すると、今のITとは桁違いのマーケットになる。そこでLinuxが使われれば、Linux Anywhereということになる。こうした研究の成果が製品化されれば、リナックスカフェで展示して一般にアピールすることも可能だ。

秋葉原には、Linux関連企業もたくさんあるし、もともとラジオ小僧が大勢集まっていた場所だ。技術を持ったベンチャーが集まるという素地はあるわけで、リナックスカフェとしては最適な場所だと思っている。

リナックスカフェ
http://www.linux-cafe.co.jp/

・株主
ビジネスカフェジャパン
サイバーテック
パイナップルカンパニー
ロジック・アンド・マジック
インタラクティブエックス
デジタルデザイン
レーザーファイブ
栄光
ネットワーク研究所
オージス総研
ミクプランニング
デイツーイーツー
（以下予定）
サントリー
エーセントテクノロジー
シーズ

・協賛企業・団体
日本ヒューレットパッカード
コンパックコンピュータ
東日本電信電話
清水建設

・協力・支援企業・団体
法政大学
日本パーソナルコンピュータソフトウェア協会
朝日監査法人（アンダーセン）
ホライズン・デジタル・エンタープライズ
ミラクル・リナックス
リナックスプロフェッショナル協会
合資会社マイカ
札幌BizCafe

Distribution

新着ディストリビューション

HOLON Linux 3.0

タイピングソフトなどパーソナルユーザー向けのソフトで知られるホロンのHOLON Linuxが、カーネル2.4対応のディストリビューションとしてバージョンアップされた。ユーザーニーズに合わせた利用価値の高いアプリケーションをフルインストールで2.8Gバイトと、数多くバンドルしている。通話料無料の電話サポートはLinux入門者には心強い味方だ。

HOLON Linux 3.0

ホロンから、HOLON Linuxの最新版「HOLON Linux 3.0」が7月13日に発売された。HOLON Linux 2.0と同様にPC/AT互換機とPowerPCに対応し、それぞれ「豪華」と「並」の2種類のパッケージの合計4製品で構成される。

Linuxカーネルは、安定重視のカーネル2.2.19と最新機能を取り入れたカーネル2.4.5の2つのバージョンを同梱している。Cライブラリはglibc2.2.3、X Window SystemはXFree86 4.0.3を採用している。なお、XFree86 4.0.3で対応していない古いグラフィックスコントローラのために、XFree86 3.3.6のドライバも含まれている。

標準のデスクトップ環境は、今回からGNOMEではなくKDEが採用された。KDEのバージョンは従来の1.4から2.1.1になり、KOffice（ワープロ、表計算、プレゼンテーション、チャット）やKMail（メーラ）を使えば、デスクトップ用途にも十分利用できる。Konquerorファイルマネージャはファイル表示以外にWebブラウザの機能も持っている（**画面1**）。

もちろんGNOMEも選ぶことができる。1.4.0.4にバージョンアップされ、プレビュー表示が行える高機能ファイルマネージャNautilus 1.0.3も利用可能だ。

パーティションサイズを変更できるインストーラ

HOLON Linux 3.0 for PC/ATのインストーラは、Red HatのAnacondaベースのグラフィカル／テキストインストーラだが独自に拡張している。パーティション設定の「Disk Druid」のなかで、すでに確保されているパーティションのリサイズが行えるようになった。また、テキストベースのpartedツールを選びパーティションサイズを変更することもできる。あらかじめ、デフラグなどでWindowsパーティションのファイル使用領域を整理しておく必要はあるが、1台のハードディスクにWindowsとLinuxをインストールする場合に便利な機能だろう。

インストールCDは2枚で、インストール時にDisc 1から起動すればカーネル2.2.19を、Disc 2から起動すればカーネル2.4.5のシステムをインストールすることができるようになっている。こういった工夫をしたためか、パッケージのインストール時にDisc 1とDisc 2を入れ換える指示が頻繁に現れる。標準的なパッケージを選んだ場合で十数回もCDを交換するというのは改善してほしい点だ。

フロッピーから起動する必要があるため少し面倒だが、WindowsのFAT領域にCD-ROMの内容をコピーしておき、ハードディスクからインストールすることや、LANを経由した別のLinuxマシン上にCD-ROMイメージを用意してFTP／HTTP／NFSによってインストールすることも可能だ。

また、RPMパッケージのインストー

画面1　KDE 2.1.1のデスクトップ画面
ファイルマネージャKonquerorは、URLを指定するとWebブラウザとして表示する。

画面2　アップデートを簡単に行えるSynaptic
Synapticは、パッケージ管理ツールAPT（Advanced Package Tool）をGUIで操作可能にする。

ルを補完して使いやすくするAPTツールと、それをGUIで操作できるSynapticを利用することで、インターネット経由で最新パッケージのダウンロード／アップデートが可能になった（**画面2**）。

豊富なアプリケーション

商用ソフトも豊富だ。かな漢字変換ソフトとして、Wnn6 Ver.3.0.2がすべての製品に含まれているほか、PC/AT用の「豪華」にはATOK12 SE/R2もバンドルしている。X Window System上できれいな日本語表示を行うTrueTypeフォントとして、ダイナフォントが「豪華」には15書体、「並」には5書体用意されている。PC/AT版には日英／英日の翻訳ソフトの「翻訳魂」がバンドルされる。

そのほか、Acrobat Reader、RealPlayer（ストリーミングビデオ再生）、一太郎Ark体験版、choco Technology Preview（表計算）、muffin Technology Preview（プレゼンテーション）といったソフトメーカーが提供するアプリケーションや、ホームページ作成ソフト、デジタルビデオ編集ソフト、CD-R作成ソフト、MP3対応ソフト、ゲームソフトなど多くのソフトウェアが含まれる。すべてのアプリケーションをインストールすると「並」でも3Gバイトを超えるディスク容量が必要なほどだ。

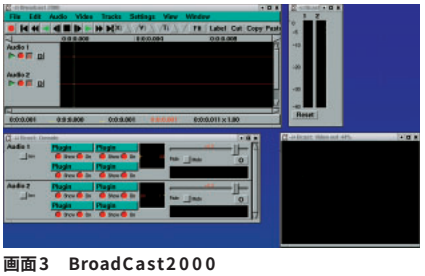
サポート

インストールからX Window Systemの起動など、Linux導入時のトラブル解決に重要なサポートに関して、ユーザー登録後90日間は件数無制限で対応してくれる。「豪華」では通話料無料の電話サポートを平日10時～21時まで受け付けるほか、FAX、メールでのやりとりも可能だ。「並」の場合は、メールによる問い合わせのみだが、追加料金で電話サポートを受けることもできる。

なお、2001年以降に発売されたMacの場合、対応状況を購入前に問い合わせたほうがよいだろう。

付属のマニュアルは、インストールとオペレーションの2冊で構成される。オペレーションマニュアルは約300ページとかなり厚みがあり、基本的な操作からKDEデスクトップの使い方はもちろんのこと、グラフィックツールGimpと3Dレンダリングソフトblenderについて合わせて90ページも割いて解説している。Macユーザーを意識したグラフィックソフト重視な点はHOLON Linuxの特色といえるだろう。

価格が1万円を超すことも多くなった最近のディストリビューション製品のなかで「HOLON Linux 3.0 for PC/AT 並」の3980円というのは、かなりお買い得感がある。きちんとしたマニュアルが付属し、サポートが受けられるので、これからLinuxを始めようという方が試してみるのによいのではないだろうか。



画面3　BroadCast2000
IEEE-1394あるいはVideo4Linuxに対応したデバイスから取り込んだデータを編集できるデジタルビデオ編集ソフト。

画面4　ゲームソフトXBlast-TNT
時限爆弾を通路に置き、爆発させて相手を倒す対戦ゲーム。ネットワーク対戦も可能だ。

製品名　HOLON Linux 3.0
価格　3980円～
問い合わせ先　株式会社ホロン
0070-800-200052
http://www.holonlinux.com/

	for PC/AT 豪華	for PC/AT 並	for PPC 豪華	for PPC 並
ATOK12 SE（かな漢字変換）	○	—	—	—
Wnn6 Ver.3.0（かな漢字変換）	○	○	○	○
ダイナフォント（日本語TrueType）	15書体	5書体	15書体	5書体
翻訳魂	○	○	—	—
OSS商用サウンドドライバ	○	—	○	—
サポート	90日間無制限の電話、FAX、メールサポート	90日間無制限のメールサポート	90日間無制限の電話、FAX、メールサポート	90日間無制限のメールサポート
価格	1万1800円	3980円	9800円	4800円

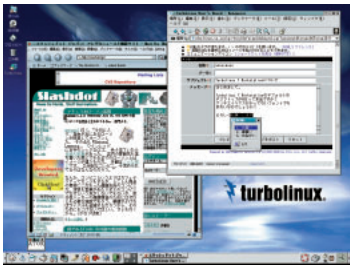
表1　HOLON Linux 3.0の製品構成と商用バンドルソフト

Distribution ▶▶▶

「Turbolinux 7 Workstation」9月7日発売

ターボリナックスジャパンは「Turbolinux 7 Workstation」を9月7日に発売する。今バージョン製品版より、若干名称が変更された。

バージョン6を大幅に刷新する内容となったTurbolinux 7 Workstationは、基本システムとしてカーネル2.4.5、glibc2.2.3、XFree86 4.1.0、KDE 2.1.1、GNOME 1.4.0.4などを採用し、デフォルトのデスクトップ環境がKDEに変わった。また、便利な機能として、新たにパーティションを作ることなくWindows環境にインストールして、フロッピーブートできる「ループバックインストール」が可能になった。製品版のマニュアルがインストール・アプリケーション・コマンド・FAQなど7つの項目で用意されるなど、Windowsユーザーや、初心者にも使いやすいLinuxが意識された構成となっている。



製品版には、かな漢字変換ソフトのATOK XとリコーのフォントDFが5書体搭載され、日本語環境がより充実している。価格は1万5800円。

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

Itanium対応Red Hat Linux 7.1発売開始

5月に発表されていた、Itanium対応「Red Hat Linux 7.1 for the Itanium Processor」が、7月9日に米Red Hatより発売開始された。特徴は、最大8CPUまで対応したマルチプロセッサシステム、2Tバイトまで対応したファイルサイズが挙げられる。また、日本語を含めた多言語にも対応している。

製品価格は499USドル。「Red Hat Network Software Manager」の6カ月間のサポートと30日間のWebサポートが付属する。

Red Hat（http://www.redhat.com/）

「Slackware 8.0」リリース

Slackware Linux Projectは、6月30日にSlackware 7.1の次期バージョンSlackware 8.0をリリースした。Slackware 8.0は、カーネル2.2.19または、カーネル2.4.5と併用し、glibc 2.2.3、XFree86 4.1.0、KDE 2.1.2、GNOME 1.4などを基本システムとして採用している。また、ProFTPD、OpenSSH、OpenSSL、mod_ssl、mod_phpのカーネル2.4対応などが行われている。Slackware 8.0でも、テキストベースのインストーラが採用され、Slackwareを使い続けているユーザーや、UNIX経験の豊富なユーザーを意識している。

製品版には、ブートディスクや完全なソースコードなどがパッケージングされており、Webにて39.95USドルで購入できる。

Slackware Linux Project（http://www.slackware.com/）

Caldera 「OpenLinux」シリーズの新バージョンをリリース

米Caldera Internationalは、6月26日「Caldera OpenLinux Workstation 3.1」と「Caldera OpenLinux Server」をリリースした。

OpenLinux Workstation 3.1は、カーネル2.4ベースのシステムに「Java JDK 1.3」や「JBuilder 4 Foundation」などを搭載し、JavaやC、C++での開発を念頭においたソフトウェア開発者向けの製品となった。基本システムには、KDE 2.1、glibc 2.2.1、XFree86 4.0.2などが採用され、複数台のLinuxマシンを同時に管理する「Caldera Volution」の60日間試用版がバンドルされている。

OpenLinux Serverは、デフォルトの設定でWebサーバ、ファイルサーバ、プリントサーバなどの機能を利用できるなど、中小規模サーバ用に開発されたディストリビューションである。ほかにも、カーネル2.4、MySQL、ドライバやパッケージの自動アップデート機能、ファイルやポートへの不正侵入防止ツールなどが搭載されている。

米Caldera International（http://www.caldera.com/）

「Linux MLD mini 1.0」発売

メディアラボ株式会社が、7月24日に「Linux MLD mini 1.0」を発売した。Linux MLD miniは、FATまたはNTFSへインストールするため、パーティションを切り直す必要がなく、Windows上から簡単にインストールできる。最大の特徴は、機能を最小限に絞ったため、インストールに必要なディスク容量が120Mバイトと小さく、メモリ32Mバイトマシンでも快適に使えるような「軽い、速い」設計となっているところだ。

メディアラボ株式会社（http://www.mlb.co.jp/）

Products

Webホスティング機能に特化した1Uラックマウント・アプライアンスサーバ

hp web hosting server appliance sa1100／1120

インターネットへ情報発信するためのWebサーバの利用価値はますます高くなっている。あらかじめWebサーバに必要な機能をインストール済みのアプライアンスサーバを利用すれば、導入後すぐに運用を開始できる。

製品名	hp web hosting server appliance sa1100/1120
価格	19万9000円（sa1100）／33万9000円（sa1120）
問い合わせ先	日本ヒューレット・パッカード株式会社 TEL 03-5344-7181 http://www.jpn.hp.com/go/serverappliance/

日本ヒューレット・パッカードから、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」シリーズとして、sa1100とsa1120の2モデルが発売されている。外観はどちらも一緒で1Uラックマウントタイプである。

低価格なsa1100は、CPUにCeleron 600MHz、メモリ128Mバイト、20GバイトIDEハードディスクを搭載している。高性能なsa1220は、CPUにPentiumIII750MHz、メモリ256Mバイト、9GバイトSCSIハードディスクを搭載している。必要に応じてハードディスクを増設することでミラーリングによって信頼性を高めることが可能だ。

本機は、フロッピー、CD-ROMドライブといった外部記憶装置やビデオカードが装着されておらず、キーボード／マウス用のPS/2やUSBも利用しない。すべてのコントロールはネットワーク（LAN）経由で行うようになっている。

Webサーバ機能以外に、Webホスティングに必要なメール、FTP、telnet／SSH、DNSなどのサーバ機能も備えている。OSはLinuxで、WebサーバにはApache、メールサーバにはsendmailといった標準的なソフトウェアを採用している。

Webベースの管理ツール

今回は、SCSIインターフェイスを装着したsa1120を試用した。最初に、前面パネルにある小型のLCDディスプレイとその右側に付いている4つの上下左右スイッチを利用して、IPアドレス、ネットマスク、デフォルトゲートウェイといったネットワークの基本設定を行う。あとは、別の

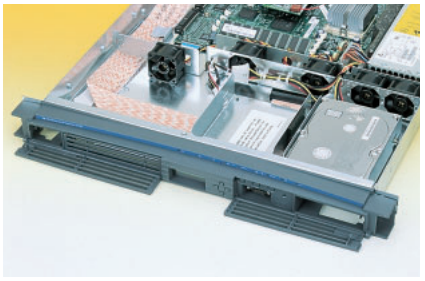


写真1　前面パネルとドライブベイ
前面パネルのセンターにLCDパネルと設定用上下左右スイッチを装着。フロッピー、CD-ROMは装着されていない。

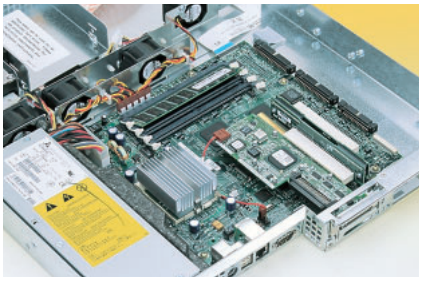
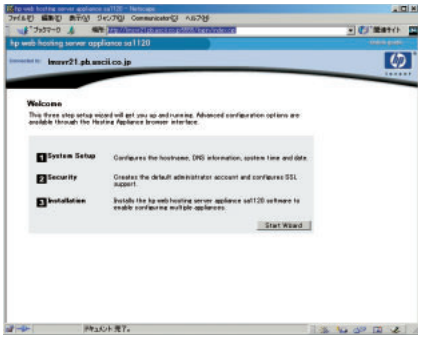
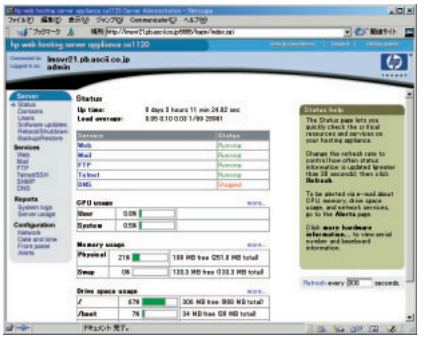


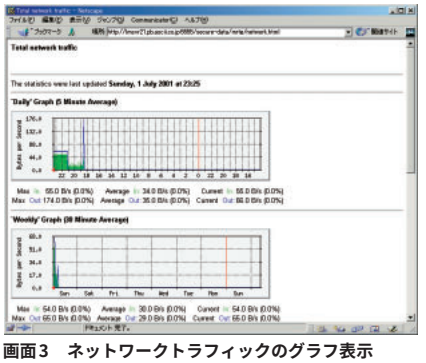
写真2　sa1120のマザーボード
チップセットは440BX。sa1120にはロープロファイルPCIカードのAdaptec 29160LPを装着している。



画面1　初期設定ウィザード
管理ツールはWebブラウザでポート5555に接続する。最初はウィザードでシステム設定を行う。



画面2　管理ツールのステータス表示
各サービスの動作状態、CPUやメモリ、ディスクの使用状況を表示。左側のメニューで各種設定／表示が可能。



画面3　ネットワークトラフィックのグラフ表示
デイリー／ウィークリーのネットワークの使用状況を把握できる。CPUやメモリの状況もグラフ表示可能。

WindowsマシンなどのWebブラウザで、「http://<IPアドレス>:5555」のようにして本機に接続する。「5555」というのは、Webベースの管理ツールが使用するポート番号である。

最初に接続したときは、Webブラウザで初期設定の管理者ユーザー名とパスワードを入力すると、**画面1**のような初期設定ウィザードが表示される。ウィザードにしたがって、ホスト名、ドメイン名、DNSサーバ、時刻（タイムゾーン、NTPサーバの指定）などと、管理者のログイン名、パスワード、管理ツールのSSL利用の詳細を設定すると、設定値に合わせたサーバ構成でシステムがインストールされる。

このあと、利用者のユーザーの登録を行い、FTPでWebコンテンツを転送すれば、Webサーバとして利用できる。管理者としてログインすることで、**画面2**のように画面左側のメニューから、各種サービスの状態表示やサービスの再設定、バックアップ／リストアなどの作業を、すべてWebブラウザから行うことが可能だ。

Reportsのメニューでは、ログやCPU、ネットワーク、メモリのシステム負荷のグラフ表示を見ることができる。**画面3**は、Server usageでネットワークの利用率をデイリー／ウィークリー単位で表示したグラフである。

本機は、複数のホスト名やIPアドレスをサービスする「バーチャルドメイン機能」を備えており、複数のWeb、メール、FTPサービスを1台のサーバで提供することが可能だ。メールサーバの設定では、リレーホストやメールを受信可能なホスト／ドメイン、リジェクトするユーザー／ホスト／ドメインを指定できる。FTPサーバをAnonymous FTPとして動作させることで、ダウンロードファイルをFTPサービスで提供することもできる。

また、本機のOS、アプリケーション、BIOSの最新版が更新用アップデートとして提供され、インターネット経由でアップデート可能になっている。

1Uサイズなので19インチラックに多くのWebサーバを収納可能であり、データセンターなど短時間でシステムの設置・稼働を行う場合に適している。

CPU	Celeron 600MHz（sa1100）／PentiumIII750MHz（sa1120）
チップセット	440BX
RAM	128M／256MバイトECC SDRAM（PC133、最大1Gバイト）
ハードディスク	20GバイトIDE／9GバイトUltra160 SCSI（最大2台）
CD-ROM／フロッピードライブ	なし
グラフィックス	なし
前面パネルLCD	電源、警報、10／100Mビットリンク（2個）、送信／受信（2個） ドライブの動作表示、メール／Webサービス中の表示
ネットワーク	オンボード10/100BASE-T×2ポート（Intel 82559）
SCSIコントローラ	Adaptec 29160LP（sa1120）
インターフェイス	RS-232C×2（前面／背面）
PCIスロット	2スロット（sa1120は1スロットをSCSIコントローラで使用）
キーボード／マウス	なし
サイズ（mm）、重量	428（W）×585（D）×44.5（H）、14.5kg
電源	125W

表1　hp web hosting server appliance sa1100/1120の主な仕様

ET-NAS20G／ET-NAS60G

複数のPCを利用しているとファイル／プリンタを共有できると便利だ。SOHOや小規模オフィスなどでサーバを立てるほどでもないといった場合には、LANで接続するだけで、Windows、Macintoshから利用できるネットワークストレージが導入しやすいだろう。

製品名	ET-NAS20G／ET-NAS60G
価格	9万8000円（ET-NAS20G）／12万円（ET-NAS60G）
問い合わせ先	株式会社アイ・オー・データ機器 TEL 076-260-1024、03-4288-1039 http://www.iodata.co.jp/

パソコン周辺機器を発売しているアイ・オー・データ機器から、ホームサーバ機能を搭載したe@compo ET-NASシリーズが発売されている。内蔵するハードディスク容量の違いによって、「ET-NAS20G」（20Gバイト）と「ET-NAS60G」（60Gバイト）の2種類のモデルが用意されている。

Linux、Samba、Netatalkでファイル共有を実現

ET-NASの本体内部には、小型のマザーボードと電源基板、それと60Gバイトのハードディスクが1台内蔵されている（**写真1**）。CPUには、ナショナルセミコンダクターのx86互換プロセッサであるGeodeが使われている。メモリは32Mバイト。ネットワークコントローラはDAVICOM社のDM9102Fを搭載し、10BASE-TXまたは100BASE-Tで接続が可能だ。

LANポートは1ポートだけなので、複数のマシンを接続するには別にハブが必要である。LANコネクタのすぐ横にあるスイッチで、ネットワーク接続をハブと接続する場合のストレートと、直接PCと接続する場合のクロスとを切り替えることができるようになっている。

OSにはLinuxを採用しており、Microsoftネットワーク（SMB）をサポートするSambaとAppleShare（AppleTalk）をサポートするNetatalkを使用して、Windows／Macとのファイル共有やプリントサーバ機能を実現している。

インターネット接続の共有も可能

今回は、ET-NAS60Gを試用した。本機は出荷時に、192.168.0.2というIPアドレスに設定されている。まず最初は、PCと本機を直接接続し、PCのIPアドレスを192.168.0.3のように同一セグメントに設定する。そしてPCのWebブラウザから、http://192.168.0.2/にアクセスすると設定画面が表示される。画面上の［管理者用ログイン］ボタンをクリックするとパスワードダイアログが現れるので、adminユーザーでログインする。

サーバの名前やシステムのバージョンなど現在のサーバ情報のウィンドウが表示され、［アドミニストレーション］ボタンを押すと、管理者情報の登録画面になり、会社名やユーザー名、メールアドレスなどの入力を行う。これが済むと**画面1**のような本機の各種設定を行うメインの設定画面になる。

ここで［ネットワーク］を選び、IPアドレス／サブネットマスクなどを設定する。次にWindows共有のた

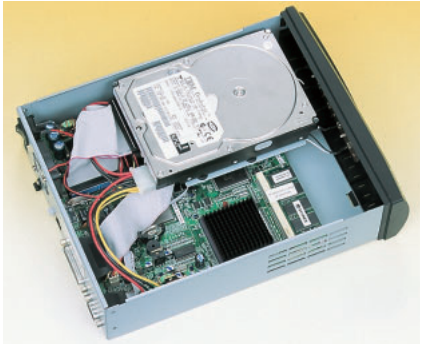


写真1　ET-NAS60Gの内部
ハードディスクドライブは、IBMのDeskstar 60GXP（61.5Gバイト、7200rpm）を使用している。

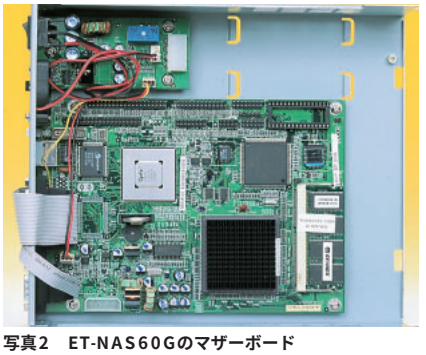
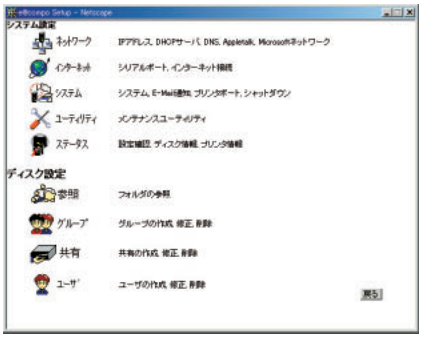
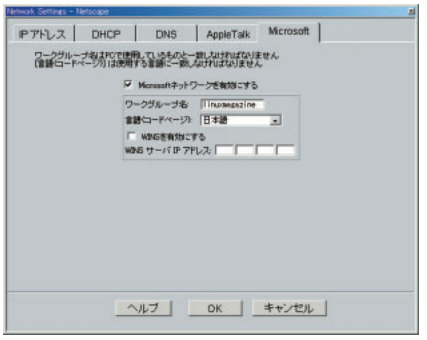


写真2　ET-NAS60Gのマザーボード
ヒートシンクでカバーされているのがCPUでNS社のx86互換Geodeを使用。次に大きいシルバーのはCX5530チップセット。

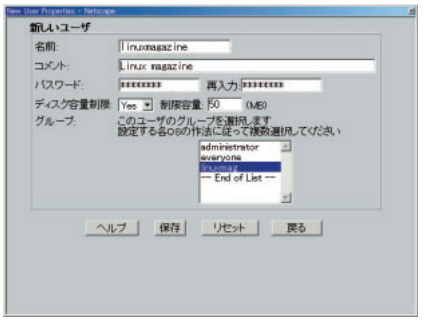
写真3　前面パネルと背面パネル
前面のランプで稼働状態を表示。背面には、LAN、プリンタ、モデム／TA用シリアルポートが用意されている。



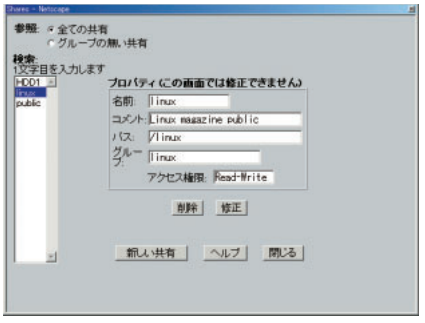
画面1　メインの設定画面
管理者用ログインを行って表示される設定画面。このメニューから各設定用のウィンドウが表示される。



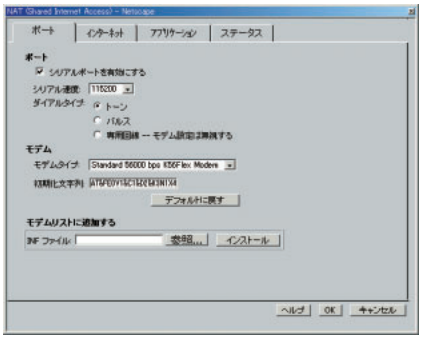
画面2　ネットワーク設定のMicrosoftタブ
Windowsネットワークのワークグループ名を入力。別のタブでIPアドレス、DHCP、DNS、Mac用のAppleTalkを設定する。



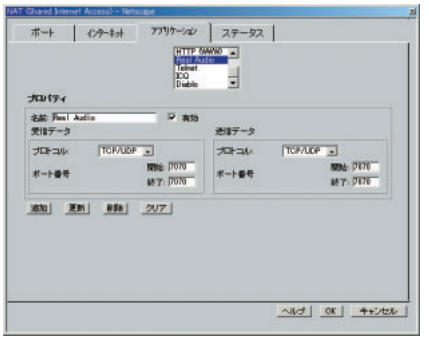
画面3　新しいユーザーの作成
名前は、Windowsのユーザー名と同一にするとよい。使用ディスク容量の制限も設定できる。



画面4　ファイル共有の設定画面
HDD1はルート（/）を表し、管理者のみアクセス可能。publicはすべてのユーザーが読み書きできる。



画面5　インターネット設定のポートタブ
ダイアルタイプ、モデムを指定する。Windows用INFファイルを利用してモデムタイプの追加が可能。



画面6　インターネット設定のアプリケーションタブ
NAT（ネットワークアドレス変換）で透過させるアプリケーションのポート番号を設定する。

めにMicrosoftタブでワークグループ名を設定し（**画面2**）、Mac用にはAppleTalkタブでゾーン名を指定する。本機のDHCPサーバ機能を使うことで、PC側の設定を「IPアドレスを自動的に取得する」で利用することが可能だ。

そしてユーザーの追加を行い（**画面3**）、PCのネットワークを正しく設定すれば、ET-NASの共有フォルダ（あらかじめpublicが用意されている）へアクセスできるようになる。必要に応じてグループを作成したり、新しい共有フォルダを作成すればよい（**画面4**）。

インターネット共有を行うには、モデムまたはTAをシリアルポートに接続する。使用する機器がモデムタイプに登録されていない場合、そのモデムのWindows 98/95用のINFファイルを使ってモデムリストに追加することが可能だ（**画面5**）。

各PCのデフォルトゲートウェイに本機のIPアドレスを設定することで、本機を経由してインターネットにアクセスすることになる。アプリケーションタブでは、インターネット経由で通信するプロトコルを指定することが可能だ（**画面6**）。

現在の仕様では、WindowsとMacの混在環境でフォルダ名に日本語を使用すると、異なるOSから参照したときに文字化けしてしまうため、英数字を使わなくてはいけないというのは残念な点だ。

昨今、個々のPCのディスク容量が増えたためバックアップが大変になってきた。重要なデータの保存用や再インストール時の一時ファイル置き場などとしてリーズナブルなネットワークストレージではないだろうか。

CPU	NS Geode 200MHz
RAM	32MバイトSDRAM（SO-DIMM）
チップセット	CX5530、Super I/O
ハードディスク	20Gバイト（ET-NAS20G）／60Gバイト（ET-NAS60G）
CD-ROM／フロッピードライブ	なし
グラフィックス	なし
ネットワーク	10/100BASE-T×1ポート
インターフェイス	パラレル×1、RS-232C×1
LED	Ready、Error、Disk、Disk Full、LANの表示
添付品	LANストレートケーブル（2m）、ACアダプタ、縦置きスタンド（2個）
サイズ（mm）、重量	210（W）×271（D）×66（H）、約2.1kg
電源	ACアダプタより供給（最大24W）
対応OS	Windows 2000／Me／98／95／NT4.0、Mac OS 8.0〜9.1

表1　ET-NAS20G／ET-NAS60Gの主な仕様

コマンドマスター
呪文使いへの道

Linuxコマンド攻略ガイド

/bin、/usr/bin、/sbin、/usr/sbin、bashの組み込みコマンドの謎が今、解き明かされる。コマンドマスターへの扉を開く、呪文の書ついに公開！

illustration : Chata Tachibana

コマンドラインを使おう

Linuxを使ううえで、コマンドラインは切っても切り離せない存在だ。Windowsユーザーであれば、コマンドラインはうっとおしい存在で、使いにくく、どうしてわざわざコマンドラインなんかを使うのだろうと思うかもしれないが、コマンドラインこそコンピュータを便利に使う魔法の呪文なのだ。

確かに、GUIは直感的でわかりやすい。起動したいプログラムの指定も簡単だし、ドラッグ&ドロップやカット&ペーストも魅力的だ。しかし、それだけ人間が細かく指示を出してやらなければならないのも事実だ。

しかし、コマンドラインはあらかじめ指定した動作を、どんなに複雑でも自動的に実行してくれる。それも忠実にだ。ドラッグ先を間違えてしまうこともなければ、ファイルを間違えることもない。あらかじめ決められた処理をきっちりこなしてくれる頼もしい存在だ。言い換えれば、コマンドラインは、コンピュータをコンピュータらしく使うための道具なのだ。

また、コマンドはひとつひとつは小さく、機能も限定されているが、多くのコマンドを組み合わせて使うことによって、強力なツールとして機能させることができる。

さらに、コマンドをスクリプトとして用意しておけば、定型的な処理、大量の処理を自動化することも簡単だ。こういったコマンドの自由度は、GUIとは比べ物にならない優位性だ。

コマンドラインを活用すれば、より便利に、より快適にコンピュータを使うことができるはずだ。今までコマンドラインを避けていたLinuxer諸氏も、ぜひ、今回の特集でコマンドマスターになってほしい。

コマンドオプション

多くのコマンドには、コマンドの動作を変更するためのオプションが用意されている。オプションは -（ハイフン）を付けるものや、--（ハイフンを2つ）付けるもの、単にAやBを指定するもの、文字列を指定するものとさまざまだが、パラメータの指定のしかたともども、おおむね一定の標準に従っている。指定できるオプションは各コマンドの項を参照してほしい。なお、コマンドによっては指定できるオプションが非常に多いものもあるが、この特集では、主に使われるもの、重要なものを厳選して紹介している。

コマンドのより詳しいオプションの指定方法などは、manページを参照してほしい。

コマンドの位置

ほとんどのコマンドは、/bin、/usr/bin、/sbin、/usr/sbinの4つのディレクトリに集約されている。/binは、UNIXのコマンドとして昔から使われていたコマンドが収録されているディレクトリだ。/usr/binも基本コマンドが収録されているディレクトリだが、現在のLinuxでは何でもアリ状態になっていて、GNOMEやKDEのプログラムまでも含まれ、Red Hat Linux 7.1ではなんとファイル数2700を超す巨大ディレクトリとなってしまっている。

このほか、/usr/local/binという管理者が独自にインストールしたり、作成したコマンドを収録するディレクトリもあるが、前述したようにほとんどのコマンド、特にRPMでインストールしたコマンド類は/usr/binに置かれるため、こちらのディレクトリにはコマンドがないことが多い。/sbinディレクトリは、本来、システムがブートする際に必要となるコマンド類のみが収録されているディレクトリであったが、Linuxでは/sbinと/usr/sbinの区別はあまりなくなっている。いずれにせよシステム管理系のコマンドが、これら2つのディレクトリに収録されている。

また、コマンドとしてでなく、シェルの内部に用意された組み込みシェルコマンドというものもある。これは、どちらかというと、Linuxを対話的に使う場合に必要となるコマンドだ。

今回はこの膨大なコマンド群の中から140以上のコマンドを選んで解説する。全てを覚える必要はもちろんないが、一通り読んでおけば何かの時に役に立つはずだ。

また、ここで紹介したコマンドのほとんどは、主要なものばかりなので、LinuxだけでなくBSDやSunOSなど、UNIX系システムで共通のコマンドだ。ほかのUNIX系システムを使う場合でも必ず役に立つはずだ。

掲載コマンドインデックス

この特集で紹介しているコマンド一覧だ。各コマンドと、収録されているディレクトリ、簡単な説明、参照ページを掲載した。コマンドの詳しい説明は本文を参照してほしい。

コマンド	ディレクトリ	内容	ページ
adduser	/usr/sbin	新規ユーザーアカウントの作成	71
alias	bash	別名の定義	74
arp	/sbin	ARPキャッシュの表示と操作	64
at	/usr/bin	指定した日付／時刻にコマンドを実行する	55
bg	bash	バックグラウンドでの実行	73
bunzip2	/usr/bin	ファイルの圧縮を行う	50
bzcat	/usr/bin	ファイルの伸張を行う	50
bzip2	/usr/bin	ファイルの圧縮／伸張を行う	50
cal	/usr/bin	カレンダーを表示する	62
cat	/bin	複数のファイルを連結する	40
cd	bash	カレントディレクトリの変更	72
chattr	/usr/bin	ext2ファイルシステム上にあるファイルの属性を変更する	47
chgrp	/bin	ファイルの所有者グループを変更する	52
chkconfig	/sbin	起動時のサービス（デーモン）の設定と表示	64
chmod	/bin	ファイルのアクセス権を変更する	52
chown	/bin	ファイルの所有者を変更する	51
clock	/sbin	CMOSクロック／システムクロックの設定	65
compress	/usr/bin	ファイルの圧縮を行う	50
cp	/bin	ファイルやディレクトリをコピーする	43
cpio	/bin	アーカイブファイルからファイルの入出力を行う	49
crontab	/usr/bin	各ユーザー用のcrontabファイルを管理する	55
date	/bin	システムの日付と時刻を、表示または設定する	61
dd	/bin	ファイルを変換してからコピーする	44
depmod	/sbin	モジュールの操作	65
df	/bin	ファイルシステムの使用状況を表示する	44
diff	/usr/bin	ファイル間の差を表示する	48
diff3	/usr/bin	ファイル間の差を表示する	48
dig	/usr/bin	ドメイン名に関する問い合わせをネームサーバに送る	58
dirs	bash	保存中カレントディレクトリの表示	72
dmesg	/bin	カーネルが出力するメッセージを表示する	57
du	/usr/bin	ディスクの使用量を表示する。	47
dump	/sbin	ファイルシステムのバックアップ	66
echo	bash	標準出力への出力	74
egrep	/bin	ファイルから文字列を検索する	42
eject	/usr/bin	リムーバブルメディアを取り出す	46
export	bash	環境変数の設定	73
fdformat	/usr/bin	フロッピーディスクを物理フォーマットする	48
fdisk	/sbin	パーティションテーブル操作	67
fg	bash	フォアグランドでの実行	73
fgrep	/bin	ファイルから文字列を検索する	42
file	/usr/bin	ファイルの種類を推測して表示する	48
find	/usr/bin	ディレクトリ階層下にあるファイルを検索する	50

コマンド	ディレクトリ	内容	ページ
finger	/usr/bin	ユーザー情報を調べる	55
free	/usr/bin	メモリの利用状況を表示する	55
fsck	/sbin	ファイルシステムのチェックと修復	67
fuser	/sbin	ファイルやソケットを使用しているプロセスの表示	67
grep	/bin	ファイルから文字列を検索する	42
groupadd	/usr/sbin	新規グループの作成	71
groupdel	/usr/sbin	グループの削除	71
gunzip	/bin	ファイルの伸張を行う	50
gzip	/bin	ファイルの圧縮／伸張を行う	50
halt	/sbin	システムの停止	68
hdparm	/sbin	ハードディスクパラメータの設定	68
head	/usr/bin	指定ファイルの最初の部分を表示する	40
help	bash	bash組み込みコマンドのヘルプ表示	74
history	bash	コマンド履歴の表示	72
hwclock	/sbin	CMOSクロック／システムクロックの設定	65
ifconfig	/sbin	ネットワークインターフェイスの設定	69
info	/usr/bin	infoフォーマットのファイルを表示する	60
init	/sbin	ランレベルの変更	69
insmod	/sbin	モジュールの組み込み	65
jobs	bash	ジョブリストの表示	73
kill	bash	プロセスの終了	73
killall	/usr/bin	名前で指定したプロセスにシグナルを送る	54
kon	/usr/bin	日本語を表示できるコンソールエミュレータ	63
last	/usr/bin	最近ログインしたユーザーのリストを表示する	56
lastlog	/usr/bin	ユーザーの最終ログイン時刻などを表示する	56
ldconfig	/sbin	実行時リンクの結合関係の作成	70
less	/usr/bin	多機能なテキストビューア（ページャ）	42
ln	/bin	ファイルへのリンクを作成する	44
locale	/usr/bin	locale情報を表示する	61
locate	/usr/bin	ファイル名データベースでファイルを探す	59
ls	/bin	ファイルのリストを表示する	45
lsmod	/sbin	モジュールの操作	65
man	/usr/bin	オンラインマニュアルを参照する	60
md5sum	/usr/bin	MD5メッセージダイジェストを計算／チェックする	63
mkdir	/bin	ディレクトリを作成する	45
mkfs	/sbin	ファイルシステムの構築	70
modprobe	/sbin	モジュールの操作	65
more	/bin	テキストファイルを画面上で閲覧する	42
mount	/bin	ファイルシステムをマウントする	46
mv	/bin	ファイルやディレクトリを移動する	43
netstat	/bin	ネットワークに関するさまざまな情報を表示する	58
nice	/bin	優先度を設定してコマンドを実行する	54
nkf	/usr/bin	漢字コードを変換する	42
nslookup	/usr/bin	DNSによる名前解決を行う	58
passwd	/usr/bin	ユーザーパスワードを変更する	61
patch	/usr/bin	差分ファイルをオリジナルファイルに適用する	49
ping	/bin	ICMPのECHO_REQUESTパケットをネットワーク上のホストに送る	58
popd	bash	カレントディレクトリの復旧	72
printenv	/usr/bin	環境変数を表示する	61
procinfo	/usr/bin	システムの状態を/procより収集し表示する	63

コマンド	ディレクトリ	内容	ページ
ps	/bin	動作中のプロセスをリスト表示する	53
pushd	bash	カレントディレクトリの保存	72
pwd	bash	カレントディレクトリの表示	74
reboot	/sbin	システムの再起動	68
rename	/usr/bin	ファイル名を変更する	43
renice	/usr/bin	実行中のプロセスの優先順位を変更する	54
restore	/sbin	ファイルシステムのリストア	66
rm	/bin	ファイルやディレクトリを削除する	44
rmdir	/bin	ディレクトリを削除する	45
rmmod	/sbin	モジュールの削除	65
rpm	/bin	RPMパッケージを利用する	61
runlevel	/sbin	ランレベルの表示	70
script	/usr/bin	端末に表示した内容をファイルに保存する	63
set	bash	シェル変数の操作	73
shutdown	/sbin	システムの停止／再起動	68
slocate	/usr/bin	ファイル名データベースでファイルを探す	59
sort	/bin	テキストファイルをソートする	40
split	/usr/bin	ファイルを分割する	41
stat	/usr/bin	ファイルやファイルシステムの状態を表示する	52
su	/bin	他のユーザーのユーザーIDを使用してシェルを起動する。	60
sync	/bin	ディスク上のデータとキャッシュ／バッファの同期	47
tail	/usr/bin	指定ファイルの末尾部分を表示する	40
tailf	/usr/bin	指定ファイルの末尾部分を表示する	40
tar	/bin	アーカイブファイルを扱う	49
tee	/usr/bin	標準出力から読みとり、標準出力とファイルに書き出す	41
telinit	/sbin	ランレベルの変更	69
top	/usr/bin	プロセスをCPU利用時間順などで表示する	53
touch	/bin	ファイルのアクセス時刻／修正時刻を設定する	47
tr	/usr/bin	文字を変換／消去する	42
traceroute	/sbin	ホストへのネットワーク経路の表示	70
tree	/usr/bin	ディレクトリ階層下をツリー表示する	51
umount	/bin	ファイルシステムをアンマウント（マウント解除）する	46
unalias	bash	別名の削除	74
uname	/bin	システムに関する情報を出力する	62
uncompress	/usr/bin	ファイルの解凍を行う	50
uniq	/usr/bin	ソートされたファイルの重複行を削除する	41
unset	bash	シェル変数の操作	73
uptime	/usr/bin	システムの連続稼働時間を表示する	54
useradd	/usr/sbin	新規ユーザーアカウントの作成	71
userdel	/usr/sbin	ユーザーアカウントの削除	71
vmstat	/usr/bin	仮想メモリの使用状況を表示する	55
w	/usr/bin	ログインしているユーザーと実行中のプロセスを表示する	57
wc	/usr/bin	ファイル中の行数、単語数、バイト数を表示する	41
whatis	/usr/bin	whatisデータベースを検索し、コマンドの1行説明を表示する	60
which	/usr/bin	コマンドのフルパスを表示する	59
who	/usr/bin	ログインしているユーザーを表示する	56
whoami	/usr/bin	現在有効なユーザーIDを表示する	57
whois	/usr/bin	ドメイン名などをwhoisサーバに問い合わせる	58
xargs	/usr/bin	標準入力からコマンドラインを作成し、それを実行する	51
zcat	/bin	ファイルの伸張を行う	50

一般的なコマンド

Red Hat Linux 7.1の/bin、/usr/binには、あわせて約3000個近いコマンドが含まれている。コマンド使いになるためには、これらをすべて覚えなければならない…… わけではない。まずは、ここで紹介するコマンドから使いこなしてみよう。

cat /bin

複数のファイルを連結する

catは、複数のファイルを連結して、結果を標準出力に書き出すコマンドだ。「連結する」という意味の「conCATenate」からとった名前であると言われている。かなり強引なネーミングだ。

つなげたいファイルはコマンドラインから指定するが、このとき、ファイルを1つだけ指定すれば、そのファイルの内容がそのまま標準出力（通常はktermなどのターミナル）に書き出される。通常は、このような使い方、つまり指定したファイルの内容を表示するために用いられることがほとんどだ。サイズが大きくてターミナルの画面に表示し切れないファイルを表示する際には、別項で紹介するmoreやlessといったコマンドを用いる。

いくつかオプションが用意されているが、主に利用するのは行頭に行番号を付加する-nオプションくらいであろう。

head /usr/bin

指定ファイルの最初の部分を表示する

オプションなしでheadコマンドを実行すると、指定したファイルの最初の10行を表示する。-nオプションを使えば、任意の行数だけ表示することもできる。たとえば、ファイルの先頭から20行を表示するなら次のようにする。

```
$ head -n 20 hoge.txt
```

このほか、指定したバイト数の分だけ表示することが可能だ。最初の100バイトを表示するなら次のようにする。

```
$ head -c 100 hoge.txt
```

tail/tailf /usr/bin

指定ファイルの末尾部分を表示する

headコマンドがファイルの最初の部分を表示するのに対し、tailコマンドは末尾の部分を表示する。-nオプションで表示する行数を、-cオプションで表示するバイト数を指定できるのも同じだ。

-fオプションは、ファイルの内容が追加されていくことを前提に、永遠にループをし続ける。このオプションを使ってログファイルを表示すると、リアルタイムにログの更新状況を監視できる。しかし、Linuxにはこの目的に特化したtailfコマンドが用意されている。tailfコマンドは、目的のファイルに追加書き込みが起きたときだけアクセスするので、tail -fよりも効率がよい。

sort /bin

テキストファイルをソートする

sortコマンドは、テキストファイル（複数指定も可能）を行単位でソートするコマンドだ。結果は、標準出力に書き出される。通常は（文字コードが）小さい順にソートされるが、-rオプションを付けると大きい順にソートされる。また、-nオプションを付けると、先頭の文字列を数字とみなし

```
$ cat test.txt
3
 2
  4
   1
$ sort test.txt
   1
  4
 2
3
$ sort -n test.txt
   1
 2
3
  4
```

画面1　sortコマンドの使用例

て、その値でソートされる。**画面1**のように、-nオプションの有無で結果が異なることがわかるだろう。

uniq /usr/bin
ソートされたファイルの重複行を削除する

uniqコマンドは同じ内容の行を削除し、1行にまとめる。ただし、比較は隣接する前の行としか行わないので、データをソートしておく必要がある。そのため、sortコマンドと組み合わせて使うことが多い。

```
$ sort datafile.txt | uniq
```

オプションなしで実行すると、重複行が1行にまとめられる。重複していない行のみを出力する場合は、-uオプションを、反対に重複行のみを出力させるなら-dオプションを指定する。このほか、-cオプションを使うとそれぞれの行が何回現れたかをその内容とともに表示する。投票の集計や単語の出現頻度を調べるときなどに便利だ（**画面2**）。

split /usr/bin
ファイルを分割する

大きなファイルを複数の小さなファイルに分けるときに使用する。オプションなしでファイル名だけを指定すると、1ファイルあたり1000行に分割し、それぞれの分割ファイルにはxaa、xab、……という名前が付けられる。

分割する大きさは、オプションで指定できる。バイト指定は-b、行数指定は-l、指定バイト以下でできるだけ多くの行を収めるときは-Cオプションを使う。-b、-Cオプションで指定するバイト数には、倍数接尾子を付けられる。使えるのは、b（512倍）、k（1024倍）、m（1048576倍）の3つ。

また、ファイル名のあとに分割ファイルのプリフィックスを指定することもできる（**画面3**）。

```
$ sort touhyou.txt | uniq -c
      7 佐藤
      5 山田
      5 斉藤
```

画面2　1行に1人ずつの名前を書いた投票データを集計

wc /usr/bin
ファイル中の行数、単語数、バイト数を表示する

テキストファイルを指定してwcコマンドを実行すると次のような出力が得られる。

```
$ wc sample.txt
     37      42    6146 sample.txt
```

出力内容は、左から行数、空白で区切られた単語数、バイト数、指定したファイルの名前を示す。この例で用いたファイルは日本語のテキストなのだが、英語などと違い日本語では分かち書きをしないために単語数が少なく見える。

tee /usr/bin
標準出力から読みとり、標準出力とファイルに書き出す

コマンドの出力をファイルに保存する場合などには、リダイレクトを使う。たとえば、lsの結果をファイルに書き込む際には次のようにする。

```
$ ls -la > ls-la.txt
```

このように標準出力をファイルにリダイレクトすると、画面には表示されない。teeコマンドは、標準入力から受け取った内容を画面に表示させながら、ファイルにも保存する。

```
$ ls -la | tee ls-la.txt
```

```
$ split -C 1440k linux-2.4.6.tar.gz kernel.
$ ls -s
合計 52736
 1444 kernel.aa   1444 kernel.af   1444 kernel.ak   1444 kernel.ap
 1444 kernel.ab   1444 kernel.ag   1444 kernel.al   1444 kernel.aq
 1444 kernel.ac   1444 kernel.ah   1444 kernel.am   1444 kernel.ar
 1444 kernel.ad   1444 kernel.ai   1444 kernel.an    400 kernel.as
 1444 kernel.ae   1444 kernel.aj   1444 kernel.ao  26344 linux-2.4.6.tar.gz
```

画面3　大きなファイルをフロッピー1枚分ずつに分ける

-aオプションを付けると、保存するファイルを上書きせずに、追加書き込みする。

tr /usr/bin

文字を変換／消去する

trコマンドは、標準入力から受け取ったテキストに対して文字の変換を行う。基本的な書式は、

```
tr 文字列1 文字列2
```

という形になる。“文字列1”にある文字は“文字列2”の対応する文字に変換される。たとえば、

```
$ tr abc ABC < alpha.txt
```

とすると、alpha.txt中の文字aをAに、bをBに、そしてcをCに変換する。変換する文字列は範囲指定もできるので、テキスト中の小文字をすべて大文字に変換するのも簡単だ。

```
$ tr a-z A-Z < alpha.txt
```

また、-dオプションを付けると指定した文字を消去する。次のようにすると、小文字がすべて削除される。

```
$ tr -d a-z < alpha.txt
```

more /bin

テキストファイルを画面上で閲覧する

1画面に収まりきらないようなテキストファイルを、画面上で閲覧する際には、moreのような「ページャ」と呼ばれるプログラムを利用する。

初期のmoreコマンドは、1画面分のテキストを表示したところで入力待ちになり、スペースキーを押すことで、次の1画面を表示する単純なものだった。だが最新のmoreは、前の画面に戻ることも可能になるなど、大幅に機能拡張されている。主なコマンドを**表1**にまとめた。

スペース	次の画面に進む
リターン	1行進む
q	終了する
b	前の画面に戻る
/パターン	パターンにマッチする文字列を検索する
:n	次のファイルに移動する
:p	前のファイルに移動する
.	前回のコマンドを繰り返す

表1　moreでよく使われるコマンド

less /usr/bin

多機能なテキストビューア（ページャ）

ページャプログラムではmoreが有名だったが、さらに多機能なページャとして作られたのがlessだ。スペースバーで1画面進み、Enterキーで1行進む。1画面のバックスクロールにはBキーを使う。操作体系はviエディタに準拠しており、各種のコマンドが利用できる。また、gzipで圧縮されたテキストファイルを直接表示できるのもGNU版lessの特徴だ。詳しい操作方法は、Hキーを押して表示されるヘルプを参照してほしい。

nkf /usr/bin

漢字コードを変換する

nkfは、Network Kanji Filterの略で、漢字コードがJIS、EUC、Shift-JISで書かれたテキストを、指定した漢字コードに変換する。-jオプションでJIS、-eでEUC、-sでShift-JISにそれぞれ変換する。オプションを指定しないとJISに変換される。たとえば、SJIS.txtをEUCコードに変換し、EUC.txtに出力するには次のようにする。

```
$ nkf -e SJIS.txt > EUC.txt
```

入力されるテキストの漢字コードを自動判別するが、うまく判別できないときは、-J（JIS）、-E（EUC）、-S（Shift-JIS）オプションで指定するとうまくいくかもしれない。

grep/egrep/fgrep /bin

ファイルから文字列を検索する

grepは、「正規表現」と呼ばれる方法で指定した文字列をファイルから抽出するコマンドだ。この奇妙な名前は、「Global Regular Expression Print」を表している。

grepには、多くのバリエーションが存在する。egrepは、拡張された正規表現が利用できる。またfgrepは、正規表現ではなく固定された文字列を検索するため、より高速だ。も

っとも、最近のマシンは高性能であり、grepでも十分に高速に動作する。

正規表現のごく一部の記法を**表2**に、grepの主なオプションを**表3**に示す。これらを組み合わせて、さまざまなパターンを検索することが可能になる。

```
$ grep '^[c-e]' /etc/passwd
daemon:x:2:2:daemon:/sbin:
emanon:x:500:500::/home/emanon:/bin/bash
champion:x:503:503::/home/champion:/bin/bash
docuso:x:501:501::/home/docuso:/bin/bash
```

上記の例では/etc/passwdファイルから、「ユーザー名がc、d、eで始まるユーザー」を抽出している。

cp /bin

ファイルやディレクトリをコピーする

cpコマンドは、ファイルやディレクトリのコピーを行う(**表4**)。

$ cp コピー元ファイル... コピー先ディレクトリ

コマンドライン上で複数のファイルやディレクトリを指定すると、最後の1つがコピー先ディレクトリとみなされる。

mv /bin

ファイルやディレクトリを移動する

mvコマンドはファイルやディレクトリを移動する。また、ファイル名の変更にも用いられる。ファイルの移動は、ディレクトリ名も含めたファイル名の変更ともみなせるので、広い意味ではファイル名を変更するコマンドと考えてもいいだろう。

$ mv 移動元ファイル... 移動先ディレクトリ

コマンドライン上で複数のファイルやディレクトリを指定すると、最後の1つが移動先ディレクトリとみなされる。

$ mv dir1 dir2 (dir1はディレクトリ)

このように指定した場合は、名前の変更とディレクトリの移動のどちらが行われるだろうか？　dir2が存在しなければ、ディレクトリdir1の名前がdir2に変更される。一方、ディレクトリdir2があるならば、dir1がdir2の下に移動する。

rename /usr/bin

ファイル名を変更する

UNIX系のOSでは、mvコマンドでファイル名を変更するのが一般的だ。しかし、ファイル名の一部を一括して変更する場合など、mvコマンドでは簡単にできないこともある。Linuxには、ファイル名の一部を変更するrenameコマンドが用意されており、威力を発揮する。コマンドの書式は、

rename 変更前 変更後 対象ファイル

となっており、これは、“対象ファイル”で指定したファイルに対し、ファイル名の“変更前”の部分を“変更後”に置き換えるという意味になる。たとえば、“.htm”が付いたファイルがたくさんあり、これらをすべて“.html”に変更したいときは次のようにする。

```
$ rename .htm .html *.htm
```

これで、＊.htmに合致するファイルに対して、ファイル名に含まれる.htmを.htmlに置き換える。

.（ピリオド）	任意の1文字にマッチ
＊	直前の項目の0回以上の繰り返しにマッチ
[123]	1、2、3のいずれかにマッチ
[a-z]	a、b、…zのいずれかにマッチ
^	行頭にマッチ
$	行末にマッチ

表2　正規表現の一部

-c	マッチした行数だけを出力する
-i	アルファベットの大文字と小文字を区別しない
-n	出力される各行の行頭に行番号を付ける
-r	ディレクトリ内のファイルを再帰的に検索する
-w	完全な語だけにマッチする

表3　grepでよく使われるオプション

-d	リンクを保存する
-i	上書きする前に確認する
-p	可能なかぎりファイル属性を保存する
-R	ディレクトリを再帰的にコピーする
-v	実行内容を表示する

表4　cpでよく使われるオプション

このほかにも、a1、a2、a3、……、a149、a150というファイルがあり、これをa001、a002、a003、……、a149、a150に変更したいというときにも有効だ。

```
$ rename a a0 a?
$ rename a a0 a??
```

1回目の実行では、a1～a9がa01～a09に変更される。そして、2回目で、a01～a99がa001～a099に変更され、目的は達成される。

rm /bin

ファイルやディレクトリを削除する

rmコマンドは、ファイルやディレクトリを削除する（**表5**）。-rオプションを指定してディレクトリを削除する際は、中にファイルがあっても実行されるので、注意が必要だ。

dd /bin

ファイルを変換してからコピーする

ddコマンドは、ファイルを指定したブロックサイズずつコピーする。その際に、データの変換も行える。

デフォルトでは、標準入力から標準出力へのコピーが行われるが、それぞれ「if＝」「of＝」によりファイルを指定することができる。たとえば、フロッピーディスク用のイメージファイルをフロッピーディスクにコピーする際には、以下のように指定する。

```
$ dd if=boot.img of=/dev/fd0
```

デバイスファイルを指定することで、通常のファイルシステムではアクセスできない領域も含めた、ディスク全体に書き込むことができる。

ファイル変換については、「conv＝」以下に指定する。主なものを**表6**に示す。

-f	存在しないファイルを指定しても、エラーを返さない
-i	1つずつファイル削除の確認を求める。「y」「Y」を押さなければ削除しない
-r	ディレクトリの内容を再帰的に削除する

表5　rmでよく使われるオプション

ascii	EBCDICからASCIIに変換
ebcdic	ASCIIからEBCDICに変換
ibm	ASCIIからIBM型EBCDICに変換
lcase	大文字を小文字に変換
ucase	小文字を大文字に変換
swab	奇数バイトと偶数バイトを入れ替える

表6　ddの変換オプション

df /bin

ファイルシステムの使用状況を表示する

UNIX系OSでは、複数のファイルシステムを1つのディレクトリツリーに統合して、単独のファイルシステムとして用いる。dfコマンドは、各ファイルシステムのサイズや使用状況を表示する。

```
$ df
Filesystem 1k-blocks     Used Available Use% Mounted on
/dev/hda5    1976492 1348764    527324  72% /
/dev/hda8    8428164 4890708   3109328  61% /home
```

各ファイルシステムにはroot用に予約された領域があるため、dfで表示される使用率は、実際よりも大きめになる。

デフォルトでは、1Kバイト（1024バイト）単位で表示されるが、オプション（**表7**）によって変更可能だ。

```
$ df -h
Filesystem    Size  Used Avail Use% Mounted on
/dev/hda5     1.9G  1.3G  515M  72% /
/dev/hda8     8.0G  4.7G  3.0G  61% /home
```

ln /bin

ファイルへのリンクを作成する

現在使われているほとんどのOSには、あるファイルを別のファイル名で参照するための仕組みが備わっている。Windowsの「ショートカット」やMac OSの「エイリアス」がその実例だ。

UNIX系OSでは、この機能を「リンク」と呼んでいる。リンクには「ハードリンク」と「シンボリックリンク」の2種

-k	1Kバイト（1024バイト）単位で表示
-m	1Mバイト（1048576バイト）単位で表示
-h	読みやすい単位で表示（1024倍）
-H	読みやすい単位で表示（1000倍）

表7　dfの表示に関するオプション

類がある。そのどちらもlnコマンドで作成することができる。

ハードリンクは、実際には1つしかないファイルに、複数の名前を付けている。以下のように作成する。

```
$ ln oldname newname
```

これで、今まで“ oldname ”という名前で参照していたファイルを、“ newname ”という名前でも参照できるようになる。実体は同じなので、ファイルnewnameの内容を変更したら、それはoldnameにも反映される。

ハードリンクで作成した複数の名前は同等なので、どちらか一方を削除してもまったく問題はない。上の例でoldnameを削除しても、そのファイルは引き続きnewnameという名前で参照可能だ。

実体が同じだということは、以下のように同一のiノード番号を持つことから確認できる。

```
$ ls
oldname
$ ln oldname newname
$ ls -i
 26730 newname    26730 oldname
```

ファイルとしての実体を共有するため、ハードリンクはパーティションをまたがって作成することはできない。

それに対してシンボリックリンクは、元のファイルに関する情報を記述したファイルを新たに作成する。Windowsの「ショートカット」やMac OSの「エイリアス」とほぼ同じ方法だ。シンボリックリンクを作成するには、-sオプションを付けてlnコマンドを実行すればいい。

```
$ ln -s original linkedfile
```

シンボリックリンクを作成後に元のファイルを削除してしまうと、もうそのファイルにはアクセスできなくなる。

```
$ ls
original
$ ln -s original linkedfile
$ ls
linkedfile   original
$ rm original
$ cat linkedfile
cat: linkedfile: No such file or directory
```

シンボリックリンクの実体は、元ファイルの情報が記されているファイルであるため、別のパーティション上に作成することもできる。

ls /bin

ファイルのリストを表示する

lsコマンドは、ディレクトリに含まれているファイルのリストを表示する。コマンドラインで作業を行っている際には、最も多く使われているコマンドであろう。

ディレクトリを指定すると、そこに含まれているファイル／サブディレクトリを、ファイルを指定すると、そのファイルをリスト表示する。デフォルトでは、「.」（ピリオド）で始まるファイルやディレクトリは表示されない（**表8**）。

mkdir/rmdir /bin

ディレクトリを作成／削除する

mkdirはディレクトリ作成、rmdirはディレクトリ削除を行うコマンドだ。どちらも複数のディレクトリを同時に指定することができる。rmdirで削除するディレクトリは、通常空でなければならない。

mkdirに-pオプションを付けると、複数の階層のディレクトリをまとめて作成することができる。

```
$ mkdir -p dir1/dir2
```

カレントディレクトリにdir1ディレクトリがなくても、順次dir1、dir2ディレクトリが作成される。また、すでに存在しているディレクトリをmkdirしようとするとエラーになる

-a	ピリオドで始まるファイルも表示する
-d	ディレクトリの中身を表示する代わりに、ディレクトリ名をリスト表示する
-F	ディレクトリ（/）、実行ファイル（*）、シンボリックリンク（@）などに印を付加する
-i	iノード番号を表示する
-l	長い形式でリスト表示する
-L	シンボリックリンクの元ファイルをリスト表示する
-R	サブディレクトリ以下を再帰的にリスト表示する
-t	修正時刻でソートする

表8　lsでよく使われるオプション

が、-pオプションはこのエラーも回避してくれる。

mkdirに-mオプションを付けると、作成するディレクトリのアクセス権を指定することができる。アクセス権の指定方法は、chmodコマンドと同様にすればいい。

rmdirに-pオプションを付けると、複数の階層のディレクトリがまとめて削除される。

```
$ rmdir dir1/dir2
```

上記のように指定すると、dir2ディレクトリだけが削除される。

```
$ rmdir -p dir1/dir2
```

それに対してこのように-pオプションを付けると、dir1以下をまとめて削除できる。

mount /bin

ファイルシステムをマウントする

UNIX系OSでは、複数のファイルシステムを1つのディレクトリツリーに統合して、単独のファイルシステムとして用いる。あるファイルシステムを別のファイルシステム上のディレクトリに割り当てることを「マウント」すると表現し、ファイルシステムをマウントするためのコマンドが、mountだ。基本的な指定方法は以下の通りだ。

```
# mount -t ext2 /dev/hda5 /home
```

この例では、/dev/hda5パーティションをext2ファイルシステムとして、/homeにマウントすることを表している。Linuxでは、標準のext2以外にも、FAT（DOS／Windows）、HFS（Mac OS）、iso9660（CD-ROM）など非常に多種のファイルシステムをマウントできる。

現在マウントされているパーティションやデバイスを確認したいときは、引数なしでmountコマンドを実行すれば、一覧が表示される。

OS起動時には、利用するパーティションが自動的にマウントされる。そのために必要な情報であるデバイス名、マウントされるディレクトリ、ファイルシステムなどは、/etc/fstabというファイルに記述されている。ここに記されているリムーバブルデバイスは、デバイス名、マウントされるディレクトリのいずれかを指定すれば、マウントできる。

```
/dev/cdrom  /cdrom  iso9660 noauto,owner,ro 0 0
```

上記のような記述が/etc/fstabにあれば、CD-ROMをマウントするためには、次のどちらの方法でもマウントできる。

```
# mount /dev/cdrom
# mount /cdrom
```

ハードディスク上のパーティションや、デバイス以外に、NFS、SMBFSなどを用いたネットワーク上のマシンにあるファイルシステム、そしてCD-ROMのイメージファイルなどもmountコマンドで扱うことが可能だ。

umount /bin

ファイルシステムをアンマウント（マウント解除）する

mountコマンドによってマウントされたデバイスやパーティションのアンマウント（マウントの解除）するのが、umountコマンドだ。

マウントされているディレクトリ、またはデバイス名のいずれかを指定すればいい。

```
# umount /mnt/cdrom
```

上の例では、/mnt/cdromにマウントされているデバイス（たいていは/dev/cdrom）がアンマウントされる。

アンマウントしたいファイルシステムが、何らかの形で利用されていると、アンマウントは失敗する。最もよくあるのは、ユーザーの誰かがカレントディレクトリをそのファイルシステム上のどこかにしている場合だ。また、そのファイルシステム上のファイルをオープンしている場合も、同様にエラーになる。

eject /usr/bin

リムーバブルメディアを取り出す

ejectコマンドを用いると、CD-ROM、フロッピーディスクなどのリムーバブルメディアをドライブから取り出すことができる。

引数には、デバイスファイル名またはマウントポイントの

ディレクトリ名を指定する。その際、デバイスファイル名の先頭の「/dev/」や、マウントポイントの先頭の「/mnt/」は、省略可能だ。たとえば、デバイス名が「/dev/hdc」で、「/mnt/cdrom」にマウントされているCD-ROMは、以下のどの方法でも取り出すことができる。

```
# eject /dev/hdc
# eject hdc
# eject /mnt/cdrom
# eject cdrom
```

もっとも、引数を指定しなかった場合のデフォルトは、CD-ROMなので、「eject」だけでもCD-ROMのトレイは開いてくれる。

リムーバブルメディアは、取り出される前に自動的にアンマウントされる。そのため、取り出そうとするリムーバブルメディアを誰かが使っている場合、つまりリムーバブルメディア上のディレクトリをカレントディレクトリにしていると、ejectコマンドは失敗する。

sync /bin

ディスク上のデータとキャッシュ／バッファの同期をとる

syncは、ディスク上のデータとメインメモリ上のキャッシュ／バッファの同期をとるためのコマンドだ。

ハードディスクなどの外部記憶装置は、メモリと比較するとアクセス速度がはるかに遅いため、ハードディスクへの書き込みが終了するのを待っていると、OS全体の性能が落ちてしまう。そこで多くのOSでは、メインメモリ上にキャッシュやバッファを持ち、性能を向上させている。

ただし、バッファ上のデータがハードディスクに書き戻される前に、停電などでマシンが停止すると、ハードディスクの整合性がとれなくなり、最悪の場合はデータが失われてしまう。

一般的にはsyncコマンドは、キャッシュやバッファの内容をハードディスクに書き戻すものとされている。しかしLinux上では、書き戻しの作業がスケジューリングされるだけで、すぐに書き戻しが行われるわけではないので、注意が必要だ。すなわち、syncコマンドを実行してから数秒は待たなければならない。rebootコマンドやhaltコマンドはこの処理を行ってくれる。

touch /bin

ファイルのアクセス時刻／修正時刻を設定する

touchコマンドは、ファイルのアクセス時刻と修正時刻を変更するコマンドだ。時刻を指定しなければ、現在の時刻に修正される。アクセス時刻だけを変更するには-aオプションを、修正時刻だけを変更するには-mオプションを用いる。

現在の時刻の代わりに特定の時刻に変更する場合は、**表9**に示した書式で指定する。「12時75分」「7月32日」のように、実際にはない時刻や日付は指定できない。

chattr /usr/bin

ext2ファイルシステム上にあるファイルの属性を変更する

ext2ファイルシステムでは、ファイルに通常の「読み出し」「書き込み」「実行」以外の属性を持たせることができる。これらの属性は、chattrコマンドによって設定／解除する。

```
$ chattr モード ファイル名...
```

モードは、「＋」「－」「＝」のいずれかのオペレータと、属性を表す文字（**表10**）からなる。「＋」は属性の設定、「－」は属性の解除を意味する。また「＝」は指定した属性のみが設定される。

du /usr/bin

ディスクの使用量を表示する

duコマンドは、ディスクの使用量を調べるためのコマンドだ。引数で指定したディレクトリ以下を再帰的に調べてくれる。特にディレクトリを指定しなかった場合は、カレントデ

-a	アクセス時刻だけを修正
-m	修正時刻だけを修正
-r file	現在の時刻の代わりに、fileのアクセス時刻／修正時刻を使用する
-t yyyymmddhhmm.ss	現在の時刻の代わりに、指定した時刻を使用する 時刻は以下の書式に従った十進数で指定する。 yyyy：年。2または4桁（1969〜2068）（省略可能） mm：月 dd：日 hh：時 mm：分 ss：秒（省略可能）

表9　touchでよく使われるオプション

ィレクトリ以下を調べる。

```
$ cd somewhere
$ du
1       ./work
15      ./.ssh
172454  .
```

-sオプションを付けると、引数で指定したディレクトリ以下の合計のディスク使用量だけを表示するので、階層が深いディレクトリに含まれるファイルの合計サイズを求めるのに利用できる。

デフォルトでは、1Kバイト（1024バイト）単位で表示されるが、オプション（**表11**）によって変更可能だ。

fdformat /usr/bin

フロッピーディスクを物理フォーマットする

fdformatは、フロッピーディスクを物理フォーマット（低レベルフォーマット）するためのコマンドだ。さまざまな形式のメディアに対応しているが、現在流通しているフロッピーディスクはほぼ2HD 1.44Mバイトに限られるうえ、FAT形式で論理フォーマットまでされているものがほとんどなので、fdformatを利用する機会はあまり多くはないだろう。

1.44Mバイトのフロッピーディスクを物理フォーマットするには、デバイス名を以下のように指定する（ドライブが1台の場合）。

```
$ fdformat /dev/fd0H1440
```

fdformatで物理フォーマットしたフロッピーディスクを利用するには、さらにmkfsコマンドで論理フォーマットを行う必要がある。

file /usr/bin

ファイルの種類を推測して表示する

fileコマンドは、指定されたファイルの種類を推測する。はじめに、そのファイルがディレクトリやデバイスファイルなのか、あるいは通常のファイルなのかを調べ、通常のファイルであれば、その内容をもとにファイルの種類を調べる。

ファイルの種類はtext、executable、dataの3種類に大別される。textは、ASCIIキャラクタだけを含み、catなどで画面に表示できるファイル、executableは実行ファイル、そのどちらでもないものはdataに分類される。さらに、ファイルの内容を見て、それがデータベースに登録された形式であればより詳しく種類を表示する（**画面4**）。このときに参照されるデータベースは、/usr/share/magicというファイルで、実行ファイルやプログラムのソースなどよく使われるファイル形式が登録されている。

diff/diff3 /usr/bin

ファイル間の差を表示する

プログラム開発時など、ファイルの修正を何度も繰り返す

A	アクセス時刻を更新しない
a	追加書き込みのみ許可
c	自動的に圧縮する
d	dump起動時にバックアップされない
i	変更不可
s	安全な削除（データを0で上書きする）
S	同期更新
u	復活可能

表10　chattrで指定できる属性

-k	1Kバイト（1024バイト）単位で表示
-m	1Mバイト（1048576バイト）単位で表示
-h	読みやすい単位で表示（1024倍）
-H	読みやすい単位で表示（1000倍）
-s	合計の使用量を表示する
-c	複数項目の合計使用量を表示する
-x	別のパーティションにあるディレクトリは除外する

表11　duでよく使われるオプション

```
$ file Desktop/
Desktop/: directory
$ file /bin/bash
/bin/bash: ELF 32-bit LSB executable, Intel 80386, version 1, dynamically linked (uses shared libs),
stripped
$ file ./dumper.tar.gz
./dumper.tar.gz: gzip compressed data, deflated, last modified: Sat Jul 15 17:00
:14 2000, os: Unix
```

画面4　ディレクトリ、実行ファイル、アーカイブファイルを調べた結果

ような作業を行うと、内容の一部だけが異なるファイル群が生じる。これらのファイル間の差異を表示するためのコマンドが、diffやdiff3である。diffが2つのファイルを扱うのに対して、diff3は3つのファイルを対象とする。diff3は、元となるファイルに2人のユーザーが同時に変更を加えてしまった場合などに有効だ。

diffには用途や好みに合わせて出力フォーマットが数種類用意されているが、context形式とunified形式（**画面5**）がよく使われている。

ソースの状態で配布されているプログラムでは、バージョンアップの際に「パッチ」と呼ばれる比較的小さなファイルが提供されることがある。パッチの実体は、新旧のソースファイル間でdiffを取った出力である。変更点がそれほど多くなければ、パッチはソース全体よりもはるかに小さくなるため、新たに配布する際の負担が少なくて済む。

patch /usr/bin

差分ファイルをオリジナルファイルに適用する

patchコマンドは、diffコマンドによって作られた差分情報を含むパッチファイルをオリジナルのファイルに適用し、変更後のファイルを生成する。フリーソフトウェアのバージョンアップ時には、ソースへのパッチファイルを配布することが多い。patchコマンドを使って旧バージョンのソースにパッチを当てることで、新バージョンのソースファイルになるわけだ。

パッチファイルには、差分を適用するファイルのパスが書かれているが、パッチを作成した環境以外でpatchコマンドを使う場合には必ずしも元のパスが有効ではない。そのような場合に有効なのが-pオプションだ。-pに続けて数字Nを指定するとパッチファイル内に書かれたパスのN番目の/（スラッシュ）までを削除して解釈する。たとえば、Linuxカーネル2.4.6から2.4.7への差分ファイルは、v2.4.6/linux/以下のファイルにパッチを適用するように作られているが、カーネルのソースが/usr/src/linux以下に展開されている場合は、/usr/srcディレクトリに移動して、

```
$ patch -p1 < patch-2.4.7
```

を実行すればよい。この場合、差分ファイルpatch-2.4.7の中に書かれたv2.4.6/linux/……の1つめの/までが無視され、linux/……と解釈されるので、カレントディレクトリのlinuxディレクトリ以下のファイルにパッチが適用される。

```
$ diff -u base.txt update.txt
--- base.txt    Wed Jul 15 05:14:38 2001
+++ update.txt  Wed Jul 15 05:15:04 2001
@@ -1,6 +1,6 @@
 The quick brown
 fox jumps over
-the lazy dog.
+the lazy tux.

 This is a sample document.
```

画面5　diffの出力例（unified形式）

tar /bin

アーカイブファイルを扱う

tarは、複数のファイルをひとまとめにしたアーカイブファイルを作成したり、元のファイルを取り出したりするためのコマンドだ。以前は、このアーカイブファイルをテープに格納してバックアップを取っていた。「tar」という名前も、「Tape ARchiver」からとられている。

tarで作成したアーカイブファイルを別項の圧縮ツール（gzip、bzip2）でサイズを小さくしたものを「tarボール（tar玉）」と呼ぶ。大部分のフリーソフトは、ソースファイルやドキュメントなどをひとまとめにしたtarボールの状態で配布されている。

tarコマンドは多くの機能を持つが、よく使うのはアーカイブの作成（c）、内容の確認（t）、そしてファイルの抽出（x）だ。圧縮ツールのgzipを利用する際には「z」、bzip2ならば「I」を指定するので、tarのコマンドラインは、**表12**のようになることが多いだろう。

cpio /bin

アーカイブファイルからファイルの入出力を行う

cpioコマンドは、アーカイブファイルからファイルの抽出や格納を行う。cpio形式以外に、tar形式、いくつかのバイナリ形式、ASCII形式などのアーカイブファイルに対応している。コピーイン、コピーアウト、コピーパスの3種のモードを持つ。

コピーインモードでは、アーカイブファイルを読み込み、指定したファイルを抽出する（実際には標準入力をアーカイブファイルにリダイレクトする）。

コピーアウトモードでは、標準入力からファイルのリスト

を読み込み、それらのファイルをアーカイブファイルに格納する（実際には標準出力をアーカイブファイルにリダイレクトする）。

コピーパスモードは、コピーアウト→コピーインと連続して行うことにより、ファイルを別のディレクトリにコピーする。各ファイルの属性やその他の情報を保持したまま、ファイルのコピーを行える。

gzip/gunzip/zcat /bin

ファイルの圧縮／伸張を行う

gzip、gunzip、zcatはLZ77アルゴリズムを用いて、ファイルの圧縮／伸張を行う。gzipが圧縮、gunzipが伸張、そしてzcatは伸張した内容を標準出力に書き出す。これら3つのコマンドは、ハードリンクで実現されており、「gzip -d」でgunzip、「gzip -c」でzcatと同様の動作をする。

そのほか、-lオプションで圧縮される前のファイルのサイズなどを表示、-1～9で圧縮率と圧縮時間を調整できる。数字が小さいと圧縮率は低いが高速になり、大きければ、時間をかけて高い圧縮率を得る。

bzip2/bunzip2/bzcat /usr/bin

ファイルの圧縮／伸張を行う

bzip2、bunzip2、bzcatは、ブロックソートテキスト圧縮アルゴリズムを用いて、ファイルの圧縮／伸張を行う。広く用いられているgzipのLZ77アルゴリズムに、特許上の問題があるため、新たに作られた。一般的に圧縮率はgzipよりも高いが、圧縮にかかる時間はgzipよりも長く、必要なメモリ量も多い。bzip2が圧縮、bunzip2が伸張、そしてbzcatは伸張した内容を標準出力に書き出す。bunzip2、bzcatは、bzip2へのシンボリックリンクであり、実際に起動するプログラムは同じだ。

-kオプションを付けると、圧縮／伸張後に元のファイルを削除しない。通常はこのほうが使いやすいだろう。

tar cvzf program-1.0.tar.gz ./program-1.0/ ./program-1.0/ディレクトリ以下をまとめてアーカイブファイルを作成する
tar xvzf program-1.0.tar.gz アーカイブファイルから元のファイルを抽出する
tar tvzf program-1.0.tar.gz アーカイブファイルの内容をリスト表示する （bzip2を使う場合は、「z」の代わりに「I」）

表12　tarでよく使われるオプションの組み合わせ

-atime N	最後にアクセスされたのがN×24時間前
-ctime N	最後にステータス変更されたのがN×24時間前
-mtime N	最後にファイルが更新されたのがN×24時間前
-group グループ名	ファイルのグループID
-user ユーザー名	ファイルの所有者での検索
-name パターン	ファイル名による検索
-size N	ファイルのサイズがNブロック

表13　検索式の主なオプション
※＋NはNよりも大きい、－NはNよりも小さい、Nはちょうど Nの指定となる

compress/uncompress /usr/bin

ファイルの圧縮／伸張を行う

compressとuncompressは、ファイルの圧縮／伸張を行うコマンドだ。UNIX系OSのごく初期から利用されていたが、今ではgzipが代わりに用いられている。また、compressで圧縮したファイル（「.Z」の拡張子を持つ）は、gzipで伸張できるため、現状では積極的にcompressを使う理由はない。

find /usr/bin

ディレクトリ階層下にあるファイルを検索する

ファイル名や所有者などの条件で、指定したディレクトリ以下からファイルを検索する。コマンドの書式は、

```
find ［パス］［検索式］［アクション］
```

という形で、“ パス ” 以下のディレクトリにある、“ 検索式 ” に合致したファイルを探し、それらに対して “ アクション ” で指定した処理を行う。“ 検索式 ” には、**表13**のようなオプションが利用できる。

“ アクション ” には、**表14**のようなオプションを指定する。指定しないと、-printが指定されたものとみなし、検索結果を表示する。**画面6**の上は、ファイル名による検索の例だ。ファイル名の指定を引用符で囲んでいるのは、シェルに “ ＊ ” を展開させないためだ。

また、-execオプションを使うと、検索したファイルに対して操作を行うことができる。**画面6**の下がその実行例である。“ -exec ” から “ ; ” までの間に実行したいコマンドを指定する。“ ; ” の前に “ \ ” が付いているのは、先ほどの “ ＊ ” と同様、シェルに解釈をさせないための処置だ。“ {} ” の部分は、検索結果のファイル名に展開される。

xargs

/usr/bin

標準入力からコマンドラインを作成し、それを実行する

標準入力からスペースまたは改行で区切られた文字列を受け取り、それを引数としたコマンドラインを作成してコマンドを実行する。書式は次の通り。

```
xargs [オプション] コマンド [コマンドの引数]
```

xargsは、作成した引数を“コマンド”と“コマンドの引数”の間に挿入して“コマンド”を実行する。findコマンドと組み合わせて使うことが多い。たとえば、次の例は、カレントディレクトリ以下からファイル名が.cで終わるもの（C言語のソースファイル）、または.hで終わるもの（ヘッダファイル）を検索し、該当するファイルの中からUDMAという文字列を検索する。

```
$ find . -name '*.[ch]' | xargs grep UDMA
```

findコマンドの出力は、ファイルのパスが1行に1個ずつ書かれたものだが、xargsはそれらをスペース区切りにしてgrepの第1引数に、UDMAという文字列を第2引数にする。

作成した引数がコマンドラインで許される文字数を超える場合は、“コマンド”を複数回起動するのがxargsのデフォルト動作だ。-Pオプションを使い、複数の“コマンド”を並列動作させることもできる。

-exec コマンド ;	“コマンド”を実行する
-print	ファイル名をフルパスで表示する

表14　アクションの主なオプション

/usr/share/doc以下から、ファイル名が“.jp”で終わるものを検索

```
$ find /usr/share/doc -name '*.jp'
/usr/share/doc/Canna-3.5b2/CHANGES.jp
/usr/share/doc/Canna-3.5b2/INSTALL.jp
..
..
/usr/share/doc/postgresql-7.0.3/README.mb.jp
/usr/share/doc/parted-1.4.7/USER.jp
```

ホームディレクトリ以下から、60日以内に更新されたファイルを検索し、fileコマンドで種類を調べる

```
$ find ~ -mtime -60 -exec file {} \;
/home/qchan: directory
/home/qchan/.bash_history: ASCII text
/home/qchan/fwconf.xwd: data
/home/qchan/netcfg.xwd: data
/home/qchan/jump.html: HTML document text
```

画面6　findの使用例

tree

/usr/bin

ディレクトリ階層下をツリー表示する

treeコマンドは、指定したディレクトリ以下のファイルやディレクトリをツリー表示するコマンドだ（**画面7**）。デフォルトでは、ファイルとディレクトリを両方表示するが、-dオプションを付けるとディレクトリのみを表示する。たとえば、/etc以下のディレクトリ構造を表示するには次のようにする。

```
$ tree -d /etc
```

また、-aオプションで、ファイル名が「.」（ピリオド）で始まる隠しファイルも表示する。このほかにも数多くのオプションが用意されているので、manを参照していろいろと試してほしい。

chown

/bin

ファイルの所有者を変更する

chownは、ファイルやディレクトリの所有者（owner）と所有者が所属するグループ（複数のグループに所属している場合は、そのうちの1つ）を変更するコマンドだ。コマンドラインは、以下のように指定する。

```
$ chown ユーザー ファイル...
```

また、ユーザーと同時にグループを変更する際には、以下

```
$ tree
.
|-- Desktop
|   |-- Autostart -> ../.kde/Autostart
|   |-- Linux Documentation
|   |-- Printer
|   |-- kontrol-panel
|   `-- www.redhat.com
|-- aaa.txt
`-- bbb.txt

2 directories, 6 files
```

画面7　ディレクトリとファイルをツリー表示する

のように「:」で区切って指定する（ピリオドでも可）。

```
$ chown ユーザー:グループ ファイル...
```

上記の書式で指定する際にユーザー名を省略すると、グループだけを変更できる。また、逆にグループ名を省略すると、指定したユーザー名のログイングループを指定したことになる。なお、いずれの場合も、ユーザー名／グループ名の代わりに、ユーザーID／グループIDを数値で直接指定することも可能だ。

一般ユーザーの場合、ファイルの所有者を変えること、つまり自分が所有しているファイルをほかのユーザーの所有にすることや、その逆はできないが、複数のグループに所属している際に、所有者グループを変更することはできる。

chownコマンドは、管理者（root）が一般ユーザーのホームディレクトリに、設定ファイルを置いたりする際に利用することが多いだろう。

-Rオプションを付けることで、ディレクトリ内のファイルやディレクトリを再帰的に処理することができる。

chmod /bin

ファイルのアクセス権を変更する

UNIX系OSでは、各ファイルについて、所有者（u）、所有者と同じグループのメンバー（g）、そしてそれ以外のメンバー（o）ごとに「読み取り」（r）、「書き込み」（w）、「実行」（x）の3種類のアクセス権を設定することができる。chmodコマンドは、このアクセス権を設定するコマンドだ。

上記のu、g、o、または全ユーザーを表すaのいずれかを使って、誰に対するアクセス権を設定するか指定する。そして、r、w、xの各アクセス権を設定（＋）または解除（－）するように指定する。

```
$ chmod g-rw test.txt
```

このように指定するとtest.txtというファイルについて、所有者と同じグループのメンバーに対する「読み取り」と「書き込み」のアクセス権が解除される。

またこれらのアクセス権を、3桁の8進数を用いて直接指定することもできる。上の桁から順に、所有者、所有者と同じグループのメンバー、そしてそれ以外のメンバーを表す。「読み取り」、「書き込み」、「実行」のアクセス権はそれぞれ4、2、1を表す。

```
$ chmod 664 shared.txt
```

上記のように指定するとshared.txtというファイルは、所有者とグループのメンバーは読み書きができ、それ以外のユーザーは読み取りのみできるようになる。

chgrp /bin

ファイルの所有者グループを変更する

UNIX系OSでは、各ファイルやディレクトリに所有者と所有者が所属するグループを指定することができる。chgrpは、ファイルやディレクトリの所有者グループ（group）を変更するためのコマンドだ。

```
$ chgrp グループ ファイル...
```

「グループ」として指定できるのは、ユーザーが所属しているグループに限られる。実際にはグループを変更する際には、あわせて所有者も変更することが多いため、所有者とグループをまとめて変更するchownコマンドがよく用いられる。

stat /usr/bin

ファイルやファイルシステムの状態を表示する

statコマンドは、ファイルの状態を調べるのに使う（**画面**

```
$ stat linux-2.4.6.tar.gz
  File: "linux-2.4.6.tar.gz"
  Size: 26940370   Blocks: 52688     Regular File
Access: (0664/-rw-rw-r--)         Uid: (  500/ p-chan)  Gid: (  500/ p-chan)
Device: 305        Inode: 702937     Links: 1
Access: Sun Jul 22 18:30:30 2001
Modify: Sun Jul 22 18:30:04 2001
Change: Sun Jul 22 18:30:04 2001
```

画面8　statでファイルの状態を表示

8)。このほか、-fオプションを使うことで、そのファイルが含まれるファイルシステムの状態を見ることもできる。

ps /bin

動作中のプロセスをリスト表示する

psコマンドは、動作中のプロセスをリスト表示する。表示する情報の選択やプロセスの選び方に応じて、多くのオプションを持つ。オプションの指定方法は、OSごとに異なっているが、現在Linuxディストリビューションに含まれているpsコマンドは、どの指定方法でも受け入れるため、必要以上に複雑になっている。実際には、頻繁に使われるオプションは、それほど多くない。

オプションなしでpsを起動すると、そのターミナル上で動作しているプロセスのうち、自分が所有しているものだけが表示される。

```
$ ps
  PID TTY          TIME CMD
29633 pts/3    00:00:00 bash
29752 pts/3    00:00:00 ps
```

aオプションを付けると、ほかのユーザーが所有しているプロセスも表示される。

```
$ ps a
  PID TTY      STAT   TIME COMMAND
29632 pts/3    S      0:00 login -- tux
29633 pts/3    S      0:00 -bash
29753 pts/3    R      0:00 ps a
```

上記の例では、rootが所有しているloginコマンドが追加されていることがわかる。

uオプションは、メモリやCPUの使用率など、より有用な情報をあわせて表示する。さらにxオプションは、ターミナルから操作できないプロセス（デーモンなど）も表示する。

上記のオプションをあわせて「ps aux」とすると、現在動作中の全プロセスが詳細な情報とともに表示される。

top /usr/bin

プロセスをCPU利用時間順などで表示する

topコマンドは、CPUの利用時間やメモリの使用量順にプロセスをソートして表示するコマンドだ。オプションを付け

```
  7:26pm  up 61 days, 22:08,  6 users,  load average: 1.00, 1.00, 1.00
51 processes: 46 sleeping, 4 running, 1 zombie, 0 stopped
CPU states:  0.0% user,  0.1% system, 99.8% nice,  0.0% idle
Mem:   255528K av,  250544K used,    4984K free,       0K shrd,    7668K buff
Swap:  265064K av,   18332K used,  246732K free                  206116K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
30466 root       9   0 49800  48M  2140 S     0.0 19.4   0:01 X
  875 xfs        9   0 11856 4164  3184 S     0.0  1.6   0:44 xfs
30474 gdm        9   0  3552 3300  2772 S     0.0  1.2   0:00 gdmlogin
12486 root       8   0  1428 1428  1056 S     0.0  0.5   0:00 bash
12441 root       9   0  1384 1384  1008 S     0.0  0.5   0:00 bash
12400 root       9   0  1312 1312  1060 S     0.0  0.5   0:00 login
12401 q-chan     9   0  1296 1296  1008 S     0.0  0.5   0:00 bash
12437 q-chan     9   0  1168 1168   884 R     0.1  0.4   0:00 top
12440 root       9   0  1100 1100   876 S     0.0  0.4   0:00 login
12253 root       9   0   908  908   768 S     0.0  0.3   0:00 run-parts
12264 root       9   0   880  880   756 S     0.0  0.3   0:00 sa1
12399 root       9   0   828  828   688 R     0.0  0.3   0:00 in.telnetd
12482 root       9   0   720  720   572 S     0.0  0.2   0:00 kon
30467 root       9   0   824  672   632 S     0.0  0.2   0:00 gdm
12252 root       9   0   580  572   492 S     0.0  0.2   0:00 crond
12263 root       9   0   552  552   464 S     0.0  0.2   0:00 awk
12266 root       9   0   512  512   448 S     0.0  0.2   0:00 sadc
```

画面9　topでメモリ使用量順にプロセスを表示したところ

ないで実行すると、CPU負荷の順にソートされるが、実行中にM（大文字）を入力するとメモリの使用量、N（大文字）でプロセスID順にソートできる。CPU負荷の順に戻す場合はC（大文字）を入力する（**画面9**）。

このほかにも、プロセスをkillしたり、reniceすることも可能だ。hを入力するとヘルプ画面になるので参考になるだろう。topを終了させるにはqを入力する。

uptime /usr/bin

システムの連続稼働時間を表示する

システムが起動してからの時間、ログインしているユーザーの数、過去1分、5分、15分のロードアベレージを表示する（**画面10**）。ロードアベレージとは、実行中のプロセスと実行待ちのプロセス数の数を足したものだ。シングルプロセッサのシステムでは、実行中のプロセス数は0～1になる。したがって、実行中、あるいは実行待ちのプロセスがない（とっても暇な）場合のロードアベレージは0になる。常にひとつのプロセスが実行され、待ちプロセスがなければロードアベレージは1、実行を待たされているプロセスがあれば1よりも大きな値となる。

nice /bin

優先度を設定してコマンドを実行する

niceコマンドは、優先度（ナイス値）を設定して任意のコマンドを実行する。書式は以下の通りだ。

```
$ nice 優先度 コマンド 引数...
```

“優先度”は、「-n 数字」のように指定する。優先度を省略した場合のデフォルトは、10である。ナイス値は、-20（最優先）から19（最も優先度が低い）の範囲をとる。一般ユーザーは正の値、つまり優先度を下げる方向にのみ指定可能である。

非常に時間がかかりそうなコマンドを実行する際にniceコマンドを利用するといい。優先度を下げることで、ほかのユーザーが走らせているプログラムの実行を邪魔しなくなるからだ。

すでに実行中のプログラムの優先度を変更するには、reniceコマンドを利用する。

renice /usr/bin

実行中のプロセスの優先順位を変更する

Linuxはマルチタスクなので、複数のプログラムを同時に動作させることができる。厳密には各プログラムを高速に切り替えながら順に実行することで同時に動いているように見せている。reniceコマンドは、指定したプログラム（プロセス）の実行優先順位を変更するコマンドだ。

一般ユーザーは自分の実行したプロセスの優先順位を下げることしかできないが、rootユーザーは全ユーザーのプロセスの優先順位を上げたり下げたりできる。コマンドの書式は次の通り。

```
renice 優先度 プロセスID
```

“優先度”は数値で指定する。この数値はniceness（人のよさ）と呼ばれるもので、数が大きいほど優先順位が低くなる。一般ユーザーは0～19まで、rootユーザーは-20～19までを指定できる。目的とするプロセスの“プロセスID”（PID）は、psコマンドやtopコマンドを実行して調べよう。

killall /usr/bin

名前で指定したプロセスにシグナルを送る

killallコマンドは、指定した名前のコマンドを実行している全プロセスにシグナルを送る。シグナルは、-HUPのように名前で指定するか、-1のように番号で指定する。シグナルの指定をしなかった場合はSIGTERMが送られる。

```
$ killall -HUP myprog
```

自分のセッション以外の全プロセスにシグナルを送るシステムV系UNIXのkillallコマンドとは働きが違うので注意してほしい。Linuxでは、killall5コマンドがシステムV系のkillallと同等の働きを持つ。

```
$ uptime
 11:26pm  up 62 days,  2:09,  4 users,  load average: 0.00, 0.00, 0.00
```

画面10　uptimeコマンドの実行例

at /usr/bin

指定した日付／時刻にコマンドを実行する

atは、ユーザーが指定した日付／時刻に任意のプログラムを実行するためのコマンドだ。引数にはプログラムを実行させたい時間を指定する。at起動後に「at>」というプロンプトが表示されるので、実行するプログラムを指定する。また、at起動時にプログラム名を記述したファイルを標準入力にリダイレクトすることもできる。

時刻の設定は、「HH:MM」形式で指定するのが基本だが、「am」「pm」を付けたり、コロンを省略できるなど、柔軟な表記が可能だ。日付についてもたとえば「July 15」という形式が基本だが、曜日（Mondayなど）や「today」「tomorrow」といった表記も受け付ける。さらにキーワード「now」のあとに数字と単位を指定することで、「今から○○後に実行」という書き方も可能だ（**表15**）。

/etcディレクトリ以下にat.allow、at.denyというファイルを置くことによって、atコマンドを利用できるユーザーを制限できる。at.allowがあれば、そこに書いてあるユーザーだけが利用できる。at.allowがなく、at.denyだけある場合は、at.denyに書かれていないユーザーだけが利用できる。デフォルトでは空のat.denyがあるので、全ユーザーが利用できる。どちらのファイルもない時は、rootユーザーだけに限られる。

crontab /usr/bin

各ユーザー用のcrontabファイルを管理する

UNIX系OSでは、定期的にプログラムを実行するためにcronデーモンが動作している。crontabは、各ユーザーが定期的に実行させたいプログラムのリストを管理するためのコマンドだ。これらのリストは、/varディレクトリ以下に置かれるが、直接編集することはできないようになっている。

時刻の表記	hh:mm形式で指定（コロンは省略可能） am、pmを付加できる midnight（0時）、noon（正午）、teatime（16時）
日付の表記	mmddyy形式で指定 （英語の）月名+日（たとえばJuly 15など） today（今日）、tomorrow（明日）
相対表記	「○○後」という形式で指定 「now + 数字 単位」（たとえばnow +3hour） minute、hour、day、week、month、yearが利用できる

表15　atの時刻／日付の書式

```
$ crontab ［-u ユーザー名］ ［ファイル］
```

上記のように指定することで、ファイル内に記述されている内容が、新たにcrontabの内容となる。-uオプションを用いてユーザー名を指定しなかった場合は、crontabコマンドを実行しているユーザーが利用される。suコマンドを利用している際には、どちらのユーザーのリストを扱っているのかわかりにくいため、明示的に-uオプションで指定するのが安全だ。

そのほか、オプションの指定によって、現在のリストの出力、削除、編集が行える（**表16**）。

free /usr/bin

メモリの利用状況を表示する

freeコマンドを実行すると、**画面11**のようにシステムのメモリ利用状況が表示される。

最初の行は、左から順に利用可能な物理メモリの総量（total）、使用中のメモリ（used）、空きメモリ（free）、共有メモリ（shared）、ディスクバッファ（buffers）、メモリ管理用のハッシュテーブルに入っているメモリ（cached）を表している。2行目は、buffersとcachedを利用可能な空きメモリと考えたときの使用中メモリ量と空きメモリ量を表している。最後の行はスワップ領域の状況だ。

vmstat /usr/bin

仮想メモリの使用状況を表示する

仮想メモリの状態を見るためのコマンド（**画面12**）。引数に数字を指定すると、その秒数ごとに情報が更新される。出力される内容の意味は**表17**の通り。

finger /usr/bin

ユーザー情報を調べる

finger ［user］を実行すると、“ user ”の情報を調べられる（**画面13**）。“ user@host ”の形式で指定すれば、リモートホストのユーザー情報を調べることも可能だが、この場合リ

-l	現在のリストを出力
-r	現在のリストを削除
-e	環境変数EDITORまたはVISUALで指定したエディタでリストを編集

表16　crontabのモード指定

```
$ free
             total       used       free     shared    buffers     cached
Mem:        255528     247480       8048          0      48148      33836
-/+ buffers/cache:     165496      90032
Swap:       265064       1632     263432
```

画面11　カーネル2.4では、sharedは常に0と表示される

```
$ vmstat
   procs                  memory    swap      io     system         cpu
 r  b  w   swpd   free   buff  cache  si  so    bi    bo   in   cs  us  sy  id
 1  0  0  18332   7584  14636 178612   0   0     0     0    7    1   6   0   6
```

画面12　vmstatの出力

モートホストでfingerデーモンが動作していなければならない。セキュリティ強化、個人情報保護の気運が高まる昨今では、fingerデーモンを動かしているサイトは少ないだろう。

last /usr/bin

最近ログインしたユーザーのリストを表示する

lastコマンドを実行すると、/var/log/wtmpファイルに記録されているログイン記録をもとに、各ユーザーのログイン／ログアウト記録を表示する。－（ハイフン）に続けて表示したい行数を指定できる（**画面14**）。また、引数にユーザー名を指定すれば、特定のユーザーについての記録を参照することもできる。

lastlog /usr/bin

ユーザーの最終ログイン時刻などを表示する

各ユーザーの最終ログイン時刻は、/var/log/lastlogというバイナリファイルに記録されている。lastlogコマンドはこのファイルをもとに、ユーザーの最終ログイン時刻などを表示する（**画面15**）。binやdaemonのようなシステムが利用する特殊ユーザーは、端末からログインすることはあり得ないので、これらのユーザーがログインした記録が残っていたらクラックされた可能性が高い。

who /usr/bin

ログインしているユーザーを表示する

whoコマンドを実行すると、現在ログインしているユーザーのログイン名、端末名、ログイン時間が表示される。

procs	
r	実行中、実行待ちのプロセス数
b	ディスクI/O待ちなどでブロックされているプロセス数
w	スワップアウトされているプロセス数
memory	
swpd	使用しているスワップ領域（kバイト）
free	空きメモリ容量（kバイト）
buff	ディスクバッファ容量（kバイト）
cache	メモリ管理用ハッシュテーブル（kバイト）
swap	
si	スワップイン（kバイト／秒）
so	スワップアウト（kバイト／秒）
io	
bi	ブロックデバイスへの書き込み（kバイト／秒）
bo	ブロックデバイスからの読み込み（kバイト／秒）
system	
in	割り込み（回／秒）
cs	コンテキスト切り替え（回／秒）
cpu	
us	ユーザーCPUタイム
sy	システムCPUタイム
id	アイドルCPUタイム

表17　vmstatで出力される内容

```
$ finger qchan
Login: qchan                            Name: (null)
Directory: /home/qchan                  Shell: /bin/bash
On since Thu Jul 19 17:14 (JST) on pts/2 from lm89.pb.ascii.co.jp
No mail.
No Plan.
```

画面13　fingerの実行例

```
$ last -5
qchan    pts/4        lmpc12          Sat Jul 21 00:22   still logged in
pchan    pts/4        lmmc30          Fri Jul 20 22:12 - 22:12  (00:00)
doronpa  pts/3        lmpc11          Fri Jul 20 18:12   still logged in
kemumaki pts/3        lmpc00          Thu Jul 19 23:59 - 00:06  (00:06)
doronpa  pts/3        lmpc11          Thu Jul 19 22:24 - 22:27  (00:02)

wtmp begins Sun Jul  1 20:49:21 2001
```

画面14　過去5つのログイン記録を表示

```
$ who
q-chan   pts/0    Jul 22 19:23
root     pts/3    Jul 19 17:15
root     pts/4    Jul 19 17:16
root     pts/5    Jul 19 17:16
```

リモートログインしているユーザーは、上記に加えリモートホスト名も表示される。

who am iと入力して実行すると、ホスト名に続けてwhoコマンドを起動したユーザーの情報を表示する（実は、オプションなしで引数が2つ付いていればwho are youだろうとなんだろうとこうなるのだが、慣習的にwho am iが使われている）。

whoami /usr/bin

現在有効なユーザーIDを表示する

自分自身が誰なのかわからなくなったときに使用する。もちろん、ふざけているのではなく、suコマンドで別のユーザーになっていると、現在のユーザーIDがわからなくなることがあるのだ。ほとんどのディストリビューションでは、初期状態でシェルのプロンプトにユーザー名を表示するようになっているので混乱する人は少ないだろうが、プロンプトを変更している場合には便利なコマンドだ。

現在有効なユーザーIDではなく、ログインしたときのユーザーIDが知りたいときは、“ who am i ”を実行しよう。

w /usr/bin

ログインしているユーザーと実行中のプロセスを表示する

uptimeコマンドの出力とともに、現在マシンにログインしているユーザーとそのユーザーが実行しているプロセスを表示する（**画面16**）。

dmesg /bin

カーネルが出力するメッセージを表示する

dmesgは、カーネルが出力するさまざまなメッセージを表示するコマンドだ。

Linuxの起動時には、ハードウェアの初期化や各種プログラムの起動が行われ、その結果が画面に表示される。これらのメッセージは、トラブルシューティングの役に立つが、最近の高速なマシンだとメッセージがどんどんスクロールしてしまうので、目で追うのは困難だ。dmesgコマンドを用いれば、あとからこれらのメッセージを確認できる。

```
$ dmesg | less
```

dmesgの出力は、たいてい一画面には収まらないので、上記のようにlessやmoreなどのページャと併用するといいだろう。

カーネルメッセージを保存する領域には限りがあるため、メッセージが増えてくると、古いものから順に削除されてい

```
Username         Port     From             Latest
root             :0                        土  6月 16 17:07:41 +0900 2001
bin                                        **Never logged in**
daemon                                     **Never logged in**
..
..
squid                                      **Never logged in**
qchan            pts/2    lmpc11.pb.ascii. 金  7月 20 23:23:49 +0900 2001
```

画面15　lastlogの実行結果

く（リングバッファ）。そのため、マシン起動時のメッセージは、起動直後でないと見られない可能性がある。

ping /bin
ICMPのECHO_REQUESTパケットをネットワーク上のホストに送る

ICMP（Internet Control Message Protocol）は、ネットワークの状態を調査するためのプロトコルであり、フロー制御や到達不能なホストの検出などが行える。

pingは、ICMPのECHO_REQUESTパケットを指定したホストに向けて送信し、その返答によってネットワークやホストの状態をチェックするためのコマンドだ。

```
$ ping www.ascii.co.jp
```

上記のように、ドメイン名またはIPアドレスでホストを指定する。ECHO_REQUESTを受けたホストはECHO_RESPONSEパケットを返す。その結果は、pingによって**画面17**のように表示される。

ネットワーク上のトラブルは、非常に多くの要因が考えられるため、その解決に手間取ることが多いが、pingコマンドを使って問題点を切り分けることが、解決の手助けになる。

現在ではセキュリティ上の観点から、ECHO_REQUESTパケットに応答しないように設定されているホストもある。そのようなホストにしつこくpingを打つと、場合によっては攻撃や攻撃のための調査とみなされる可能性もあることも気に留めておこう。

netstat /bin
ネットワークに関するさまざまな情報を表示する

netstatは、Linuxのネットワークシステムに関するさまざまな情報を表示する。表示される情報の種類は、引数で指定する（**表18**）。

nslookup /usr/bin
DNSによる名前解決を行う

ホスト名を指定してIPアドレスを引いたり、逆にIPアドレスを指定してIPアドレスを調べるのに使う（**画面18**）。オプションを指定してMXレコードなどを解決することもできる。

dig /usr/bin
ドメイン名に関する問い合わせをネームサーバに送る

dig（Domain Information Groper）は、ドメイン名に関する情報を集めるためのコマンドだ（**画面19**）。同じ用途のコマンドとして、nslookupが存在するが、digのほうがより柔軟な使い方ができるようになっている。

whois /usr/bin
ドメイン名などをwhoisサーバに問い合わせる

ドメイン名からそれを所有する組織や管理者を調べるのに

```
$ w
 12:10am  up 62 days,  2:52,  4 users,  load average: 0.31, 0.37, 0.39
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
q-chan   pts/0    lmpc11            7:23pm  0.00s  0.26s  0.00s  w
doronpa  pts/1    lmpc33           Thu 7pm  3days  1.01s  0.45s  slogin flsvr00.
o-chan   pts/3    lmpc41           Fri 6pm  1:08  10.69s 10.44s  slogin lmsvr05.
```

画面16　wコマンドの実行例

```
$ ping www.ascii.co.jp
PING at2.ascii.co.jp (210.140.231.23) from 172.16.42.91 : 56(84) bytes of data.
64 bytes from 210.140.231.23: icmp_seq=0 ttl=244 time=43.2 ms
64 bytes from 210.140.231.23: icmp_seq=1 ttl=244 time=40.1 ms
64 bytes from 210.140.231.23: icmp_seq=2 ttl=244 time=117.9 ms

--- at2.ascii.co.jp ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 40.1/67.0/117.9 ms
```

画面17　pingコマンドの使用例

使う。たとえば、example.comについて調べるなら、

```
$ whois example.com
```

を実行すればよい。whoisデータベースから得られた情報をネットワーク管理以外の目的で使用してはいけない。

locate/slocate /usr/bin

ファイル名データベースでファイルを探す

locateコマンドは、指定されたファイルのありかを表示するコマンドだ（**画面20**）。システム中に存在するファイルをデータベース化し、これ参照するのでfindコマンドよりもはるかに高速な検索ができる。ただし、検索用データベースが更新されたあとに作成したファイルは見つけられない。多くのディストリビューションでは、データベースを毎日更新するように設定されている（Red Hatなどでは、午前4時2分に更新する）。

slocateは、locateのセキュリティ強化版となっているが、locateはslocateへのシンボリックリンクになっていることが多い。

which /usr/bin

コマンドのフルパスを表示する

whichコマンドは、コマンド検索パス（環境変数PATHに設定されたパス）の中から、指定されたコマンドを探し、その場所をフルパスで表示する。これによって、コマンドを実行する際に呼び出される実行ファイルを調べることができる。オプションを付けないで実行すると、コマンド検索パスの優先順位が高いものから1つだけを表示するが、-aオプションを付けるとマッチしたものすべてを表示する（**画面21**）。

引数なし	オープンされているソケットの一覧を表示する
-r	カーネルのルーティングテーブルを表示する
-g	マルチキャストグループメンバーシップ情報を表示する
-i	ネットワークインターフェイスの状態を表示する
-M	IPマスカレードされた接続を表示する
-s	各プロトコルの統計情報の一覧を表示する

表18　netstatが出力する情報

```
$ nslookup www.ascii.co.jp
Server:  ns.ascii-linux.com
Address:  192.168.102.15

Non-authoritative answer:
Name:    at2.ascii.co.jp
Address:  210.140.231.23
Aliases:  www.ascii.co.jp

$ nslookup 202.239.88.250
Server:  ns.ascii-linux.com
Address:  192.168.102.15

Name:    flsvr00.ascii-linux.com
Address:  202.239.88.250
Aliases:  250.88.239.202.in-addr.arpa
```

画面18　名前→IPアドレス、IPアドレス→名前の解決をする

```
; <<>> DiG 8.3 <<>> ascii.co.jp
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; QUERY SECTION:
;;      ascii.co.jp, type = A, class = IN

;; AUTHORITY SECTION:
ascii.co.jp.  1D IN SOA  ascns1.ascii.co.jp. postmaster.ascii.co.jp. (
                                 200106262       ; serial
                                 2H              ; refresh
                                 30M             ; retry
                                 5w6d16h         ; expiry
                                 1D )            ; minimum
;; Total query time: 34 msec
;; FROM: misa.localdomain to SERVER: default -- 202.232.171.25
;; WHEN: Tue Jul 24 11:59:18 2001
;; MSG SIZE  sent: 29  rcvd: 83
```

画面19　digの出力例

man /usr/bin

オンラインマニュアルを参照する

UNIX系のOSには、コマンドや関数のオンラインマニュアルが用意されている。manコマンドは、これらのマニュアルを読むためのコマンドだ。manコマンドの書式は次のようになっている。

man ［オプション］［セクション］コマンド名

オンラインマニュアルは、ジャンルごとにセクション分けされている（**表19**）。特定のジャンルものを読みたいときは“セクション”で指定する。たとえば、passwdのようにコマンドと設定ファイルが同じ名前の場合、設定ファイルの書式を見たければ、man 5 passwdとする。“オプション”については、man manを実行して確認してほしい。

Vine Linuxなどでは、日本語のマニュアルを参照するjmanコマンドも用意されている。

whatis /usr/bin

whatisデータベースを検索し、コマンドの1行説明を表示する

whatisコマンドは、manページの1行説明を集めたデータベースからキーワードを検索し、コマンドや関数の1行説明を表示する（**画面22**）。データベースの更新は、/usr/sbin/makewhatisコマンドで行う。

セクション	内容
1	ユーザーコマンド
2	システムコール（カーネルが提供する関数）
3	ライブラリ関数
4	/dev ディレクトリのスペシャルファイル
5	ファイルの書式
6	ゲーム
7	その他
8	システム管理用ツール
9	Linux 独自のカーネルルーチン用のドキュメンテーション
n	新しいドキュメンテーション
o	古いドキュメンテーション
l	独自のシステムについてのローカルなドキュメンテーション

表19　セクションの分類

info /usr/bin

infoフォーマットのファイルを表示する

GNUのツールには、infoフォーマットのドキュメントが用意されているものが多い。これらのドキュメントは階層構造をとっており、infoコマンドやemacsなどを使って読むことができる。infoコマンドをオプションなしで起動すると、トップレベルのメニューが表示されるので、読みたい項目を選択する。また、項目を指定して起動することもできる。たとえば、

```
$ info emacs
```

とすると、トップレベルのメニューにあるemacsの項目を表示する。

su /bin

ほかのユーザーのユーザーIDを使用してシェルを起動する

suは、ほかのユーザーのユーザーIDを用いて、新たにシェルを起動するコマンドだ。ログインし直さずにほかのユーザーの環境を利用することができる。

ユーザー名はコマンドラインから指定するが、省略した場合は、rootになる。このように、suコマンドを使うとデフォルトではroot（super user）になるので、「Super User」の略だと思われがちだが、実際は「Substitute User」の頭文字である。

シェル変数などの環境は、元のユーザーのものがそのまま

```
$ locate mice
/dev/input/mice
/usr/share/pixmaps/gnobots2/mice.png
```

画面20　ファイル名に“mice”を含むファイルを検索する

```
$ which -a test
/usr/bin/test
~/bin/test
```

画面21　whichの実行例
ホームディレクトリのbinディレクトリに自作のtestコマンドを置いたが、これは/usr/bin/testよりも優先順位が低いのでフルパス指定をしないと実行できない。

```
$ whatis for
for                  (n)  - ``For'' loop
for [builtins]       (1)  - bash built-in commands, see bash(1)
```

画面22　whatisは複数のセクションにまたがるキーワードを検索できる

引き継がれ、カレントディレクトリも変わらない。たとえば、管理用コマンドを利用するためにsuでrootユーザーになっても、コマンドパスは一般ユーザーのままなので、/sbinや/usr/sbinにあるコマンドを入力しても、「command not found」などとエラーメッセージが返ってきてしまう。このような場合は、「－」「-l」「--login」のいずれかのオプションを指定する。

```
$ su -
```

すると通常のログインと同様の処理が行われるため、最初からrootユーザーとしてログインしたのと同じ環境が得られる。もちろん、カレントディレクトリも/rootに変更されている。

passwd /usr/bin

ユーザーパスワードを変更する

passwdコマンドをオプションなしで実行すると、自分のログインパスワードを変更できる。この場合、現在のパスワードをたずねられるので、正しいパスワードを入力してから新しいパスワードを設定する。

一般ユーザーは自分のパスワードしか変更できないが、rootユーザーは指定したユーザーのパスワードを変更できる。このほか、-lオプションでアカウントをロックしたり、-uオプションでロックを解除することも可能だ。-gオプションは、グループパスワードを変更する際に用いる。

printenv /usr/bin

環境変数を表示する

引数を付けずに実行すると、現在設定されている環境変数とその内容が表示される。調べたい環境変数を引数として指定すると、その内容が表示される。

```
$ printenv LANG
ja_JP.eucJP
```

locale /usr/bin

locale情報を表示する

localeとは、プログラムにハードコーディングすべきでない、言語や文化に特有の情報のことで、これには、使用する言語や日付のフォーマット、ソート順などが含まれる。最近のLinuxではlocaleをサポートすることで、多くの言語や文化に対応している。localeコマンドは、localeに関する環境変数の内容を表示する。日本では、“ ja_JP.eucJP ”というlocaleを使うことが多い。

-aオプションを付けて実行すると、そのシステムで利用できるlocaleの一覧が表示される。

rpm /bin

RPMパッケージを利用する

rpm（Red Hat Package Manager）は、Red Hat系のディストリビューションで用いられているパッケージマネージャだ。RPM形式のパッケージは、プログラムの配布形態としてかなり普及してきており、最近では、Red Hat系以外のディストリビューションや、ほかのOSでもrpmコマンドを持つものがあるほどだ。

RPMパッケージのインストール、アンインストール、クエリーをはじめとして、ソースRPMパッケージからバイナリパッケージを作成するビルドなど、非常に多くの機能を持つ。

プログラム開発を行わないユーザーが主に利用すると思われるオプションは、**表20**の通りだ。

date /bin

システムの日付と時刻を、表示または設定する

dateは、システムクロックの日付と時刻を表示するコマンドだ。また、新たに設定することも可能だ。

引数なしでdateコマンドを実行すると、localeに応じてさまざまな形式で現在の日付と時刻が表示される。

```
$ date
Wed Jul 18 11:27:10 JST 2001
```

表示形式を変更するには、「＋」に続けて文字列やフィールド記述子（**表21**）と呼ばれる記号を指定する。

```
$ date +"Today is %b %d, %Y"
Today is Jul 17, 2001
```

引数の先頭が「＋」以外の時は、引数を日付と時刻とみな

して、システムクロックを設定し直す。書式は、**表22**の通りだ。この書式にしたがってすべて数字で記述する必要がある。また、システムクロックを設定できるのは、root権限を持ったユーザーに限られる。

uname

/bin

システムに関する情報を出力する

unameコマンドは、現在動作しているマシンに関するさまざまな情報を出力する。以下のオプションを選択可能だ。

-s OSの名前（デフォルト）
-n ホスト名
-r OSのリリース番号
-v OSのバージョン
-m マシンの種類（CPU）

また-aオプションを指定すると、上記の情報がすべて出力される。

```
$ uname -a
Linux test.example.com 2.4.2-2 #1 Sun Apr 8 20:21:34
EDT 2001 i686 unknown
```

cal

/usr/bin

カレンダーを表示する

calコマンドは、テキスト形式のカレンダーを出力するコマンドだ。引数なしの場合は、今月のカレンダーだけが表示される。今年1年分のカレンダーを表示するには、-yオプションをつければいい。数字の引数を1つ指定すると、その値を西暦とみなして、1年分のカレンダーが表示される。

```
$ cal 6 2001
```

%a	省略形の曜日（Mon）
%b	省略形の月の名前（Jan）
%d	日
%m	月
%y	年の下2桁
%Y	年
%X	時刻（HH:MM:SS形式）
%Z	タイムゾーン

表21　dateでよく使われるフィールド記述子

```
      June 2001
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

-mオプションを指定することで、1週間が月曜日から始まるように表示できる。また-jオプションを指定すると、その年の1月1日からの日数（ユリウス日付）が表示される。

そのほか、あまり実用性はないのだが、以下のように年月を指定すると、変わったカレンダーが表示される。

```
$ cal 9 1752
   September 1752
Su Mo Tu We Th Fr Sa
       1  2 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

これは、現在使われているグレゴリオ暦と、それ以前のユリウス暦の切り替わりが1752年9月にあったとみなしているためだ。

インストール／アンインストール	
-i	パッケージをインストールする
-U	古いパッケージがあれば、削除してからパッケージをインストールする
-e	パッケージをアンインストールする
クエリー	
-qi	パッケージ情報を表示する
-ql	パッケージに含まれるファイルの一覧を表示する
-qp	インストールされていないパッケージに関するクエリーを表示する
-qf	ファイルを指定すると、そのファイルを含んでいるパッケージ名を表示する

表20　rpmの基本的なオプション

mmddhhmmyyyy.ss	
mm	月
dd	日
hh	時
mm	分
yyyy	年（2または4桁）（省略可能）
ss	秒（省略可能）

表22　dateコマンドで用いる日付／時刻の書式

md5sum

/usr/bin

MD5メッセージダイジェストを計算／チェックする

MD5は、メッセージダイジェスト関数と呼ばれるもののひとつで、与えられたデータから128ビットのフィンガープリント（指紋）を計算する。この関数は、元のデータが1ビット違うだけでもまったく異なった結果を返すという特徴を持ち、データの改変などを検知するために考案された。FTPサイトから入手したファイルが壊れていないかを調べたり、ファイルの改ざんが行われていないかをチェックするのに使われる（**画面23**）。

script

/usr/bin

端末に表示した内容をファイルに保存する

どのようなコマンドを実行し、どのような結果が表示されたのかを保存したいときにはscriptコマンドを使おう。

scriptを引数なしで実行すると、端末への表示内容をtypescriptというファイルに記録し始める。exitを実行することで記録は終了する。引数としてファイル名を指定することも可能だ。また、-aオプションを付けて実行すると、ファイルを上書きせずに追加記録する。

kon

/usr/bin

日本語を表示できるコンソールエミュレータ

konを使うとコンソール画面に日本語を表示することができる。通常はオプションを付けずに起動すればよい。

フレームバッファ（FB）コンソールでは利用できないので、FBコンソール環境ではJFBTERM（http://www3.justnet.ne.jp/~nmasu/linux/jfbterm/indexn.html）を利用しよう。

procinfo

/usr/bin

システムの状態を/procより収集し表示する

/procディレクトリには、メモリの利用率やCPU負荷をはじめとするシステムの状況を収めた疑似ファイルが収められている。procinfoコマンドは、これらの情報を見やすく整形して表示する（**画面24**）。

```
$ md5sum orig.iso bake.iso
45024df19c7f1cb26af77231a221b3b0  orig.iso
1fa64a5f00003df9256895a00d2fb463  bake.iso
```

画面23　md5sumでファイルをチェックする
約640Mバイトのファイルorig.isoと、その内容を1ビットだけ変更したbake.isoのMD5は全然違う。

```
$ procinfo
Linux 2.4.2-2 (root@porky) (gcc 2.96 20000731 ) #1 1CPU [lmpc12]

Memory:      Total        Used        Free      Shared     Buffers      Cached
Mem:        255528      250540        4988           0       59044       63012
Swap:       265064       18536      246528

Bootup: Mon May 21 21:17:46 2001    Load average: 0.99 0.97 0.96 2/51 7973

user  :      17:45:52.30   1.2%  page in :  1821831
nice  :  57d  8:20:08.50  92.9%  page out:  2207569
system:       6:06:15.47   0.4%  swap in :   110541
idle  :   3d 10:05:54.53   5.5%  swap out:    67002
uptime:  61d 18:18:10.79         context :1464918016

irq  0: 533629080 timer                 irq  8:         1 rtc
irq  1:     14940 keyboard              irq  9:     34296 usb-uhci, usb-uhci
irq  2:         0 cascade [4]           irq 11:   1527444 eth0
irq  3:         1                       irq 12:    109948 PS/2 Mouse
irq  4:         1                       irq 14:   1419126 ide0
irq  5:         0 mga@PCI:1:0:0         irq 15:  11662139 ide1
irq  6:        79
```

画面24　procinfoの実行例

管理系コマンド

/sbin、/usr/sbinディレクトリには、システム管理に関連するコマンドが収録されている。このため、ほとんどのコマンドはrootユーザー権限が必要となるものだ。逆にいえば、rootユーザー必須のコマンドばかりということになる。

arp

/sbin

ARPキャッシュの表示と操作

ARP（Address Resolution Protocol）とは、TCP/IPで通信する際に必要となるMACアドレス（ハードウェアアドレス）とIPアドレスを相互に変換するためのプロトコルだ。

TCP/IPでは、データ（イーサネットフレーム）を通信先のマシンに送信する場合、送信先のMACアドレスが必要となる。しかし通常、送信先のIPアドレスはわかっていても、MACアドレスは不明である。そこで、IPアドレスを元にARPリクエストをブロードキャストし、送信先のマシンからMACアドレスを返信してもらう（つまり、「該当のIPアドレスを持っている機器はMACアドレスを教えてくれ」というリクエストを送る）。これによって相手先のMACアドレスを知ることができるのだが、送信の度にARPリクエストをブロードキャストするのは効率が悪いため、一度通信した通信先のMACアドレスとIPアドレスの対応をキャッシュしておく。これが、ARPキャッシュだ。

このARPキャッシュの表示と操作を行うためのコマンドがarpだ。

もっとも、ARPキャッシュは一定時間を過ぎると自動的にクリアされるため、ARPキャッシュの内容を変更しなければならないことはほとんどない。一般的に使われるのは、同一のIPアドレスに違う機器を接続した場合などにARPキャッシュの内容を強制的にクリアする場合などだ。それ以外では、相手先のMACアドレスを調べる場合や、どのMACアドレスにどのIPアドレスが割り当てられているかを調べる場合に使用する。

ARPキャッシュを表示するには、単にarpコマンドを実行すればよい。

```
# arp
```

このとき、**画面1**のように表示される「Address」はマシン名、「HWaddress」はMACアドレスだ。

特定の機器の情報だけを表示させる場合は-aオプションに続けてホスト名またはIPアドレスを指定する。

```
# arp -a hogehoge.ascii.co.jp
```

ARPキャッシュからエントリを削除する場合は、-dオプションに続けて、ホスト名またはIPアドレスを指定する。

```
# arp -d hogehoge.ascii.co.jp
```

なお、ARPキャッシュを表示するだけなら一般ユーザーも実行することが可能だ。

chkconfig

/sbin

起動時のサービス（デーモン）の設定と表示

システム起動時の各サービスの起動／無効を設定するのがchkconfigコマンドだ。各ランレベルに対して、起動するサービスを設定することができる。ただし、このコマンドはRed Hatツールなので、Red Hat系ディストリビューションでのみ利用できる。

現在設定されているサービスの設定を確認する場合は--listオプションに続けてサービス名を指定する。

```
# chkconfig --list dhcpd
```

```
# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
hogehoge.ascii.co.jp     ether   00:11:22:33:44:55   C                     eth0
foobar.pb.ascii.co.jp    ether   00:55:44:33:22:11   C                     eth0
```

画面1　arpコマンドの実行結果

表示された0〜6までの数字が各ランレベルにおけるサービスの起動／無効（オンが起動）を意味する。

オプションを指定しなければすべてのサービスの設定を表示することができる。

サービスの起動／無効の設定は、--levelオプションに続けてランレベルとサービス名、起動（on）または無効（off）を指定する。たとえば、dhcpdをランレベル3と5で起動する場合は次のように指定する（**画面2**）。

```
# chkconfig --level 35 dhcpd on
```

--levelオプションを指定しない場合は、ランレベル3、4、5におけるサービスの有効／無効が設定される。たとえば、次のように指定すると、ランレベル3、4、5でdhcpdが無効になる。

```
# chkconfig dhcpd off
```

また、on/offの代わりにresetを指定すると、デフォルトの設定値（initスクリプトに記述されている）に設定される。

```
# chkconfig dhcpd reset
```

なお、現在の設定情報を表示するだけなら一般ユーザーも実行することが可能だ。

clock/hwclock /sbin

CMOSクロック／システムクロックの設定

Linuxでは2つの時計が動いている。1つはカーネル内部で動くシステムクロック（カーネルクロック）、もう1つはマシンの内蔵時計であるCMOSクロック（ハードウェアクロック）だ。これら2つの時計は別々に時を刻んでいるが、残念ながら誤差があるためどちらの時計も少しずつずれていく。

clockコマンドはこれら2つの時計を同期するためのコマンドだ。システムクロックをCMOSクロックの時計に合わせる場合は、-sオプションを指定する。逆に、CMOSクロックをシステムクロックの時計に合わせる場合は、-wオプションを指定する。

```
# clock -s
# clock -w
```

何もオプションを指定しないか、-rオプションを指定した場合はCMOSクロックに設定されている時刻を表示する。

時計を操作する同様のコマンドとして、hwclockコマンドがある。hwclockコマンドはclockコマンドと同等の機能を持っているが、さらに任意の時間に合わせる機能を持っている。

現在のCMOSクロックの時刻を表示するには、--showオプションを指定する。

```
# hwclock --show
```

CMOSクロックの時刻を設定するには、--setオプションに続けて、--dateオプションを指定する。たとえば、2001年9月10日22時30分00秒に設定するには、

```
# hwclock --set --date="09/10/2001 22:30:00"
```

と指定する。指定できる時刻の形式はdateコマンドと同じだ。

clockコマンドと同様に、システムクロックとCMOSクロックの同期を取ることもできる。システムクロックをCMOSクロックの時計に合わせる場合は、--hctosysオプションを指定する。CMOSクロックをシステムクロックの時計に合わせる場合は、-systohcオプションを指定する。

なお、clock、hwclockコマンドともに、現在の時刻情報を表示するだけなら一般ユーザーも実行することが可能だ。

depmod/insmod/lsmod/modprobe/rmmod /sbin

モジュールの組み込み／削除／操作

Linuxのカーネルは、あとから機能を追加できるようにカーネルモジュールが用意されている。カーネルモジュールは、カーネルを小さくするのに役立つ。つまり、不必要なデバイスドライバなどはカーネルに組み込まず、必要になった際にシステムに組み込めるように設計されているのだ。カーネルモジュールがなければ、新しいハードウェアを追加するたびに必要なデバイスドライバを組み込んだカーネルを再構築しなければならなくなる。ちょうど、Windowsのデバイスドラ

```
# chkconfig --list dhcpd
dhcpd           0:オフ  1:オフ  2:オフ  3:オフ  4:オフ  5:オフ  6:オフ
```

画面2　chkconfigコマンドの実行結果

イバと同じようなものだと考えればいいだろう。

モジュールを組み込むには、insmodコマンドにモジュールを指定する。

```
# insmod pcmcia_core
```

このとき、insmodコマンドはモジュールのデフォルトディレクトリからモジュールを検索する。モジュール名にパスを付ければそのモジュールを組み込むこともできる。

insmodコマンドに-kまたは、--autocleanオプションを付けると、1分間使われなかったモジュールはカーネルによって自動的に削除される。

組み込まれているモジュールを表示するには、lsmodコマンドを使う。オプションは必要ない（**画面3**）。

組み込んだモジュールを削除するにはrmmodコマンドを使う。オプションに削除するモジュールを指定すればいい。

```
# rmmod pcmcia_core
```

ただし、使用中や参照中のモジュールは削除することができない。

モジュールには依存関係があり、モジュール内から別のモジュールを呼び出すこともある。このような場合は、depmodコマンドを使って依存関係のリストを作成する。

```
# depmod -a
```

-aは依存関係のリストを作成する際、/etc/modules.confに記述されているオプションやエイリアスを読み込むオプションだ。-Aオプションを指定した場合はタイムスタンプを参照し、変更されたファイルのみの依存関係を更新する。

通常は、-aオプションを指定すればいいだろう。

依存関係のリストは/lib/modules/＜カーネルのバージョン＞ディレクトリにmodules.depとして作成される。

```
# lsmod
Module                  Size  Used by
pcmcia_core            43456   0 (autoclean) (unused)
agpgart                23168   0 (autoclean)
3c59x                  25312   1 (autoclean)
  :
  :
```

画面3　lsmodコマンドの実行結果

モジュールを組み込む場合は、modprobeコマンドにモジュール名を指定する。

```
# modprobe pcmcia_core
```

insmodコマンドと同じようだが、依存関係リストを参照して必要なモジュールをすべて組み込む点が異なる。modprobeコマンドのほうが便利なので、通常はこちらを使うことが多いだろう。なお、-rオプションを指定すれば、組み込んだモジュールを削除することができる。

```
# modprobe -r pcmcia_core
```

なお、システムを起動するたびにモジュールを組み込むには、/etc/modules.confに別名を登録する必要がある。

dump/restore **/sbin**

ファイルシステムのバックアップとリストア

ファイルシステムのバックアップとリストア（復元）を行うコマンドがdumpとrestoreだ。tarコマンドやcpioコマンドでバックアップを取ることも可能だが、dumpコマンドは差分バックアップを取る機能がある。たとえば、フルバックアップを取ったあと、変更があったものだけ（フルバックアップした日時よりも新しいタイムスタンプのファイル）をバックアップするといった機能だ。これを使えば、毎日のバックアップを手軽に取ることができる。

本来、dumpコマンドはテープデバイスに対してファイルシステムをバックアップする。しかし、ハードディスクなどのファイルシステムにバックアップを作成することも可能だ。ただし、/ディレクトリ以下をすべてバックアップする場合は、バックアップを別のデバイス（パーティション）に作成しなければならない。これは、バックアップしているファイル自体もバックアップしようとしてしまうからだ。

パーティションを分割しているのであれば、別のパーティションにバックアップを作成すれば問題ないが、パーティションを分割していない場合は、ディレクトリ単位でバックアップしよう。

たとえば、/homeディレクトリ以下を/tmp/homebackupにバックアップするには次のように指定する。

```
# dump 0u -f /tmp/homebackup /usr
```

オプションに指定した0はダンプレベルだ。ダンプレベルは0〜9までの数字を指定する。ここで指定した0はすべてのファイルをバックアップすることを意味する。

差分をバックアップするには、順に数字を大きくしていけばよい。

たとえば、毎週日曜日にフルバックアップを取り、毎日差分バックアップを取る場合は、日曜日が0、月曜日が1、火曜日が2……と指定するようにする。

先の例で指定したuオプションは/etc/dumpdatesにバックアップした日付を追加するオプションだ。ダンプレベルを指定した場合、dumpdatesに記載されているバックアップ日時よりも新しいファイルが差分バックアップ時にバックアップされる。

なお、dumpコマンドを実行する際はシングルユーザーモードで行うほうがいいだろう（特にフルダンプ時）。

dumpコマンドで作成したバックアップをリストアするには、restoreコマンドを使う。

ファイルシステムに作成したバックアップをリストアする場合は、xfオプションを指定する。

```
# restore xf /tmp/homebackup
```

このコマンドを実行すると、カレントディレクトリにバックアップをリストアする。特定のファイルやディレクトリのみをリストアする場合は、さらにファイル名やディレクトリ名を指定すればよい。

```
# restore xf /tmp/usrbackup hogehoge
```

なお、差分バックアップをリストアするには、ダンプレベル0から順にリストアしていく必要がある。

fdisk /sbin

パーティションテーブル操作

fdiskコマンドは、パーティションテーブルを操作するためのコマンドだ。起動オプションにデバイス名を指定して起動した場合、メニュー形式の対話型インターフェイスを使ってパーティションテーブルを操作することができる。たとえば、IDEハードディスクのユニット0を操作したい場合は、

```
# fdisk /dev/hda
```

として起動する。以降はメニュー形式で表示され、パーティションを操作することが可能だ。

パーティションテーブルの情報を参照するだけなら、-lオプションを指定すればメニュー形式ではなく標準出力に出力される。

fsck /sbin

ファイルシステムのチェックと修復

ファイルシステムに異常（矛盾）がないかをチェックしたり、不用意な電源断などによって発生したファイルシステムの異常を修復するためのコマンドがfsckだ。

fsckコマンドはデバイス名を指定して起動する。

```
# fsck /dev/hda1
```

デバイス名の代わりに、/や、/usrなどといったマウントポイントを指定することもできるが、システムが書き込み中のファイルシステムを修復するとファイルシステムを破壊する危険があるので、できるだけアンマウントした状態でチェックするようにしよう。

/パーティションの異常など、どうしてもマウントした状態でファイルシステムの修復をしなければならないときは、ファイルシステムをバックアップ後、シングルユーザーモードにしてから行うほうがいいだろう。

fuser /sbin

ファイルやソケットを使用しているプロセスの表示

誰がそのプログラムやファイルを使っているのかを表示するのがfuserコマンドだ。たとえば、CD-ROMをアンマウントしようとしたら、誰かが使用中でアンマウントできなかった場合などに、誰が使っているのかを調べることができる。

fuserコマンドは、ファイルやデバイス名などを指定する。

```
# fuser -v /mnt/cdrom
```

```
# fuser -v /mnt/cdrom
                     USER        PID ACCESS COMMAND
/mnt/cdrom           hogehoge   2033 ..c..  gmc
                     root      kernel mount  /mnt/cdrom
```

画面4　fuserコマンドの実行結果

-vは冗長表示で、実行しているユーザー名なども一緒に表示するためのオプションだ。表示される内容は**表1**のとおりだ。**画面4**の例では、/mnt/cdromをユーザーhogehogeが使用していて、プロセスIDは2033だということと、カーネルがマウントしているということがわかる。

また、-kオプションを指定すればそのプロセスにKILLシグナルを送信して、プロセスを終了させることができる。

```
# fuser -k /mnt/cdrom
```

ただし、これは危険な方法なので、どうしてもプロセスを終了しなければならないような場合にのみ使用するようにしたほうがいいだろう。

なお、誰がそのプログラムやファイルを使っているのかを表示するだけなら一般ユーザーも実行することが可能だ。

halt/reboot/shutdown /sbin

システムの停止

システムを停止したり、再起動する場合に使うのがこれらのコマンドだ。システムを停止する場合は、haltコマンド、システムを再起動する場合はrebootコマンドを実行する。

```
# halt
# reboot
```

現在では、halt、rebootコマンドはshutdownコマンドを呼び出す。

shutdownコマンドはinitコマンドを呼び出してシステムを停止する。通常は、-h（システムの停止）か、-r（再起動）オプションに続けて、システムを停止する時間を指定する。時間は絶対時間（hh:mm）形式か、何分後（+m）で指定する。また、「now」を指定した場合はただちにシステムを停止する。

項目	意味
USER	プロセスの所有者
PID	プロセス番号またはkernel（カーネルがアクセスしている場合）
ACCESS	ファイルアクセスの形式
	c　カレントディレクトリとして使用中
	e　実行中
	f　オープン中のファイル
	r　ルートディレクトリとして使用中
	m　mmapされたファイルか共有ライブラリ
COMMAND	実行されているコマンド

表1　fuserコマンドで表示される内容

```
# shutdown -h +5
# shutdown -r now
```

PCがAPMをサポートしていれば、-hオプションを指定した場合は電源が切れる。

shutdownコマンドを実行時、各端末にメッセージを送信することができる。たとえば、「shutdown at 22:00 pls. logout」というメッセージを送信したい場合は次のように実行する。

```
# shutdown -r 22:00 shutdown at 22:00 pls. logout
```

なお、-cオプションを指定することでshutdownコマンドを取り消すことができる。

hdparm /sbin

ハードディスクパラメータの設定

hdparmは、ハードディスクやCD-ROMのパラメータの設定や表示を行うコマンドだ。コマンドはデバイス名を指定して実行する。

たとえば、hdaのパラメータを表示する場合は、次のように指定する。

```
# hdparm /dev/hda
```

-iオプションを指定すると、ハードディスクの詳しい情報が表示される。

ハードディスクパラメータを変更する場合は、**表2**にあるオプションを指定する。たとえば、マザーボードとハードディスクユニットがUltraDMA/66に対応しているのであれば、

オプション	意味
-c0	32ビットI/Oサポート無効
-c1	32ビットI/Oサポート有効
-c3	32ビットI/Oサポート有効（スペシャルsyncシーケンス）
-d1	DMA転送有効
-d0	PIO転送有効
-i	ハードディスクの情報の表示
-t	ベンチマークテスト（リード）
-T	ベンチマークテスト（キャッシュリード）
-X66	UltraDMA/33有効（-d1オプションと併用）
-X68	UltraDMA/66有効（-d1オプションと併用）
-X69	UltraDMA/100有効（-d1オプションと併用）

表2　hdparmコマンドの主なオプション

```
# hdparm -c1 -d1 -X68 /dev/hda
```

とすればよい。

ただし、ハードウェアが対応していないオプションなどを設定した場合や、ここで紹介していないオプションなどを設定した場合、ハードディスクユニットそのものを壊してしまったり、ファイルシステムを破壊してしまう可能性もあるので注意して使う必要がある。特に、-Xオプションは危険なので注意して使うようにしたい。

hdparmで設定した値は、システムを再起動した際にクリアされてしまうので、/etc/rc.d/rc.localの最後などにコマンドを記述しておこう。

ifconfig /sbin

ネットワークインターフェイスの設定

ifconfigは、カーネルに認識されているネットワークインターフェイスの設定を行うコマンドだ。単にコマンドを実行した場合は、すべてのネットワークインターフェイスの情報を表示する（**画面5**）。

ネットワークインターフェイスにIPアドレスを割り当てる場合は、IPアドレスを指定する。このとき、netmaskオプションを指定すれば、ネットワークマスクも設定できる。たとえば、ネットワークインターフェイスeth0にIPアドレス192.168.0.1、ネットマスク255.255.255.0を割り当てる場合は、

```
# ifconfig eth0 192.168.0.1 netmask 255.255.255.0
```

と指定する。

また、ifconfigコマンドはネットワークインターフェイスの有効／無効の切り替えを行うことができる。オプションはネットワークインターフェイス名（通常は、eth0など）と、up（有効）またはdown（無効）を指定する。たとえば、eth0を無効にする場合は、

```
# ifconfig eth0 down
```

と指定する。

なお、ネットワークインターフェイス情報を表示するだけなら一般ユーザーも実行することが可能だ。

init/telinit /sbin

ランレベルの変更

システムの動作環境を規定するランレベルは0から6までとS（またはs）がよく使われる。これ以外にも、a、b、cなどを使うこともできる。ランレベルの0はシステムの停止、1とSはシングルユーザーモード、6は再起動に割り当てられている。これ以外のランレベルについては、ディストリビューションごとに異なっているが、おおむね**表3**のようになっている（Red Hat Linuxの場合）。

このランレベルを変更するのが、init、telinitコマンドだ。どちらも同様のコマンドだが、telinitコマンドはinitコマンドを呼び出す。

ランレベルを5に変更したい場合は、

```
# init 5
# telinit 5
```

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:60:97:84:F8:F0
          inet addr:192.168.0.1  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:259 errors:0 dropped:0 overruns:0 frame:0
          TX packets:94 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:11 Base address:0xe800

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

画面5　ifconfigコマンドの実行結果

として実行する。

ランレベルに0を指定すれば、システムが停止する。

ランレベルのS以外は/etc/inittabに記述されたラベルなので、7〜9といったラベルを定義すれば独自のランレベルを作成することも可能だ。

なお、現在のランレベルを調べるには後述のrunlevelコマンドを使う。

runlevel /sbin

ランレベルの表示

現在のランレベルを調べるのがrunlevelコマンドだ。runlevelコマンドにはオプションは存在しない。単にコマンドを実行すれば、現在のランレベルと1つ前のランレベルが表示される。

```
# runlevel
3 5
```

この例では、1つ前のランレベルが3、現在のランレベルが5だという意味だ。

ldconfig /sbin

実行時リンクの結合関係の作成

プログラムのライブラリは、静的リンク（スタティックリンク）と、実行時リンク（ダイナミックリンク）の2つがある。静的リンクは、ライブラリをプログラムそのものに含める形式で、別途ライブラリを用意する必要はないが、プログラムサイズが大きくなる。

一方、実行時リンクはプログラムサイズが小さくなるが、別途ライブラリが必要となる。多くのプログラムから使われるようなライブラリは実行時リンクとして用意されることが多い。

ランレベル	動作
0	システムの停止
1	シングルユーザーモード
2	シングルユーザーモード（NFSなし）
3	マルチユーザーモード（テキストベース）
4	未使用
5	マルチユーザーモード（X Window Systemベース）
6	再起動
S（またはs）	シングルユーザーモード

表3　Red Hat Linuxにおけるランレベル

ldconfigコマンドは、ライブラリの結合関係を作成するプログラムだ。通常は、/etc/ld.so.confに指定された情報を元に、必要な設定を行う。通常は結合関係を作成する必要はないが、独自に実行時ライブラリを用意した場合に実行する。オプションは特に必要ない。

mkfs /sbin

ファイルシステムの構築

ファイルシステムの作成を行うのがmkfsコマンドだ。ファイルシステムを作成するパーティションと、ファイルシステム形式を指定する。作成できるファイルシステムはシステムによって異なるが、ext2やreiserfsなどが指定できる。

```
# mkfs -t ext2 /dev/hda2
```

mkfsコマンドは、実際には各ファイルシステムごとの専用プログラムが呼ばれる。このため、システムに用意されていないファイルシステムは作成することができない。

traceroute /sbin

ホストへのネットワーク経路の表示

LANなどで、ほかの経路へルーティングされていない場合を除き、インターネットでは通常、さまざまなホストを経由して相手先ホストと通信している。この、ネットワーク上の相手先ホストへどのように接続されているかを表示するのがtracerouteコマンドだ。

tracerouteコマンドには、相手先ホストのマシン名かIPアドレスを指定する。

```
# traceroute www.hogehoge.co.jp
```

コマンドを実行すると、正常に相手先ホストと通信できた場合は**画面6**のように表示される。

表示される最初の数字はホップ数と呼ばれるもので、相手先ホストまで到達するまでに中継したルータの数を意味している。続く2つのアドレスは中継したルータ名とIPアドレスだ。IPアドレスからルータ名を解決できなかった場合は、2つともIPアドレスが表示される。最後の3つの数字はホストに到達するまでの時間だ。この数字が大きくなっている中継点があれば、その中継点への接続に時間がかかっているとい

うことだ。

なお、ホップ数が多くなればなるほど、相手先ホストまでのネットワーク的な距離が遠いということを意味する。

tracerouteコマンドはデフォルトで30ホップまで表示することが可能だが、これは-mオプションにホップ数を指定することで変更できる。

adduser/useradd /usr/sbin

新規ユーザーアカウントの作成

新規のユーザーアカウントの作成と、既存のユーザーアカウントの変更を行うのが、adduser、useraddコマンドだ。どちらのコマンドも実際は同じものだ。

新規のユーザーアカウントを作成する場合は、ユーザー名を指定する。このとき、/etc/passwdに書かれるコメント（多くはフルネーム）や、所属するグループを指定することができる。たとえば、Hoge Hogeoさんのアカウントをhogehogeで作成し、グループをusersに所属させるには、

```
# useradd -c "Hogeo Hoge" -g users hogehoge
```

と指定する。作成されたアカウントはパスワードが設定されていないので、passwdコマンドなどを使ってパスワードを設定する必要がある。

groupadd /usr/sbin

新規グループの作成

新たなグループを作成するのがgroupaddコマンドだ。作成するグループ名を指定するだけでいい。たとえば、linuxmagというグループを作成するのであれば、

```
# groupadd linuxmag
```

とする。このとき、-gオプションに続けてグループIDを指定することもできる。

```
# groupadd -g 758 linuxmag
```

グループIDは、ほかのグループIDと重複しないように/etc/groupファイルを見て設定する必要がある。

groupdel /usr/sbin

グループの削除

既存のグループを削除するのがgroupdelコマンドだ。たとえば、linuxmagというグループを削除するには次のように指定する。

```
# groupdel linuxmag
```

なお、ユーザーが属しているグループや、グループに属しているファイルが存在してはならないので、グループを削除する前に、ユーザーがそのグループに属していないか、グループに属しているファイルがないかを調べておく必要がある。

userdel /usr/sbin

ユーザーアカウントの削除

既存のユーザーアカウントを削除するのがuserdelコマンドだ。たとえば、ユーザーアカウントhogehogeを削除する場合は次のように指定する。

```
# userdel -r hogehoge
```

-rオプションは、ユーザーのホームディレクトリと、メールスプールも同時に削除する。ただし、そのユーザーが所有していたファイルやcronで設定しているコマンドなどは削除されないので、削除するユーザーが所有権を持つファイルがある場合は、別途削除するか、所有権をほかのユーザーに変更する必要がある。削除したユーザーを同名で作成しなおしたとしても、別のユーザーとして扱われるので（ユーザーIDが異なる）、削除する際は注意が必要だ。

```
# traceroute www.hogehoge.co.jp
traceroute to www.hogehoge.co.jp (123.123.123.123), 30 hops max, 38 byte packets
 1  router (111.122.133.144)  1.768 ms  1.330 ms  1.227 ms
 2  uso.foobar.ne.jp (123.124.125.126)  17.641 ms  15.620 ms  15.986 ms
 3  123.123.111.222 (123.123.111.222)  26.481 ms  25.058 ms  25.482 ms
 4  www.hogehoge.co.jp (123.123.123.123)  27.351 ms *  26.681 ms
```

画面6　tracerouteコマンドの実行結果

bash組み込みコマンド

bashには、組み込みコマンドが用意されている。これらのコマンドは、bash内部で処理されるものだが、そのぶん基本的なコマンドが多い。なお、bashの組み込みコマンドといっても、ほとんどはそのほかのシェルでも同様に利用できる。

cd

bash

カレントディレクトリの変更

現在のカレントディレクトリを変更するのがcdコマンドだ。cdコマンドに続けて、変更するディレクトリを指定する。たとえば、/tmpディレクトリに変更するのであれば、

```
$ cd /tmp
```

とすればよい。ディレクトリは相対パスを指定することもできる。たとえば、1つ上のディレクトリにあるhogehogeディレクトリに変更する場合は、

```
$ cd ../hogehoge
```

と指定すればいい。

cdコマンドにオプションを指定しないで実行した場合や、~ を指定した場合はカレントディレクトリがホームディレクトリに変更される。

pushd/popd/dirs

カレントディレクトリの変更と保存

ディレクトリを移動したあと、元のディレクトリに戻りたいことがよくある。こういった場合はcdコマンドではなく、pushdコマンドを使うと便利だ。pushdコマンドは、カレントディレクトリを変更する際に、変更する前のカレントディレクトリを保存する。保存されたカレントディレクトリに戻るには、popdコマンドを使う。たとえば、カレントディレクトリがホームディレクトリで、カレントディレクトリを/tmpディレクトリに変更後、元のディレクトリ（ここではホームディレクトリ）に戻る場合を考えてみよう。

まず、カレントディレクトリがホームディレクトリの状態で、

```
$ pushd /tmp
/tmp/  ~
```

とすれば、カレントディレクトリが/tmpディレクトリに変更される。この際、変更する前のカレントディレクトリが保存される。次に、

```
$ popd
```

とすることで、元いたディレクトリに戻ることができる。

pushdコマンドは実行されるたびにカレントディレクトリを順に保存していき、popdコマンドが実行されると新しいカレントディレクトリから順に戻していく。つまり、/tmp、/var、/etcディレクトリへ変更した場合、popdコマンドは順に/etc、/var、/tmpディレクトリと戻っていく。

保存されているディレクトリを表示するにはdirsコマンドを使う。

```
$ dirs
/etc  /var  /tmp  ~
```

保存されているディレクトリは、一番左側が1つ前のカレントディレクトリ、次が2つ前のカレントディレクトリ……となっている。

history

bash

コマンド履歴の表示

これまで実行したコマンドの履歴を表示するのがhistoryコマンドだ。単にhistoryと入力すればこれまで実行したコマンドを表示する。

```
$ history
```

オプションに数値を指定すれば、最近実行したコマンドを

指定した数値ぶん表示する。たとえば、5回前までのコマンドを表示するには、5を指定する。

```
$ history 5
  121  ls
  122  pushd /tmp
  123  vi hogehoge
  124  popd
  125  history 5
```

左側に表示されている数値は、コマンドを実行した数でコマンドを実行するごとに増えていく。

過去のコマンドをもう一度実行するには、「!」に続けてこの数字を指定する。たとえば、121をもう一度実行するのであれば、

```
$ !121
```

とすればよい。なお、直前のコマンドのみ「!!」とするだけで実行することができる。

set/unset

bash

シェル変数の操作

現在使用しているシェル内でのみ有効なシェル変数をすべて表示するにはsetコマンドを使う。

```
$ set
```

setコマンドは、本来bashのオプションを設定するためのコマンドなのだが、オプションなしで実行した場合はすべてのシェル変数を表示する。

シェル変数に値を代入する場合は、＝を使う。たとえば、シェル変数「TERM」に「vt100-w」を代入するには次のように指定する。

```
$ TERM=vt100-w
```

unsetコマンドを使えば、設定したシェル変数を削除することができる。

```
$ unset TERM
```

export

bash

環境変数の設定

シェル変数を環境変数へエクスポートしたり、環境変数を設定したりするのがexportコマンドだ。

```
$ export TERM
```

直接環境変数を指定するには、変数名と値を＝で指定する。

```
$ export TERM=vt100-w
```

環境変数の表示については、printenvコマンドを参照のこと。

bg/fg/jobs/kill

bash

ジョブの操作

Linuxはマルチタスクであるため、プログラムをバックグラウンドで実行することができる。プログラムをバックグラウンドで実行すれば、時間のかかる処理が実行中でもすぐに別の作業を行うことができる。

プログラムをバックグラウンドで実行するには、コマンド名の後ろに&を付けて実行する。

```
$ sort hogehoge > hogehoge2.txt &
[1] 2694
```

このとき表示される[ ]で括られた数値はジョブ番号で、その後ろの番号はプロセス番号だ。

バックグラウンドでどのようなジョブが実行されているかはjobsコマンドを実行すればわかる。

```
$ jobs
[1]-  Stopped     less /var/log/messages
[2]+  Running     sort hogehoge > hogehoge2.txt
```

「Stopped」と表示されたジョブは停止中、「Running」と表示されたジョブは実行中を意味する。停止中のジョブの処理を再開するのがbgコマンドとfgコマンドだ。bgコマンドはバックグラウンドでジョブを再開し、fgコマンドはフォアグ

ラウンドでジョブを再開する。オプションには%に続けてジョブ番号を指定する。

```
$ fg %1
$ bg %1
```

ジョブ番号を指定しなかった場合は、最新のジョブ（jobsコマンドで＋記号が表示されている）を指定したことになる。

すでにフォアグランドで実行しているジョブは、Ctrl+Zで停止してからbgコマンドを使えばバックグランドで実行することができる。

バックグラウンドで実行されているジョブを削除するにはkillコマンドを使う。オプションはジョブ番号を指定する。

```
# kill %1
```

killコマンドはプロセスを指定することで、プロセスを削除することもできる。この場合、プロセスにTERMシグナルを送信する。TERM信号を送信してもプロセスを終了できないときは、KILLシグナルを送信しよう。

```
# kill -kill 2694
```

ただし、KILLシグナルを送信してプロセスを終了しようとすると、プロセスがゾンビとして残ってしまう可能性もある。KILLシグナルは最後の手段と考えよう。

alias/unalias bash

コマンドの別名定義

長いコマンドや、よく使うコマンドを短いコマンドとして別名を付けるのがaliasコマンドだ。たとえば、manコマンドにmという別名を付ければ、mと入力するだけで、manと入力したときと同じように実行できる。

```
$ alias m='man'
```

また、すでにあるコマンドと同じ名前で別名を定義することもできる。たとえば、lsコマンドでディレクトリの内容を表示するとき、必ず-laFというオプションを付けて実行したい場合は、

```
$ alias ls='ls -laF'
```

とする。以降のlsコマンドはls -laFと実行したのと同じことになる。設定されている別名を表示するには、単にaliasと入力すればよい。

別名を取りやめる場合はunaliasコマンドを使う。

```
$ unalias ls
```

echo bash

標準出力への出力

指定した文字列や、シェル変数などを表示するのがechoコマンドだ。

```
$ echo Hello! Linuxer
Hello! Linuxer
$ echo $TERM
vt100-w
```

文字列やシェル変数以外でも、シェルスクリプトが処理できるワイルドカードなども指定できる。

help bash

bash組み込みコマンドのヘルプ表示

bashの組み込みコマンドのヘルプを表示するのがhelpコマンドだ。helpコマンドに続けて、コマンドを指定する。

```
# help echo
```

コマンドを指定しない場合は、すべての組み込みコマンドの簡単な使い方を一覧表示する。

pwd bash

カレントディレクトリの表示

カレントディレクトリを表示するのがpwdコマンドだ。

```
$ pwd
/home/hogehoge
```

コマンドを実行するとカレントディレクトリが表示される。

Mozilla、Netscape、
Galeon、Opera、…etc.
Linuxで「使える」ブラウザを一挙に紹介

Webブラウザカタログ 2001・夏

新たなるスタンダードを目指す

Mozilla

最新バージョン： 0.9.2
開発プロジェクト：http://www.mozilla.org/

Mozillaは、Netscape Communicatorのオープンソース化によって産声をあげたWebブラウザである。現行の最新バージョンは0.9.2。Linuxのほかにも、Windows、Macといった複数のプラットフォームをサポートしている。

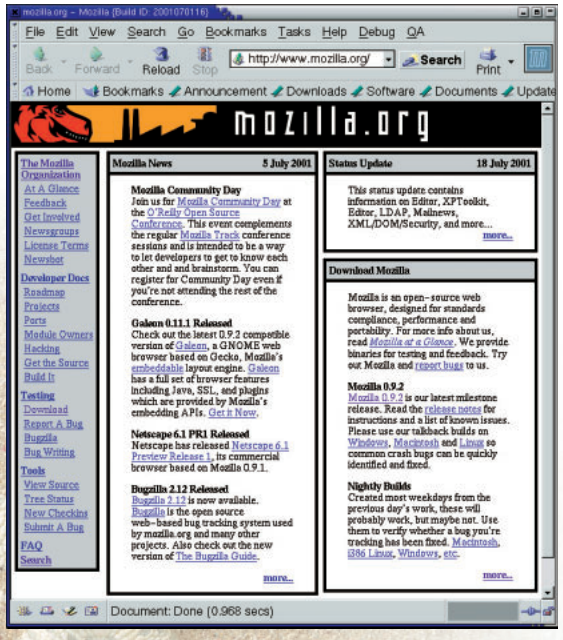
日本語
タブ
CSS
SSL
履歴保管

Geckoの採用

当初は、Netscapeより公開されたNetscape Communicator 5.0のソースコードを元に開発が進められていたが、Mozillaプロジェクトと並行して進められていたNGLayoutプロジェクトによって開発された「Gecko」というレンダリングエンジンが非常に優れていたことから、このレンダリングエンジンの採用を決定、以後、Netscape Communicator 5.0とはまったく異なるWebブラウザとして生まれ変わったのである。

Geckoは、レンダリング「エンジン」というだけあって、Mozillaの動力源となる中核的なコンポーネントだ。HTML 4.0、HTTP 1.1、CSS 1/2、XML 1.1、JavaScript 1.5、PNG／GIF／JPEGといった画像ファイル処理など、最近のWebブラウザに求められる機能のほとんどをサポートし、多様化するWebページのブラウズを可能とする。標準で多言語処理機能を備えており、日本語の表示も問題なく行える。

また、独立したコンポーネントとして再利用可能な設計になっていることから、Galeon、Nautilusなどの他のWebブラウザにもレンダリングエンジンとして採用されている。このことからも、優れた機能性を有していることがうかがえるだろう。



画面1　Mozilla.org in Mozill
基本的なインターフェイスは旧バージョンのNetscape Navigatorから大きな変更はないが、中身はまったく新しいブラウザだ。

充実のプラグイン

最近のWebページには、さきほど挙げたマークアップ言語や画像以外にも、ムービーやアニメーションといった多彩なコンテンツが含まれている。こうしたコンテンツのブラウズには、Macromediaの Flash Playerや AppleのQuickTime Movie Playerなどの専用プラグインが必要になる。

Linux版Mozillaでは、Netscape Communicator／Navigator 4.xのプラグインを流用できる。前出のFlashやAdobeの Acrobat Readerのプラグインなどが動作する。これらのプラグインは標準ではインストールされないため、Netscape用のプラグインをインストールして、必要なファイルをpluginsディレクトリにコピーし、MIMEタイプを適切に設定しなければならない。

入手方法とインストール時の注意点

開発プロジェクトであるMozilla.orgのほか、日本国内ならRingServer Project（http://www.ring.gr.jp/）でも入手可能だ。

配布形態は、ソースコードまたは各プラットフォーム用のバイナリ。このうちLinux用のバイナリには、Xで動作する専用のインストーラが含まれてい

るので、インストール作業自体は簡単だ。X上の端末エミュレータからインストーラを起動し、画面の指示に従っていけばよい。Linuxバージョンは、RPMパッケージおよびdebパッケージでの配布も行われている。

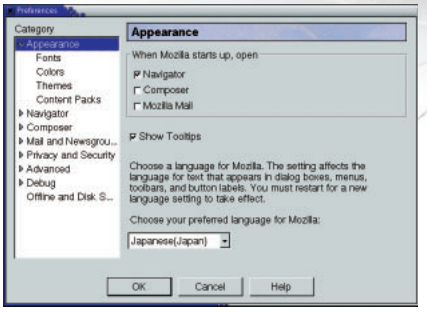
また、Red Hat Linux 7.1やKondara MNU/Linux 2.0などの新しいディストリビューションには、標準でMozillaのパッケージが含まれている。インストールに自信がない場合や、うまく動作しない場合は、これらのディストリビューションを利用すれば、確実にMozillaを体験できる。

動作に必要なソフトウェア要件としては、glibc 2.1以上、libjpeg 6.2、そしてJavaアプレットの動作に必要なJRE（Java Runtime Environment）などが挙げられる。基本的には、これらの要件を満たすディストリビューションであれば動作するが、インストールに際しては、各バージョンのリリースノートを参照してほしい。

日本語パック

メニューの表示などを日本語化する日本語パック（JLP：Japanese Language Pack）が、「もじら組」（http://www.mozilla.gr.jp/）を通じて配布されている。もじら組は、日本でのMozillaに関する日本語情報の提供を目的としており、日本語パックの配布のほか、本家であるMozilla.orgのドキュメントの和訳作業などを進めているプロジェクトだ。

日本語パックは、配布元のWebサイトでリンクをクリックするだけで自動的にインストールできる。本来なら、そのあと［View］メニューの［Languages and Web Contents］から［Jpanese（Japan）］を選択、Mozillaを再起動すれば、メニューが日本語化されるはずなのだが、0.9.2では、［View］メニューの［Preferences］で設定ダイアログを呼び出し、**画面2**のように［Appearance］を選択して［Choose your preferred language］のドロップダウンリストから［Japanese（Japan）］を指定する必要がある。設定変更後に再起動が必要な点は同じだ。



画面2　言語設定の変更
言語の変更のほか、Mozillaのオプションすべては、このダイアログで設定する。

ルック＆フィールの変更

Mozillaは、メニューなどのユーザーインターフェイスの外観を変更するテーマ機能を備えている。デフォルトのテーマは、伝統的なNetscapeに似たものだが、［表示］メニューの［テーマを適用］から、その環境にインストールされているテーマを選択することで、**画面4**に示すようにインターフェイスの外観をガラリと変えることが可能だ。

Mozillaのユーザーインターフェイスは、XMLとJavaScriptを用いて実現されており、そのコードはXUL（XML based User Interface Languageの略、「ズール」と発音）という言語で記述されている。テーマ機能にも、この仕組みが利用されているので、独自のテーマの作成や既存のテーマウのカスタマイズなども可能なわけだ。

まだ数は少ないが、ネット上で公開されているテーマもある。GNOMEやKDEのテーマでお馴染みのTheme.orgにも、いくつかのMozilla用テーマが登録されている。URLはhttp://x.themes.org/。表示されるページの左側にある［resources］から「mozilla chromes」をクリックすると見つかるはず。ただ、Mozillaのバージョンに依存する場合があるので、それぞれの注意書きをよく読んでからインストールするようにしてほしい。

画面4　テーマを変更したMozilla
このように、Mozillaの変更にあわせてウィンドウマネージャのテーマを変えるとさらに「いい感じ」になる。

画面3　日本語化されたメニュー
日本語になったからといって操作性が大きく向上するわけではないが、やはり「使い心地」がよい。

Gecko搭載のタブブラウザ

Galeon

最新バージョン： 0.11.1
開発プロジェクト：http://galeon.sourceforge.net/

日本語
タブ
CSS
SSL
履歴保管

Geckoをエンジンとし、GNOMEベースの軽快なGUIインターフェイスを身にまとった、新世代のLinux Webブラウザがここで紹介するGaleonだ。、タブ機能という武器を携えて、本家Mozillaの牙城を脅かす！？

Galeonは、MozillaのレンダリングエンジンGeckoを採用したGNOMEベースのWebブラウザだ。

MozillaのXULを利用したユーザーインターフェイスは、汎用性やユーザーカスタマイズの面で大きなアドバンテージがあるが、動作が重いことが弱点である。Galeonでは、Geckoの優れたレンダリング機能はそのままに、より軽快な操作感を目指している。

開発プロジェクトは非常に活発に運営されているようで、頻繁に新しいバージョンがリリースされている。最新バージョンは、7月20日にリリースされたばかりの0.11.2。これは、Mozilla 0.9.2に対応している。

Mozillaが必須

Geckoをレンダリングエンジンとしている関係上、インストールに際しては、Mozillaがあらかじめインストールされている必要がある。バージョンによって、ベースとなるMozillaのバージョンが異なるので、開発プロジェクトのWebサイトに掲載されているリリースノートで必ず確認してほしい。

Galeonのインストールには、このほかにもソフトウェア要件に関して気をつけるべき点が多い。GNOMEアプリケーションであるため、gnome-libsやORBit、GConf、oafなどのバージョンの確認が必要となるのだ。配布されているRPMパッケージにも依存関係が設定されているので、“ rpm -qR ”としてインストール前に環境が整っているかどうか確認しておこう。もちろん、ソースからコンパイルする場合にも、同様のソフトウェア環境が要求される。

ちなみに、バージョン0.11.2のRPMパッケージをインストールするには、以下のパッケージがインストールされていなければならない。

- **Mozilla 0.9.2**
- **gnome-libs 1.2.0以上**
- **ORBit 0.4.0以上**
- **libglade 0.13以上**
- **libxml 1.8.11以上**
- **gnome-vfs 0.6.0以上**
- **GConf 0.12以上**
- **oaf 0.6.2以上**
- **gdk-pixbuf 0.10.0以上**

画面1　Galeonのメインウィンドウ
シンプルなインターフェイスがGaleonの特徴の1つ。GNOMEアプリケーションなのでGTK＋やSawfishのテーマを使って外観を変更することも可能だ。

最大の武器はタブ機能

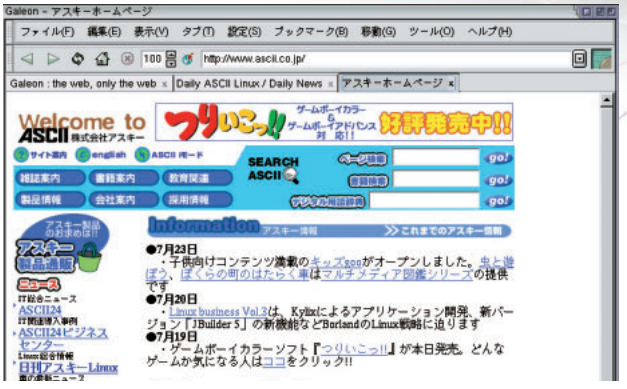
Geckoの採用により、レンダリングに関しては、Mozillaと同等の機能を有している。現時点でGaleonがMozillaを凌ぐ最大のセールスポイントは、タブ機能にあるといってよい。

タブ機能は、DonutやBug BrowserなどのIEコンポーネントを利用した

WindowsのWebブラウザでお馴染みのものだ。オープンしているWebページごとにタブが付けられ、タブをクリックすることでページを切り替えられる。複数のWebページを交互に参照する場合などに非常に便利な機能である。

バージョン0.10.xまでは、デフォルトではタブ機能が有効になっておらず、設定を変更する必要があったが、0.11.1以上のバージョンからは標準でタブ機能が使えるようになった。これで、より多くのユーザーにGaleonの魅力をアピールできるようになるはずだ。

画面2　とても便利なタブ機能
複数のページを切り替えながらブラウズできてとても便利。一度使うと手放せなくなる？



日本語処理は万全

これもGeckoの機能にかかわる部分なのだが、日本語を含むWebページのブラウズには何の問題もない。標準の状態で、Shift-JIS、EUC-JP、ISO-2022-JPのエンコードが可能だ。また、フォームなどへの日本語入力も、かな漢字変換ソフトとX用入力メソッドが適切に設定されていれば問題なく行える。

メニューなどの日本語化には、GNOMEの多言語処理機能が利用されている。普通にインストールするだけで、その環境で指定されているロケール情報に従ってメニューやメッセージに使用される言語が自動的に判定されるので、ユーザー側で特に設定する必要はない。

豊富なブックマーク機能

URLのブックマーク機能は、Webブラウザに欠かせないもの。Galeonは、この点でも有効なソリューションを提供してくれる。通常のブックマーク機能に加えて、自動ブックマーク、スマートブックマーク、MozillaおよびNetscape Communicator／Navigatorとのブックマークのインポート／エクスポート、専用のブックマークエディタといった機能を備えているのだ。

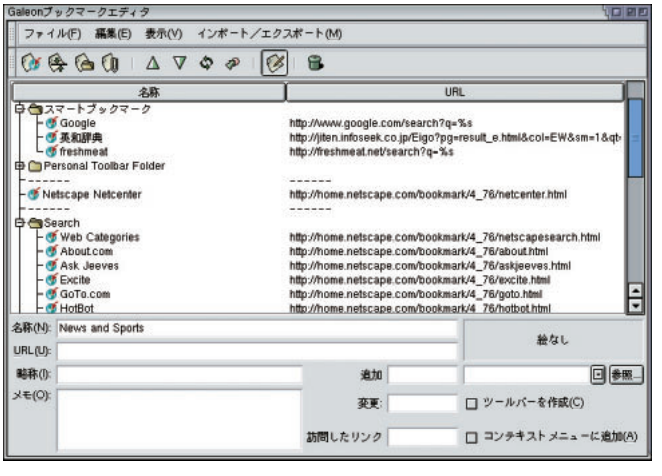
このうち、自動ブックマークは、従来からある履歴の保存機能に似た機能で、訪れたWebサイトのURLを自動的に保存してくれる。履歴と異なるのは、保存したURLの情報をブックマークとして扱える点。通常のブックマークメニューに登録されるので、履歴情報のウィンドウを開く手間を省いて、素早くURLを指定できる。ただし、履歴機能のように1週間分を記録するといったものではなく、オプションに指定した件数だけブックマークとして登録され、古いものは上書きされる。このため、比較的新しいURL情報しか記録されない。なお、Galeonは従来の履歴保管機能も備えている。

スマートブックマークは、検索や辞書などの機能を提供するアプレットだと思ってもらえばよい。**画面3**に示すような小型のテキストボックスを持ち、入力した語句を検索なら検索エンジン、辞書であればネット上の辞書サービスに送り、結果をURLで受け取ってウィンドウに表示してくれる。

専用のブックマークエディタ（**画面4**）は、ツリー形式でブックマークの階層を表示するようになっており、わかりやすいメニュー構成とあわせて直感的な操作を可能にしている。前述のブックマークのインポート／エクスポートもこのツールから行える。

画面3　スマートブックマーク
とても小さなインターフェイスを持つツールだが、その効果は大きい。

画面4　専用のブックマークエディタ
Galeonの持つ豊富なブックマーク機能を、ここで一元的に管理できる。使い勝手もなかなかのものだ。



日本語化が待たれる高速・多機能ブラウザ

Opera

最新バージョン： 5.0
開発プロジェクト：http://www.opera.com/

日本語
タブ
CSS
SSL
履歴保管

Operaは、北欧から登場した世界最速のレンダリング能力を誇る高機能Webブラウザだ。操作の多用性と柔軟なカスタマイズ機能を持つ優れたインターフェイスを備えている。日本語サポートへの期待も高い。

Operaは、ノルウェーに本拠を置くOpera Softwareが開発したWebブラウザである。

独自のレンダリングエンジンを備え、世界最速を謳っている。また、プログラムサイズが小さいのも特徴で、PDA、セットトップボックス、Webアプライアンスなどの搭載ブラウザとして採用されている。

速くて軽いだけでなく、タブによるページの切り替えなどの多彩な機能を持ち、CSSやJavaScript、XML、SSLといった標準規格もサポートしている優れたWebブラウザなのだが、非常に残念なことに日本語処理に対応していない（にもかかわらず、ここで紹介するのは日本語化への期待を込めてのことである。どうか実現しますように）。

Linux、Windows、Mac、Solaris、BeOS、OS/2といった多くのプラットフォームに対応しており、Linux版の最新バージョンは5.0だ。

フリーソフトウェア？

Operaは、基本的には無償提供されるフリーソフトウェアだが、ツールバーの右側に企業の広告バナーが表示される。バナー表示をなくすには、ユーザー登録を行い、規定の料金（39USドル）を支払う必要がある。

GPLなどに基づいて配布されるオープンソースソフトウェアと異なる点にも留意しておきたい。ソースコードは提供されておらず、リバースエンジニアリングなどの行為も禁止されている。無許可の再配布もライセンスに抵触するので注意が必要だ。

配布形態とインストールについても、ここで触れておこう。当然のことながら、配布されるのはバイナリのみ。RPM、deb、tar＋gzipの各形式が用意されている。なお、Operaの動作にはQtライブラリが必須となるが、スタティックにQtをリンクしたバージョンが用意されているので、必ずしもシステムにQtがインストールされている必要はない。

画面1　Opera 5.0のメインウィンドウ
ウィンドウが3ペインに分割されるためWebページを表示する範囲がどうしても狭くなってしまう。1280×1024ピクセル以上の解像度なら快適に使えるのだが……。

多彩なユーザーインターフェイス

Operaの基本的なインターフェイスは3ペイン構成だ（**画面1**）。右側が通常のブラウズ領域、左側が「Hotlist」と呼ばれるブックマークのツリー表示領域で、Hotlistはさらにツリー部分と選択されたフォルダ内のブックマークを一覧表示する部分に分かれている。一覧表示されたブックマークをダブルクリックすることで、そのURLにアクセスできる。この機能は、もうひとつの大きな特徴であるタブ機能とあわせて、マウス操作による優れたWebナビゲーション環境をユーザーに提供している。

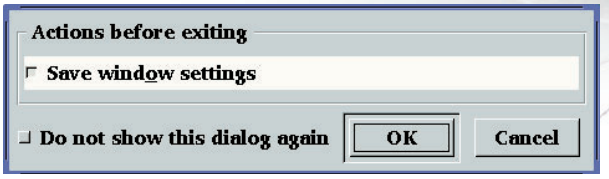
3ペイン構成の欠点として、画面1からもわかるように、ブラウズ領域が狭

くなることが挙げられる。ディスプレイ解像度の低い環境では、かなりつらいものがあるはずだ。Operaには、この点についても柔軟な解決策が用意されている。Hotlistの代わりに「Bookmark bar」を使うことで、ブラウズ領域を広げることができるのだ。

Bookmark barには、ブックマークツリー上にある各フォルダが1つのボタンとして表示され、ボタンを押すと、そのフォルダ内にあるサブフォルダとブックマークが表示されるという仕組みになっている（**画面2**）。

ブックマークツリーのどのレベルをBookmark barとするかは、ユーザーが任意に設定できる。たとえば、ツリーのトップレベルを指定してBookmark barにすべての登録済みブックマークを含めたいなら、ルートにあたる［Bookmarks］を右クリックしてポップアップメニューから［Show on Bookmark bar］を選択すればよい。

Hotlist、Bookmark barのほかにも、一般的なナビゲーションボタンを備えたTool barとProgress Barが用意されている。これらのインターフェイス要素はすべて、表示／非表示を切り替えられるだけでなく、ウィンドウのどの位置（上下左右）に配置するかを設定できるようになっている。つまり、好みに合わせてインターフェイスを柔軟にカスタマイズできるのである（**画面3**）。



画面4　終了時に表示されるオプションダイアログ
終了時の状態を保存したい場合は、［Save Window setting］をチェック。オフにすると、前回保存した状態が引き続き保持される。

終了状態の保存と自動リロード

非常に便利であるのに、ほかのWebブラウザにあまりみられないのが「終了時にオープンしているWebページを保存する」という機能だ。

Operaの終了時には、**画面4**のダイアログが表示される。このとき、［Save window setting］をチェックしておくと、次回起動時に終了時点で開いていたWebページ（window）が自動的にブラウズされる。つまり、終了時の状態が復元されるわけだ。複数のタブページを開いていた場合にも、そのすべてが復元される。アナログ回線でダイヤルアップといった低速な接続環境では必ずしも有効ではないが、高速で安定したインターネット環境が整っているなら、かなり使える機能だ。

同じ「自動もの」としては、Webページの自動リロード機能も便利。これは文字どおり、表示されているページを自動的に更新してくれる機能だ。ニュースや株価情報など、コンテンツが定期的に更新されるサイトを長時間ブラウズする際などに有効だろう。

機能自体のオン／オフと更新間隔はタブページごとに設定可能になっている。ブラウズ領域の任意の場所を右クリックし、ポップアップメニューから［Reload every］を選択すると設定用のサブメニューが表示される。更新間隔は、プリセットされた5～30分の値から選んでもよいし、［Custom］をクリックして設定ダイアログで任意の値を指定することもできる。

画面2　Bookmark barを有効にした状態
ブラウズ領域は広がるが、ツリー表示のHotlistに比べると操作性の面では少し落ちる。どちらを選ぶかは、ユーザーしだいだ。

画面3　よりシンプルにカスタマイズ
Netscape Navigatorライクな（？）インターフェイス構成にしてみた。このような柔軟性がOperaの「売り」のひとつ。

Web機能も充実のKDEファイルマネージャ

Konqueror

最新バージョン： 2.1.1
開発プロジェクト：http://www.kde.org/

日本語
タブ
CSS
SSL
履歴保管

ファイルマネージャとしてKDEデスクトップ環境の中核を担うKonquerorは、さらにWebブラウザとしての機能をも備える。デスクトップからネットまでをシームレスにつなぐインターフェイスが最大の魅力だ。

KDEの標準ファイルマネージャであるKonquerorはWebブラウザ機能を備えている。Webブラウザとしても高い完成度を持っており、一般的な機能を過不足なくサポートする。

KDE環境で動作

Konquerorを利用するには、当然のことながら、KDE環境が必要となる。配布もKDEに含まれる形で行われている。KDEは、kdebase、kdelibs、kdeutilesといった機能ごとに分割されたパッケージとして配布されている。Konquerorが含まれているのは、kdebaseパッケージとなる。

また、KDEのデスクトップ環境をまるごとインストールする必要があるので、関連するソフトウェアの依存関係などに注意しなければならない。インストールするパッケージ数も多く、かなり大変な作業となる。必要とされる知識も多い。日本語化のためのパッチなども必要となるため、ここはディストリビューションに標準で提供されるバージョンを利用するほうが賢明かもしれない。

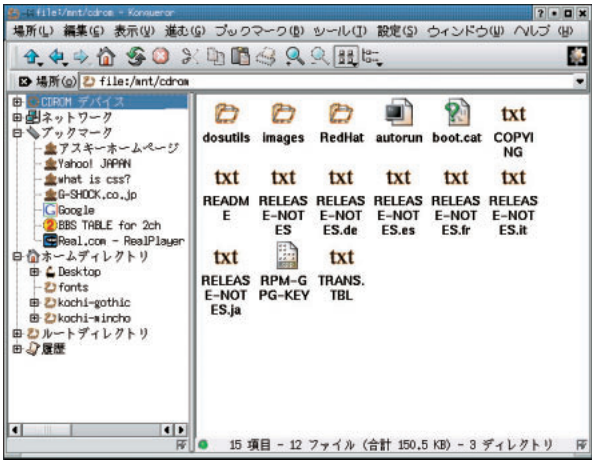
なお、日本KDEユーザー会（http://www.kde.gr.jp/）が作成した日本語化パッチを適用したバイナリパッケージがRingServer Project（http://www.ring.gr.jp/）を通じて配布されている。日本語化された最新のKDEデスクトップがほしいという方は、このパッケージを利用するのもひとつの手だ。

ファイルマネージャである利点

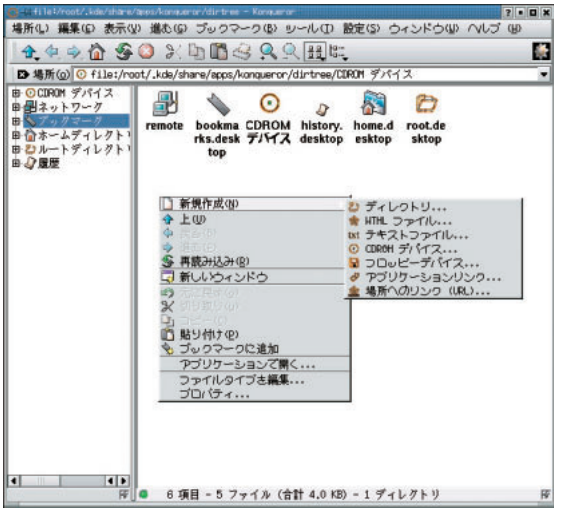
Konquerorは、ファイルマネージャとしての利点を生かした優れたインターフェイスを持っている。「サイドバー」と呼ばれるインターフェイスがそれだ。［ウィンドウ］メニューにある［サイドバーを表示］をチェックすると、ウィンドウの左側に**画面2**に示すようなツリーが表示される。

サイドバーに表示されるツリー階層のルートには、［ネットワーク］、［ブックマーク］、［ホームディレクトリ］、［ルートディレクトリ］、［履歴］という5つのフォルダがある。名前からわかるように、これらのフォルダには、ブックマークや履歴として保存されたURL、ディレクトリツリー、LAN上の共有リソー

画面1　Webページを表示したKonqueror
このページは英語だが、日本語ページの表示にも問題はナシ。JavaScriptやNetscapeプラグインのサポートなどWebブラウザとしての機能も十分。



画面2　サイドバーを表示した状態
ファイルマネージャ＋Webブラウザの機能を活かすには、このビューが最適だ。Windowsのエクスプローラに近い操作性を持っている。



画面3　サイドバーへのアイテム追加
ハードディスクドライブから、ネットのURLまで、さまざまな場所へのポインタを追加できる。

画面4　ブックマークエディタ
このエディタを使ってNetscape NavigatorやMozillaのブックマークをインポートすることもできる。

スへのポインタなどが含まれている。

このインターフェイスを使うことで、ユーザーはローカルディスク上のディレクトリからLAN上の共有リソース、さらに外部ネットワークのURLへと、マウス操作だけで簡単に行き来できるようになる。

サイドバーはユーザーによるカスタマイズも可能だ。サイドバーの設定は、［進む］メニューにある［サイドバーの設定］から行う。サイドバーのツリーには、CD-ROMやフロッピードライブなどのデバイス、任意のディレクトリ、アプリケーションへのリンク、URLといったアイテムを追加することができる（**画面3**）。

ユーザーの設定しだいで、ローカル／リモートを問わない任意の「場所」へのアクセスポイントを設けられるわけだ。うまく活用できれば、さらに柔軟で使い勝手のよいインターフェイスが実現するはずである。

Webブラウザとしての Konqueror

サイドバーにフォルダとして登場したように、ブックマークと履歴によるURLの保管というWebブラウザの基本中の基本というべき機能は、きっちり押さえられている。このうちブックマークについては、専用のエディタ（**画面4**）による編集・管理、MozillaやNetscapeのブラウザからのインポート／エクスポートといった十分な機能が備わっている。履歴のほうはどうかというと、こちらはやや貧弱で、残念ながら専用の管理インターフェースは用意されていない。

そのほかのWebブラウザの機能についても見ておこう。

●独自開発されたレンダリングエンジンは、HTML 4.0、JavaScript、CSS 1/2、SSLなどの標準規格をサポート

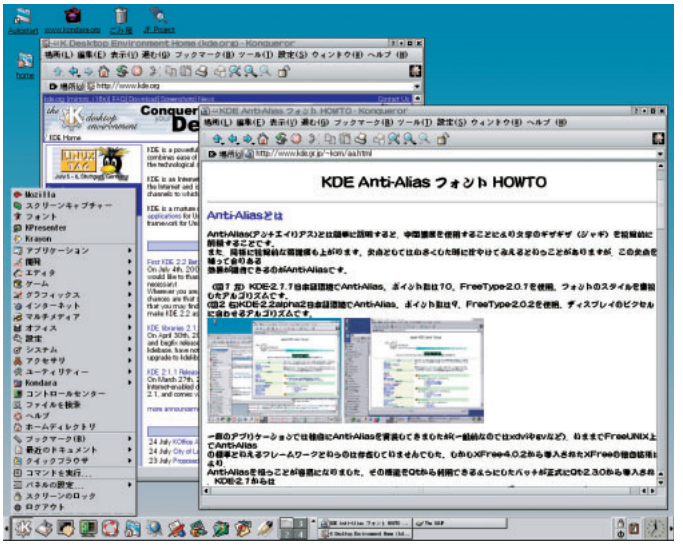
●文字コードの自動判別や文字セットごとのフォント指定など、多言語処理に対応

●FlashやPDFを処理するNetscapeのプラグインとの互換性を持つ

このように、専用のブラウザではないにもかかわらず、KonquerorはWebページのブラウズに必要十分な機能を有しているのだ。

さらにKDEのアンチエイリアス機能を有効にすれば、KonquerorによるWebページの表示にもアンチエイリアシングがかかり、とても美しいフォントでブラウジングすることができる。

画面5　アンチエイリアスを設定したKDEデスクトップ



Webブラウザ機能を備えたGNOMEファイルマネージャ

Nautilus

最新バージョン： 1.0
開発プロジェクト：http://nautilus.eazel.com/

日本語
タブ
CSS
SSL
履歴保管

最初の正式バージョンがリリースされた直後に開発元であるEazelが経営上の理由から業務停止に……。GNOMEの次世代ファイルマネージャとしての期待が高いだけに、今後も開発が続けられるように願いたい。

Nautilusは、Eazelが開発したGNOME用のファイルマネージャで、MozillaをベースにWebブラウザとしても動作する。

簡易Webブラウザ

もともとがファイルマネージャであるため、Webブラウザとしての機能はそれほど豊富ではない。

しかし、Mozillaをベースとしているということは、レンダリングエンジンがGeckoであるということである。したがって、Webコンテンツのブラウズ能力自体は高いものがある。CSSの解釈や、Flashなどの特殊なコンテンツの再生なども問題なく処理できる（**画面2**）。標準の状態で日本語の表示にも問題はない。

ただ、JavaScriptを使った新しいウィンドウのオープンなど、一部サポートされていない機能もある。このため、ページは表示されるのに、提供されている機能は使えないといったケースもありえる。

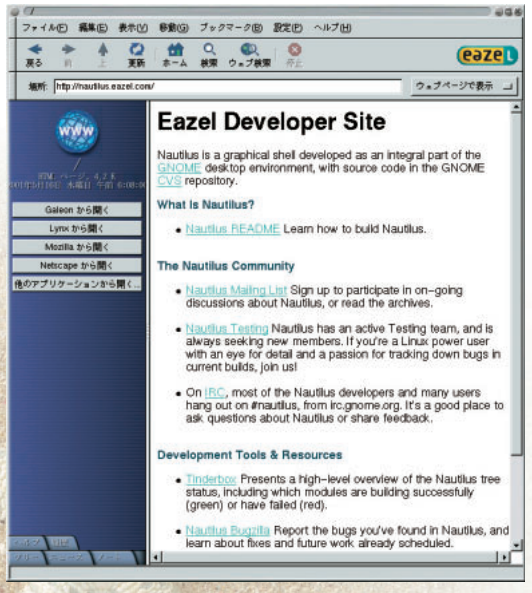
たとえば検索サイトで、ブラウズと語句の入力はできるのに、いざ検索を実行しようとするとエラーになるといったことがあるのだ（**画面3**）。

とはいえ、Webページのブラウズに最低限必要な機能は備えているので、ファイルマネージャとして使いながら、ちょっとWebサイトを覗いてみたいといった場合に使えるかもしれない。

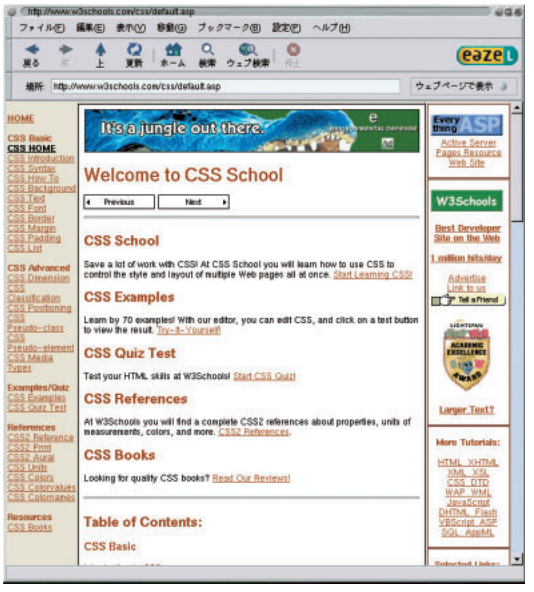
日本語入力について

前述したように、Webコンテンツに含まれる日本語は正しく表示することができる。また、フォームなどへの日本語入力も基本的にはOKだ。ただし、入力については、Mozillaのバージョンが古いと、正しく処理できないことがあるので注意が必要だ。

今回確認したNautilus 1.0＋Mozilla 0.9.2という組み合わせでは、日本語まわりの処理について特に問題は発生しなかったので、NautilusのWebブラウザ機能を利用する際には、できるだけ最新バージョンのMozillaをインストールしておくようにしよう。



画面1　NautilusでWebページを表示
HTMLのレンダリングはGeckoベースなので問題なし。いま少し機能が追加されるだけで、Webブラウザとしてかなり使えそうなのだが……。



画面2　CSSを使ったページもOK
CSSの解析もGeckoの機能のひとつ。画面上部にあるGIFアニメやFlashもきちんと動作していた。

配布元とインストールの注意点

現在、Nautilusの主な配布元となっているのは、GNOMEプロジェクト（http://www.gnome.org/）と、独自にGNOMEのパッケージ化を行っているXIMIAN（http://www.ximian.com/）だ。

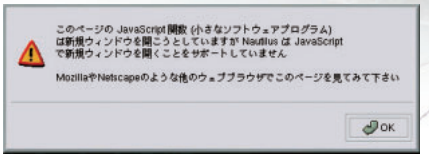
GNOMEプロジェクトでは、ソースコードおよびRed Hat、SuSEなどの各ディストリビューション用のRPMファイルを配布している。

XIMIANは、同社がHelix Codeという名称だった当時から、独自にGNOMEのカスタマイズと配布を行ってきた。現在は、XIMIAN GNOME 1.4をリリースしている。XIMIAN GNOMEは、基本的に独自のインストーラによるネットワークインストールという形態で提供される。

いずれにせよ、Nautilusを動作させるには、GNOMEをバージョン1.4にアップグレードする必要がある。ライブラリやユーティリティなどソフトウェアの依存関係がかなり複雑なので、それぞれの配布元が提供しているインストールに関するドキュメントをよく読んでから作業に移るようにしたい。

なお、執筆時点でNautilusを標準採用している日本語ディストリビューションは、Kondara MNU/Linux 2.0のみである。

とにかく一度は試してみたいといった場合にはKondaraがお勧めだ。



画面3　JavaScripに関するメッセージ
日本語の表示、入力とも問題はないのだが、検索ボタンをクリックすると、画面のメッセージが表示されてしまう。

Column

ファイルマネージャとしてのNautilus

Webブラウザとして見た場合、高機能とは言えないNautilusだが、「本業」であるファイルマネージャとしてはどうなのだろうか。

■武器は表現力

ひと目見てまず気がつくのが、その「美しさ」である。ディレクトリやファイルを示すアイコンは、かなり凝ったものになっている。ファイル名などのテキストも、これまでのLinuxには見られなかったほど美しいフォントで表示される。ただ見た目がきれいなだけでなく、多くの情報を視覚的に表現できる点がNautilusの最大のセールスポイントだ。

下の画面をよく確認してほしい。これは、test-userというユーザーアカウントでログインし、Nautilusで/homeを表示したところだ。左側のペインに表示されているアイコンのラベルテキストからは、「カレントディレクトリがhomeであること」と「homeの最終更新日時が今日の12時37分であること」がわかる。また、下にある鉛筆に禁止マークがついたアイコンは、test-userがhomeに対して書き込みを行う権利を与えられていないことを示している。

右側のペインにある各ディレクトリのアイコンは、test-userのパーミッションの状態によって、それぞれ異なった表示になっている。httpdディレクトリは参照は可能だが書き込みは禁止されており、ircディレクトリは参照も不可、ホームディレクトリであるtest-userディレクトリに対してはすべての操作が可能であるといったことがわかる。

ファイルについても、ファイルタイプ別のアイコン、画像ファイルのサムネイル表示、テキストファイルのプレビュー表示、ディレクトリと同様のパーミッション情報など視覚的に情報が表現される。

また、Nautilusを使ってデスクトップ画面を描画することで、デスクトップ上のアイコン、アイコンのラベルなどを従来のGMC（GNOME Midnight Commander）のものより「美麗」にすることができる。描画の切り替えは［設定］メニューの［編集設定］で表示されるダイアログで行う。ダイアログの右側のリストから［ウィンドウ＆デスクトップ］を選択し、［デスクトップ描画にNautilusを使用.］をチェックすればよい。

ファイルマネージャモードのNautilus

通常のデスクトップ（上）とNautilusによる描画の違いに注目

新世紀に復活を期す

Netscape 6

最新バージョン： 6.01
Webサイト： http://www.netscape.com/

日本語
タブ
CSS
SSL
履歴保管

Netscape 6は、Mozillaをベースに独自のカスタマイズとチューニングを施したWebブラウザだ。正式リリースの最新バージョンは6.01、現在、次期正式リリースとなるバージョン6.1のベータ（RC1）が公開されている。

Mozillaとの違いは？

最新バージョンの6.01は、Mozillaのマイルストーンビルド0.6に相当するソースコードを元に開発されている。基本的なWebブラウザとしての機能は、ベースとなるバージョンのMozillaと同等だといっていい。**画面1**と**画面2**を比べてもらえば、外見上もほとんど変わらないことがわかるだろう。では、両者の違いはどこにあるのだろうか？

ダウンロード時に気づく違いは、NetscapeがLinux版として、日本語、英語、フランス語、ドイツ語という主要な言語用のパッケージを用意している点だ。Mozillaは、1つのバイナリで多言語をサポートするように設計されており、配布の際には各プラットフォーム用に1つのバイナリしか用意されない。シングルバイナリ／多言語サポートという点では、Netscape 6も同じなのだが、各言語向けに設定された配布用のパッケージを用意することでユーザーの便宜を図っているのだ。

使い始めて気づく違いとしては、初めて触れるユーザーでも一定のレベルで使いこなせるように、適度にカスタマイズが施されている点だ。たとえば、「Sidebar」にニュース、天気予報、買い物などの情報が表示されるようにデフォルトで設定されている（**画面3**）。もちろん、日本語版では日本語の情報が表示されるようになっている。また、ステイタスバーには、Netscape 6のプラグイン情報から、株価情報、芸能ニュースまで、数多くのWebサイトが登録されたブックマークが用意されている（**画面4**）。

最大の違いは性格！？

ここまでに挙げた相違点は、かなり細かいものばかりだ。どれも設定しているか、していないかの違いだけで、やろうと思えばユーザー側でカスタマイズできる範囲の違いでしかない。ただ、こうした細かな配慮が「使い心地」

画面1　Netscape 6
デフォルトで「Modern」テーマが適用されている。これも、少しでも見栄えをよくするというユーザーへのアピールかも？

画面2　Mozilla 0.9.2
同じ「Modern」という名前のテーマを適用したMozilla。少し色合いが違う程度で外観上の差はほとんどない。

に影響するのも確かだ。また、使う側からすれば、「できないこと」をできるようにするよりも、「すでにできているもの」を使わないことのほうが楽であることは間違いない。

ちょっとニュアンスは違うかもしれないが、Mozillaが「機能は用意しておいたので、あれこれ試しながら使ってみてください」というスタンスであるのに対して、Netscape 6のほうは「十分に活用できるよう準備してきましたので、どうぞお使いください」といった感じがするのだ。

実は、こうした両者の性格の違いは、現時点でリリースされているバージョンの持つ意味合いの違いを反映している。Netscape 6はオープンソースソフトウェアとして開発され、無償で配布されているが、Netscapeという企業のれっきとした「製品」である。一方、Mozillaの現行バージョンはというと、やはりオープンソースで開発され、一般向けに配布されているものの、あくまでも正式リリースに向けてのテストを目的とした「ベータ版」的な位置付けにある。

製品であるNetscape 6は使いやすく便利な環境を、Mozillaはテストの共通の基盤となるベーシックな環境をユーザーに提供する必要があるというわけだ。

インストール時の注意点

日本語版のNetscape 6は、NetscapeのWebサイトで配布されている(http://home.netscape.com/ja/browsers/6/)。インストーラのみをダウンロードしてネットワークインストールする方法と、インストーラを含む全バイナリをアーカイブしたパッケージをダウンロードしてローカルでインストールする方法がある。

どちらの場合も、ダウンロードしたアーカイブを展開し、X上の端末エミュレータからnetscape-installerを実行する。あとは、画面の指示にしたがってインストールを進めていくだけだ。

と、すんなりいくはずだったのだが、今回の特集での評価のためにインストールした際に、ちょっとした問題があることが判明した。「全部入り」バージョンのほうを使ってインストールしたところ、途中でインストーラが止まってしまったのだ。あれこれ調べた結果、原因はパーミッションとファイルの配置にあることがわかった。

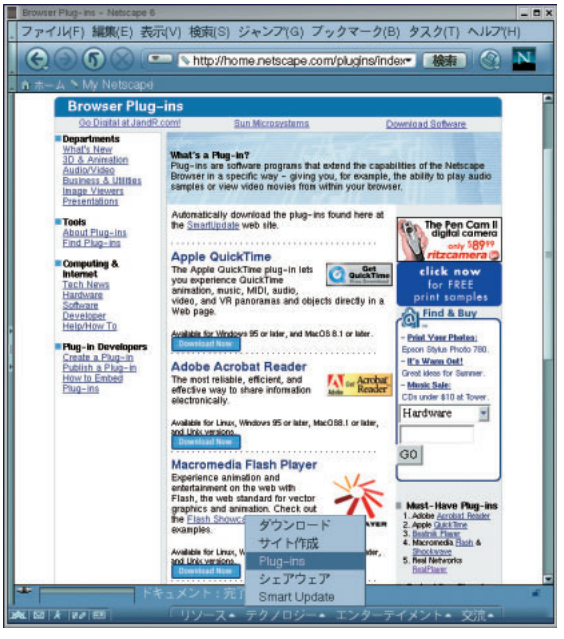
Netscape 6のインストールセットに含まれるファイルは、Mozillaとほぼ同じ内容になっている。ところが、一部のファイルのパーミッションと配置がMozillaとは異なっていたのだ。これを修正したところ、きちんとインストールすることができた。必要な修正は次の2点。

●アーカイブの展開先ディレクトリ（netscape-installer）にあるbrowser.xpiとpsm.xpiに実行属性を与える

●xpiディレクトリを作成し、拡張子が「.xpi」の全ファイルを移動する

これで準備完了。今度こそ、X上の端末エミュレータからnetscape-installerを実行し、あとは画面の指示にしたがってインストールを進めていくだけ……でいいはずだ。

画面3　My Sidebarを表示したところ
デフォルトの設定で、asahi.com、楽天市場などからの情報が随時更新されて表示されるようになっている。



画面4　ステイタスバー埋め込まれたブックマークボタン
こちらもデフォルト状態で、すぐに使えるように設定済み。画面のプラグイン情報は、ぜひ一度チェックを！

W3m

テキストWebブラウザの可能性を示す

最新バージョン： 0.2.1
開発プロジェクト： http://ei5nazha.yz.yamagata-u.ac.jp/~aito/w3m/

日本語
タブ
CSS
SSL
履歴保管

もともとはHTMLファイルを表示するページャソフトとして開発されたw3mだが、徐々に機能強化されていくうちにWebブラウザとしての人気が高まってきた。口コミで噂が広まったのは「本物」の証だ。

w3mは、伊藤彰則氏が開発した国産のテキストWebブラウザだ。作者ご本人がホームページ上で明かされているところによると、「WWWを見る」の頭文字（？）をとって命名されたそうだ。

テーブル、フレームに対応

w3mの大きなアピールポイントのひとつは、テキストWebブラウザながらテーブルとフレームをサポートしている点だ。フレームは**画面1**に示すように、テーブルに置き換えて表現される。

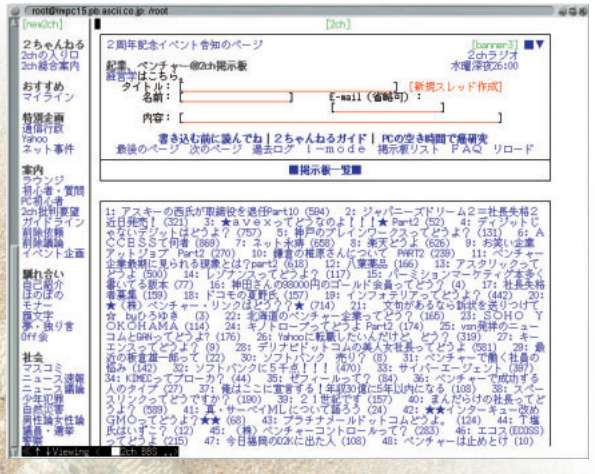
もともとテキストブラウザは、掲示板やニュースサイトなどのように文字情報が中心のWebサイトのブラウズに適しているのだが、従来のテキストブラウザでは、そのサイトでフレームが使用されていると表示することすらできなかった。

w3mなら、画面1のようにXの端末エミュレータ上で起動して画面サイズを大きくしてやれば、あたかも通常のブラウザで表示しているかのようにフレームを使ったWebページをできる。これは、かなりすごいことだ。

なお、CSSには対応していないが、フレームの場合と違って、CSSはサポートされていないWebブラウザでの表示に影響を与えることはない。CSSのスタイルは反映されないが、テキスト自体は表示されるのだ。

マウスが使える！

w3mを初めて使った際に驚かされるのが、マウスによる操作が可能なことである。ハイパーリンクが指定されているテキストをマウスでダブルクリックすれば、リンク先のURLを表示することができる。これは、GUIインターフェイスを備えたふつうのWebブラウザであれば、まさにふつうの操作なのだが、テキストブラウザでこれができるのは驚きだ。



画面1　フレームを含むWebページを表示したところ
かなりの精度でフレームからテーブルへ変換されるので、見た目の違和感は少ない。

リンクの操作以外にも、フォームの入力領域をダブルクリックするとテキスト入力用のプロンプトが表示されるし、フォームのボタンに付けられたテキストをダブルクリックして「押す」こともできたりと、マウスで行える操作が結構ある。

これらのマウス操作は、X上の端末エミュレータから起動した場合だけでなく、gpmでマウスが有効になっているコンソールでも行える。

テキストブラウザの定番へ

そのほかのWebブラウザに必要な機能もひと通り備えている。ブックマーク、簡単な履歴保存、クッキーの管理、プロキシ経由の接続などが可能で、さらにSSLにも対応している。

テーブル／フレームのサポートとマウスによる軽快な操作性を備えたw3mは、テキストWebブラウザの定番となりつつある。Plamo Linux 2.1.xやVine Linux 2.1.5、Kondara MNU/Linux 2.0など、収録されているディストリビューションも増えてきているので、ぜひ一度お試しあれ。

Lynx

テキストブラウザの老舗はいまだ健在

最新バージョン： 2.8.4
開発プロジェクト：http://lynx.browser.org/

日本語
タブ
CSS
SSL
履歴保管

どのLinuxマシンにもいる気さくなおじさんといった風情を漂わせるのが、Webブラウザ界の長老ともいうべき存在のLynxだ。最近では使いこなせるユーザーも減ってきたが、その実力はまだまだ健在である。

Lynxといえば世界標準ともいえるテキストWebブラウザだ。1980年代にカンザス大学で生まれたこのプログラムは、多くの時間と多くの開発者の手を経て、さまざまな機能が加えられてきた。そして、現在もなお進化中である。

また、Linuxを含むUNIX系OS、VMS、Windows、OS/2、Mac（ベータテスト中）といった数多くのプラットフォームに移植されている。

豊富な機能とオプション

Lynxは、ほとんどのLinuxディストリビューションに標準で採用されている。多くのユーザーに好まれ、長く使われてきたことからもわかるように、Webブラウザとしてのベーシックな機能（ブックマークや履歴の保管など）は十分に備えている。

また、ユーザー側でキーバインドなどを自由に変えられるほか、マウスの使用から、クッキーの管理、HTMLに含まれる改行（BRタグ）の無視など、あらあゆるレベルで操作性や機能のオプションを変更できる。

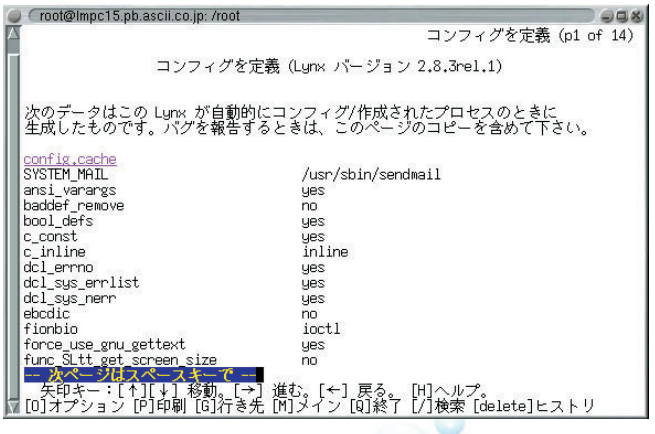
これらの設定は、/etc/lynx.cfgに保存されている。実際に内容を見ると、その設定オプションの多さに驚かされる。反面、完全に使いこなすには、かなりの知識と自分なりの工夫やアイデアが必要だろう。

使える場面は？

テキストベースのブラウザなので、画像やアニメーション、音楽ファイル、ムービーといったコンテンツを含んだWebページのブラウズには、向いていないだろう。しかし、w3mのところでも触れたニュースサイトや掲示板、あるいは「テキスト系」とか「文章系」と呼ばれるテキスト情報中心のサイトのブラウズでは、最新のグラフィカルなWebブラウザを凌ぐ操作性を発揮することもある。

これらのテキスト中心のサイトは、派手さがないので目立たないが、実は現在のインターネットのひとつの潮流になっている。そんなサイトは、渋くLynxを使って眺めるのが「通」というものだ。

このほかにも、古いノートPCなどのようにX Window Systemを動作させるのが少しつらいマシンで使うことも考えられる。また、HTMLで書かれたソフトウェアのマニュアルなどを見るときも、いちいちXでWebブラウザを起動して［開く］メニューをクリックして、ダイアログでディレクトリを選んで……、とやっていくよりも“lynx ファイル名”と1つのコマンドでパッと開いたほうが便利である。



画面1　Lynxのオプション表示画面
lynx.cfgの内容はLynxインターフェイス内からも確認できる。

これからもLynx

GUIインターフェイスがあたりまえになり、マルチメディアコンテンツが増えてきた現在でも、Lynxには多くのユーザーがいる。これは、テキストベースにはテキストベースなりの良さがあり、Lynxがその良さを備えている証拠でもある。ちょっとGUIの世界を離れて、一度Lynxに触れてみてほしい。

選定！ LinuxのベストWebブラウザ

ここまでは各ブラウザごとに、その機能や特徴を紹介してきたわけだが、以下では、お互いを比較しながら、ブラウザ選びのポイントへと話を進めていこう。そして最後にお勧めのベストWebブラウザを決定したい。

検討対象を絞り込む

紹介したすべてのブラウザを機能ごとに比較していたのでは、ページ数がいくらあっても足りないので、いくつかの要素をもとに対象とするものを絞り込んでいこう。

■何はともあれ、まずは日本語

日本語文化圏で暮らし、日本語の情報を必要とする我々にとって、やはり重要になってくるのが日本語処理の問題だろう。ということで、いきなりOperaが脱落してしまうことになる。ただし、その機能性の高さに敬意をはらって、ときおりゲスト参加してもらうことにする。

■多様化するWeb表現への対応

この表題はいささか大げさすぎる感じだが、現在、インターネット上に存在するWebサイトのほとんどが、画像ファイルやスタイルシートなどのデザイン要素を含んでいる。

w3mは、外部プログラムによる画像処理やCGIなどの動的コンテンツをもサポートした優れたテキストWebブラウザであるし、個人的にも非常に好きなソフトウェアなのだが、表現力という点でグラフィカルなブラウザに及ばないのはいたしかたない。残念ではあるが、今回のところ評価の対象からは外れてもらおう。

■Webブラウザではない

Nautilusは、あくまでファイルマネージャであって、Webブラウザとしての役割がメインではない。今回使ってみた範囲でも、常用のブラウザとして使うのは、ちょっとつらい感じがした。今後、Webブラウザとしての機能を強化する必然性も特にないように思う。ファイル管理機構のさらなる強化のほうが「なすべきこと」だろう。

Konquerorもファイルマネージャではあるが、こちらはWebブラウザとして機能することを前提として設計されたものと思われる。紹介ページでも触れたように、単体のWebブラウザとして見た場合でも必要十分な機能を備えている。検討対象として残しておくことにしよう。

■Mozillaか？Netscape 6か？

この2つのブラウザは、もともとは同じソースコードをもとに開発されている。機能や操作性もほぼ同等ということもあり、どちらか一方を検証していけばいいだろう。ここはNetscape 6といくのが「筋」である。

前にも述べたように、マイルストーンビルドとして配布されているMozillaは、あくまでテストを目的としたものである。一方のNetscape 6は正式リリース版だ。ソフトウェアとして評価するなら、やはり後者を対象とするのがフェアであろう。

ということで、メインの比較対象は、Galeon、Konqueror、Netscape 6としたい。なお、今回の検証のベースとしたディストリビューションは、Red Hat Linux 7.1である。

レイアウトの再現能力

作者が意図したとおりにWebページを再現する能力は、Webブラウザに欠

画面1　日本語に未対応のOpera
この画面は弊社のWebサイトのトップページをブラウズしたところ。完全に文字化けしてしまっている。

かせないものである。基本となるマークアップ言語の構文解析、画像ファイルやフォントのレンダリング、プラグインによる特殊なコンテンツの処理などが機能として求められるのだ。

■サポートするHTMLのバージョン

今回、評価の対象とした3つのブラウザは、すべてHTML 4.0をサポートしている。また、スタイルシートについても、CSS 1とCSS 2を三者ともサポートしている。ただし実装の違いによっては、同じ標準規格をサポートしているブラウザでも、結果として表示されるページに違いが出るということが結構あるものだ。まずは、そのあたりから確認してみよう。

画面2から画面4は、それぞれのブラウザで同じHTMLファイルを表示した結果だ。3つの画面を見てまず気がつくのはフォントの微妙な違いだ。もちろん、テスト条件を揃えるためにフォントは同じ設定にしてある（日本語文字セットの標準フォントがDynaフォントのdfgothicp、欧文文字セットの標準フォントがAdobeのTimes）。また、HTMLでフォントフェイスは指定していない。

●Galeon

タイトル部分の欧文フォントと和文フォントのバランスがとても悪い。そのほかの部分でも欧文フォントの文字送りが若干詰まり気味だ。

●Konkueror

欧文と和文のバランスは良いが、ほかの2つに比べると、行送りやマージンの幅がやや大きい。また、見出しと本文の間の改行が無視されている。

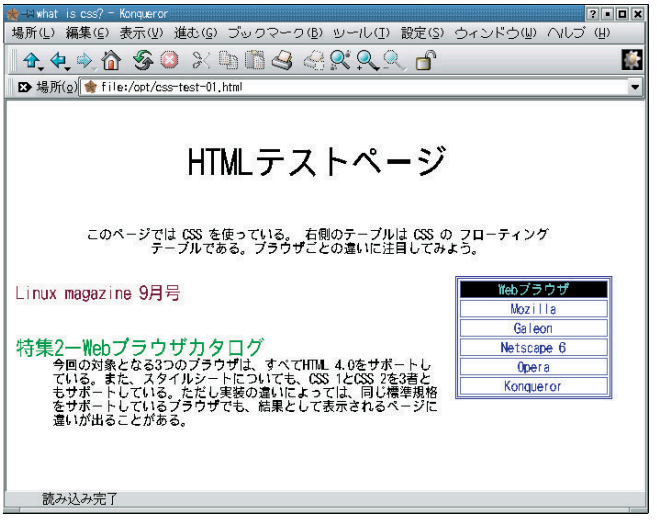
●Netscape 6

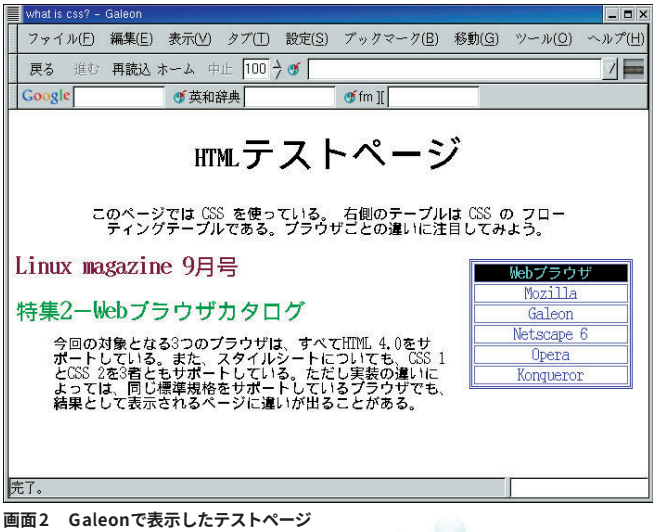
欧文と和文のバランスが良く、全体として見やすい表示だ。ほかの2つと比べるとテーブルの幅がなぜか少し狭くなっている。

ページの右側にあるテーブルは、常にWebブラウザの描画領域の右端にくるように、CSSを使って指定している。また、やはりCSSのスタイル指定で、テーブルの右端と描画領域の右辺との間のマージンも明示的に指定してあり、ウィンドウの大きさにかかわらずテーブルが一定の位置に表示されるのが正しい。この点については、3つとも正しく解釈してレンダリングしている。

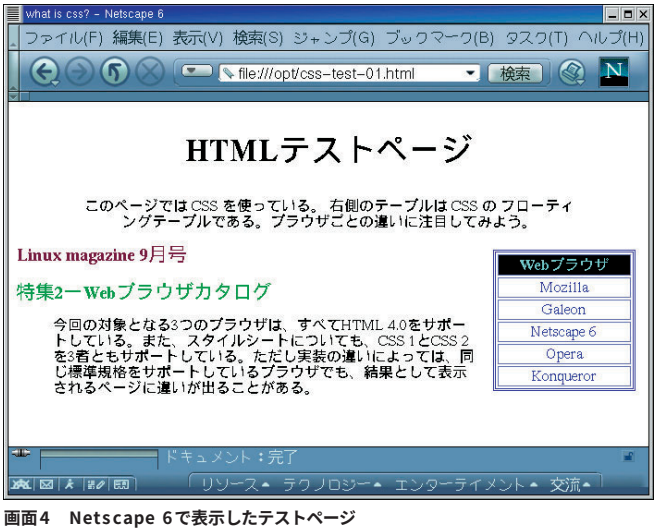
CSSは非常に機能が多いので、これだけのテストでは何とも言えないが、サポートの一端は確認できたのではないだろうか（ちなみに、CSSを完全にサポートしていないWebブラウザで同

画面3　Konquerorで表示したテストページ

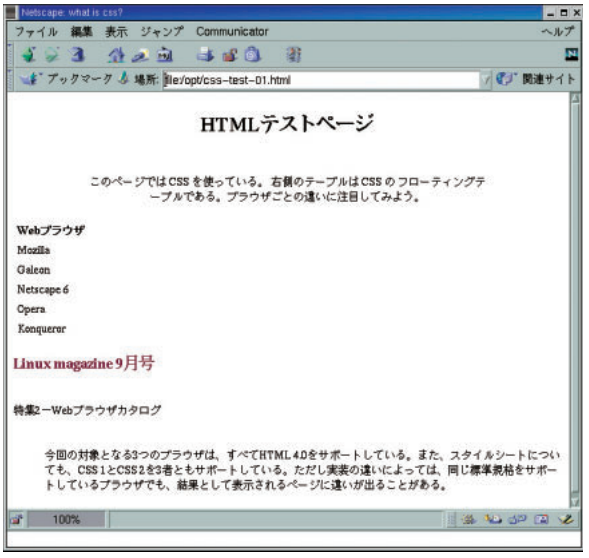




画面2　Galeonで表示したテストページ



画面4　Netscape 6で表示したテストページ



画面5　Netscape Communicator 4.76で表示したテストページ
前ページにあるほかのブラウザによる表示とはまったくことなる結果になってしまう。

じHTMLを表示すると**画面5**のようになってしまう)。

結果として、バランスの良いレイアウトでNetscape 6がやや上回った印象がある。ただし、ほかの2つにも極端な減点材料はなく、及第点はクリアしたといったところだ。

■Macromedia Flash

バナー広告や企業のWebサイトなどで、よく使われるコンテンツのひとつにMacromediaのFlashがある。特殊なコンテンツの代表例として、各ブラウザで正しく処理できるかを検証してみよう。

Macromediaでは、Linux版のNetscape Communicator用のFlash Playerプラグインを無償で配布している。同社のWebサイト（http://www.macromedia.com/jp/）から入手できるほか、ディストリビューションによっては、標準でインストールされている場合もある。今回テストに使用したRed Hat 7では、Flash 4用のプラグインがNetscape Communicatorのパッケージに含まれていたが、検証作業は最新バージョンのFlash 5にアップグレードして行うことにした。

Netscapeのプラグインのセットアップは、必要なファイルをpluginsディレクトリにコピーするだけで完了だ。Flash Playerの場合は、libflashplayer.soとShockwaveFlash.classという2つのファイルを/usr/local/netscape/plugins/にコピーすればよい。フルパスは、Netscape本体がインストールされているディレクトリによって異なるので注意すること。GaleonはMozillaのプラグインを利用しているため、/usr/lib/mozilla/plugins/にも2つのファイルをコピーした（このパスもインストール環境によって違いがある)。

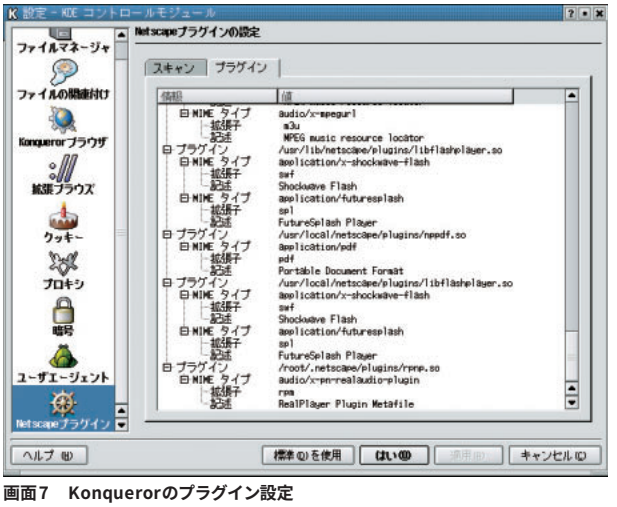
テストの結果は以下のようになった。

●Galeon、Netscape 6
Flash 4、Flash 5とも問題なく動作。

●Konqueror
Flash 4のみ動作。Flash 5のコンテンツを含むページは表示することもできなかった。

KonquerorでFlash 5のコンテンツが動作しなかった原因については、まったく謎のままである。アクセスすると、すぐにページがブランクになり「読み込み完了」状態になってしまい、あとは何

画面6　Galeonで表示したFlash 5のコンテンツ
動作はまったく問題ない。右端にあるカラーパレットから色を選んで車の色を変える機能もきちんと動作した。



画面7　Konquerorのプラグイン設定
少しわかりづらいかもしれないが、Netscape 6のインストールディレクトリ下にあるプラグインを参照している。

も起こらないのだ。Konquerorの設定ツールでチェックしたところ、Netscape 6で使っているのと同じプラグインを参照している（**画面7**）のだが……。

タブ機能でリードするGaleon

Webブラウザのユーザーインターフェイスは、複数表示したWebページをタブで切り替えられる、いわゆる「タブブラウザ」の登場によって大きな転換点を迎えた。少し大げさに聞こえるかもしれないが、それくらい従来のブラウザとは操作性に大きな違いがあるのだ。

Webブラウザを使う目的のひとつに、ネット上に公開されている数多くの情報の中から自分がほしいと思うものを効率よく探し出し、それにすばやくアクセスするということがあると思う。また、ひとつの情報が掲載されたWebページから、別のページの関連情報が見つかったりもする。タブブラウザは、こういったケースで最も威力を発揮する。具体的な例として、メーリングリストや掲示板のアーカイブで情報を探す場合を考えてみよう。

まずは、トップページの検索機能を使って、関連のある投稿のみに絞り込むことから始まる。続いて、検索結果のページが表示される。その中に望みの情報が含まれているかどうかは、検索された投稿に目を通していくしかない。タブ機能があれば、それぞれの投稿を別々のページで開いて、タブで切り替えながら相互に参照できる。

タブ機能がなければ、［戻る］とか［進む］とかいったナビゲーションボタンでページを行き来するか、新しいウィンドウで開いてウインドウを切り替えながら参照していくことになる。ナビゲーションボタンを使う場合、何回ボタンをクリックすれば、さっき見ていた投稿に戻れるのかわからなくなったりするだろうし、一度の操作で望みのページに切り替えることもできない。複数のウィンドウを開く方法は、タブ機能に近いイメージがあるかもしれないが、2つや3つならともかく10を超えるウィンドウが開いたデスクトップをかき分けて目的のウィンドウにたどり着くのは、これまでの経験に照らし合わせても結構大変なことだ。

さて、長々とタブブラウザの優位性を説いてきたわけだが、この特集で紹介したブラウザでタブ機能をサポートしているのは、GaleonとOperaだけである。中でもOperaは、タブ機能と前半のパートで紹介した「Hotlist」機能の組み合わせによる、柔軟で操作性に優れたユーザーインターフェイスを備えている。かえすがえすも日本語が処理できないのが非常に残念である。

Linuxで動作して、日本語が処理できるタブブラウザとなると、今のところGaleonしかない。複数のWebページをブラウズする際の操作性という面では、ほかの日本語対応ブラウザを1歩も2歩もリードしているといっていい。

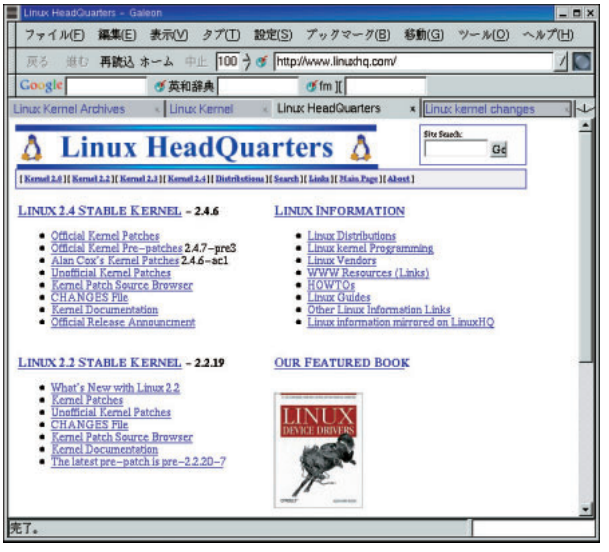
ユーザーインターフェイス再考

タブ機能がないからといって、KonquerorとNetscape 6を切り捨ててしまうのもどうかと思うので、ユーザーインターフェイスについてもう少し考えてみよう。

タブ以外でWebページのナビゲーションをサポートする機能といえば、先ほど少し触れたナビゲーションボタンと、Webブラウザの基本的な機能であるURLのブックマークが挙げられる。Netscape 6では、この2つの機能を強化することで操作性を高めている。

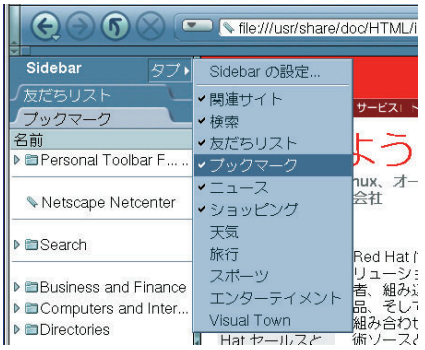
Netscape 6のナビゲーションボタン

画面8　新規タブを開くポップアップメニュー
このように、リンクを右クリックして新しいタブページとして開くことができる。



画面9　複数のタブを開いたところ
タブをクリックするか、タブの列の右端にある矢印ボタンを使うことでページを行き来することができる。

画面10　［戻る］ボタンのもう1つの機能
ページタイトルのみの表示だが、目的のページが見つかれば一度の操作でそれを表示できる。



画面11　［Sidebar］へのブックマークの登録
こうして登録しておけば、ブックマークにアクセスしやすくなる。一度試してみよう。

には、下向きの三角形をした小さなボタンが付いており、これをクリックすると**画面10**のように「戻る」ボタンなら現在のページの前に表示されていたWebページのリストが表示される（もちろん「進む」ボタンはその逆）。いわば、簡易版の履歴保管のような機能である。

ブックマークに関しても、ちょっとした工夫が見られる。「Sidebar」にブックマークを登録してツリー表示できるようになっているのだ。デフォルトではオフになっているので、Sidebarの［タブ］ボタンをクリックして表示されるメニューにある「ブックマーク」をチェックしておこう（**画面11**）。

Konquerorにも同様の機能を持つ、こちらはカタカナ表記の「サイドバー」があり、デフォルトでブックマークと履歴が登録されている。

これらの機能を使いこなせば、Webナビゲーションの操作性がずいぶん良くなるだろう。

フォントの話

レイアウトの再現性のところでも取り上げたフォントについて、もう少し詳しく検討しておきたい。

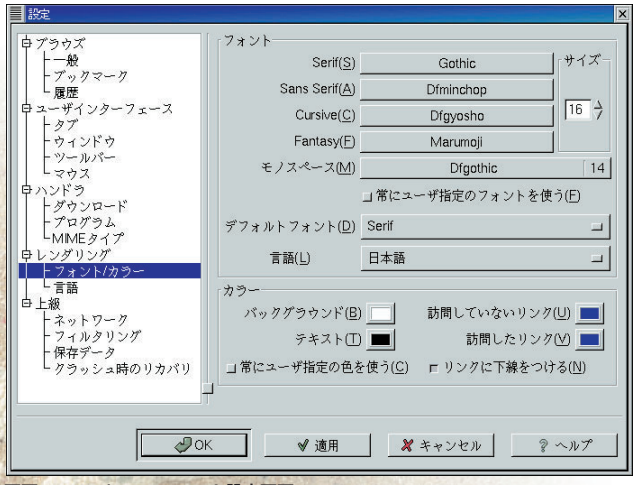
優れたデザインのWebページや貴重な情報を提供してくれているサイトなどを訪れていても、表示されるフォントが汚いと長時間見る気にはとてもなれないだろう。Webブラウザのユーザーにとって、フォントの問題はかなり重要なポイントなのだ。

ブラウザ側に設定可能なフォントオプションの数は、それぞれに異なっていて、Galeonが5つ、Konquerorが6つ、Netscape 6が3つとなっている（**画面12**～**画面14**）。これらは、各文字セットごとに指定することができる。もちろん、指定できるフォントの種類はシステム環境によって異なってくる。また、ブラウザ側で基本となるフォントサイズを変更することもできる。

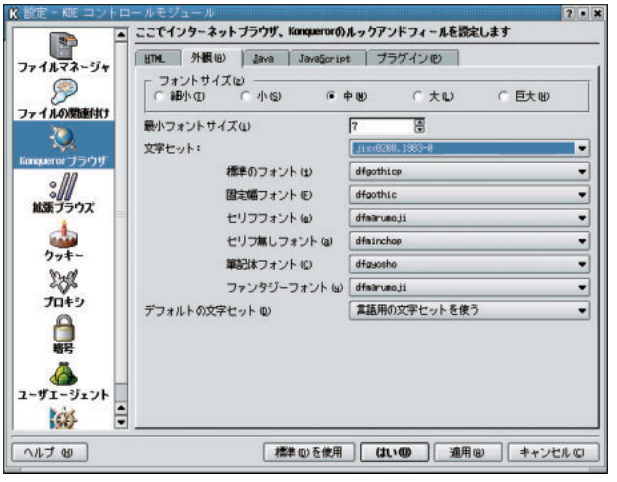
Webブラウザは、これらのフォントオプションと、HTMLに記述されている文字セット情報、フォント情報をもとに、どのくらいのサイズで、どのシステムフォントを使ってテキストを表示するかを決定する。

つまりWebページの作者がフォントフェイスとして“M○ゴシック”と指定していたとしても、そのフォントがシステムになかったり、ブラウザが解析できなかったりした場合には、適切なフォント情報に置き換えられるため、最終的にユーザーが目にするフォントは、そのユーザーのシステムにインストールされているディスプレイフォントのどれかということになる。

いろいろなサイズのテキストをすべて、きちんと表示するには、TrueTypeフォントなどのスケーラブルなフォントを利用するしかないが、



画面12　Galeonのフォント設定画面



画面13　Konquerorのフォント設定画面

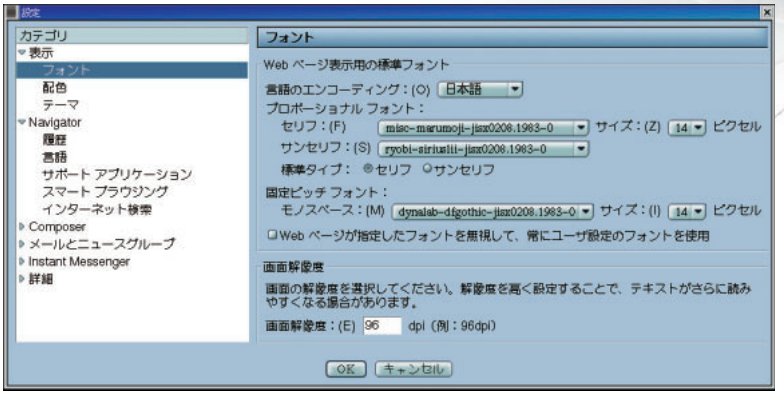
固定サイズのフォントを使っている場合にも、Webブラウザのフォントオプションを工夫することで、かなり美しさが違ってくる。**画面16**と**画面17**は、同じフォントを使って同じWebページを表示している。かなり見栄えが違うが、変更したフォントオプションはサイズだけである。

フォントについては、Webブラウザ自身というより、それが動作するシステム環境に依存する。X Window System（XFree86）、ウィンドウマネージャ、統合デスクトップ環境を適切にセットアップして、きちんとしたフォント利用環境を整えておかなければならない。必要とされる知識も多くなる。誌面の都合上、残念ながら詳しく解説できないが、ここはひとつお気に入りのWebブラウザでフォントに関する情報を探してみてはどうだろうか。

まとめ

ここまで説明した機能面以外では、「安定して動作するか」ということも重要なポイントだ。

画面14 Netscape 6のフォント設定画面



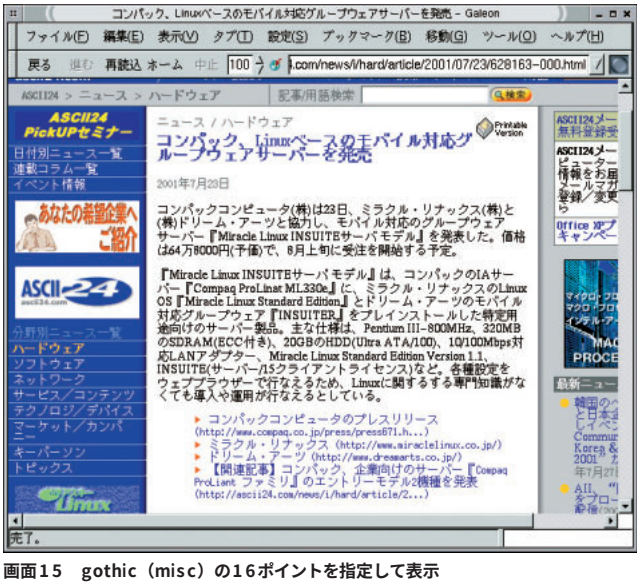
テストで使用した範囲内では、Netscape 6が最も安定して動作した。次にGaleon、そしてKonquerorという順であった。Konquerorは、今回紹介したそのほかのブラウザをすべて含めても、堅牢性という面ではかなり劣っている印象を受けた。

ただ、GNOMEのデスクトップ環境でテストを行ったそのほかのブラウザには、そう大きな差異がなかったことから考えると、KDEか、あるいはQtライブラリの設定になにか問題があったのかもしれない（OperaはQtライブラリをスタティックにリンクしたバージョンを使用した）。

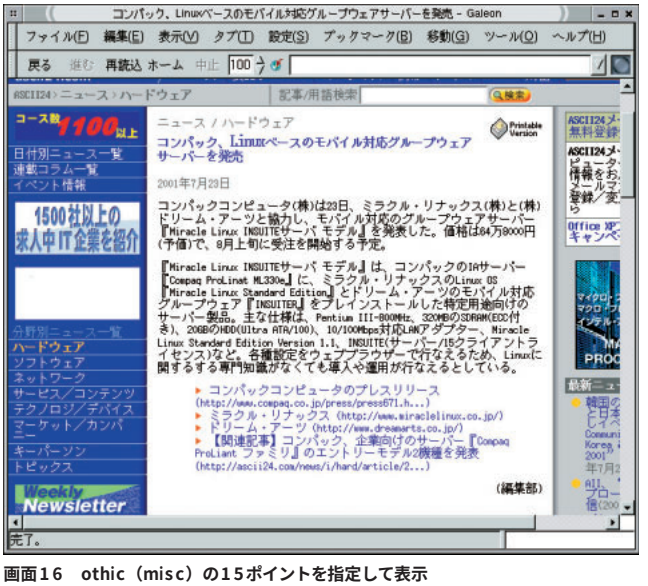
■マイ・ベストワンは？

読み進めてくれた読者の皆さんは、すでに私の選ぶベストブラウザがおわかりのことと思う。ふだんからタブブラウザを使っている身として、やはりその機能は捨てがたいということで、お勧めのWebブラウザは「Galeon」ということにしたい。

次点はNetscape 6。この2つのブラウザのベースであるMozillaと、それを生み出したMozilla.orgのみなさんに大感謝しつつ、Mozilla 1.0のリリースを祈りながら、今回の特集を終えることにしよう（Operaの日本語対応も期待してます＞Opera Software様）。



画面15　gothic（misc）の16ポイントを指定して表示
固定サイズのフォントの場合、ある大きさを超えると見出しの部分のように文字が崩れてしまう。



画面16　othic（misc）の15ポイントを指定して表示
標準のフォントサイズを小さくすることで、大きな文字の崩れをある程度抑えることができるのだ。

レーザーファイブ L-Card+

カード型Linuxマシンで遊ぼう

第1回　L-Card+の紹介

文：吉田 功
Text : Isao Yoshida

「L-CARD+」は、レーザーファイブから発売されている、名刺サイズでLinuxが動いてしまう組み込み用のコンピュータです。この小さなサイズの基板に、イーサネットインターフェイスと、コンパクトフラッシュスロットを内蔵し、多彩な機能を持つマイクロプロセッサを使用していて、いろいろなことができそうです。そこで、数回に渡って、L-Card+で遊んでみようと思います。

L-Card+ってなに？

L-Card+は、名刺サイズの超小型Linux搭載ワンボードマイコンです（**写真1**）。サイズは91mm×60mm、重さはCF（コンパクトフラッシュメモリ）を含めて50g程度です。消費電流も100mA程度なので、電池での動作も可能です。

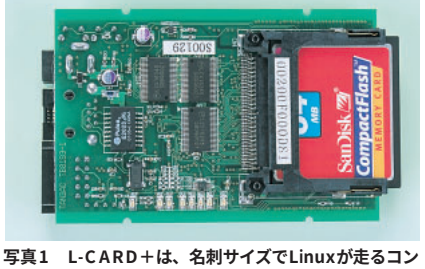


写真1　L-CARD+は、名刺サイズでLinuxが走るコンピュータ

ディスプレイやキーボード、マウスは付いていませんが、シリアルインターフェイスをPCにつなぎ、通信ソフトでアクセスすると、rootユーザーでログインすることから始まり、/binや/etcなど、見慣れたLinuxのファイル構造を見ることができます。小さいながらも、立派なLinuxマシンなのです。

工作機械、自動販売機、計測機器など、特に無人化された機器を遠隔操作して制御したりデータ収集するようなシステムに組み込むといった用途に威力を発揮します。ほかには、オリジナルの情報携帯端末にしたり、Webサーバやライブカメラなどにも使えそうです。

制御用のワンボードマイコンのOSにLinuxを使うのはもったいないようにも思えますが、遠隔操作で制御や通信をさせる場合、LANなどのネットワークを使用することを考えると、プロトコルスタックをデフォルトで搭載しているLinuxを使えばとても簡単にプログラミングできます。もちろん、インターネットへの接続も簡単です。

■L-Card+のスペックは？

L-Card+は、とても小さな基板ですがいろいろな機能を持っています。**表1**は、カタログに書かれている主な仕様です。

マイクロプロセッサには、VR4181（**写真2**）というMIPS互換のアーキテクチャを持つデバイスが使われています。これは、Windows CEなどの携帯端末で使うことを考えて作られたプロセッサで、LCDコントローラ、キーボードインターフェイス、CFコントローラなどをプロセッサ内部に持っています。

VR4181のコアには、64ビットCPUであるVR4110が使われていますが、VR4181は32ビットモードで動作しま

す。クロックは66MHzで、内部にキャッシュメモリを8Kバイト持っています。また、浮動小数点演算命令には対応していませんが、そのぶん、消費電力が少なくなっています。

L-Card＋は、主記憶メモリとして16MバイトのSDRAMを搭載しています。重いアプリケーションでなければ十分使えそうです。また、2Mバイトのフラッシュメモリ（FROM）を搭載していて、システムの起動に使われています。具体的には、この中にLinuxのカーネルを組み込んであり、CFをマウントさせ、アクセスできるだけのプログラムが書かれています。

10BASE-Tに対応するイーサネットインターフェイスも1つ持っています。こんな小さな基板なのに、ちゃんとRJ45のモジュラコネクタが付いているので、使い勝手はとても良好です。

写真2　NEC VR4181
MIPS互換のRISCアーキテクチャを採用したポケットPC用の多機能マイクロプロセッサ。

シリアルインターフェイス（RS-232C）は、PCなどの端末につないでL-Card＋を操作するのに使います。さすがに、このコネクタはRS-232Cの規格とは違うものを使用していますが、PC用シリアルインターフェイス（9ピンのD-SUBコネクタ）に変換するケーブルが付属します。このシリアルインターフェイスは、システムコンソールとして使われ、デフォルトで115200bps、最大値は230Kbpsまで対応します。

また、基板上にCFのスロットコネクタを持っていて、CFをハードディスクのように使います。現状では、256Mバイトまでの CFと、1Gバイトのマイクロドライブの動作が確認されているそうです。

拡張コネクタとして、64ピンの小型コネクタが2個実装されています。ISAバス・ライクな信号（VR4181系ISA信号）を持っていますので、いろいろな周辺装置を接続するには都合がよさそうです。また、汎用入出力ポートや、サウンド（アナログ入出力）など、さまざまな信号が出ています。これらは、マイクロプロセッサに内蔵された機能です。

基板上には、LED（発光ダイオード）が8個付いており、そのうちの5個は自分でプログラミングすることで制御できます。残る3つのLEDは、LAN用に2個、FROMのアクセスランプ用に1個が割り当てられています。

電源電圧は3.65～8.5Vまで、広い範囲で使えます。5V／2.3AのACアダプタが付属しますので、普通はこれを使えばOKですが、電池で動作させることもできます。

■L-Card＋の基板

L-Card＋の基板は、多機能なVR4181のおかげでシンプルな構成になっています（**写真3**）。基板の表側（A面）の右にある一番大きなLSIがマイクロプロセッサのVR4181です。その横に2つ並んでいるのが、64Mビットの SDRAMで16Mバイトあります。その上の四角いLSIはイーサネット用のコントローラで、Crystal社のCS8900Aが使われています。LAN LSIの左側には、MAX3232というシリアルインターフェイス（RS-232C）用のICがあります。

左に並んだコネクタ（**写真4**）は、シリアルインターフェイスコネクタ、イーサネットコネクタ、DC電源コネクタです。シリアルコネクタには10ピンのヘッダコネクタが使われていますが、

CPU	NEC VR4181 66MHz
SDRAM	16Mバイト
フラッシュROM	2Mバイト
ネットワーク	10BASE-T ×1（CS8900A）
シリアルポート	RS-232C ×1（10ピンコネクタ）
拡張スロット	コンパクトフラッシュ Type Ⅰ、Ⅱ準拠
拡張コネクタ	64ピン×2 VR4181系ISA信号、LCD、サウンドI/Oなど
LED	LAN用×2、フラッシュROMアクセス用×1、ユーザー用×5
ボードサイズ	91mm×60mm
電源電圧	3.65～8.5V
付属品	ACアダプタ（DC5V／2.3A）、シリアル変換ケーブル、CD-ROM（コンパクトフラッシュ用イメージ、クロス開発環境、資料など）
価格	3万8600円

表1　L-Card＋の主な仕様

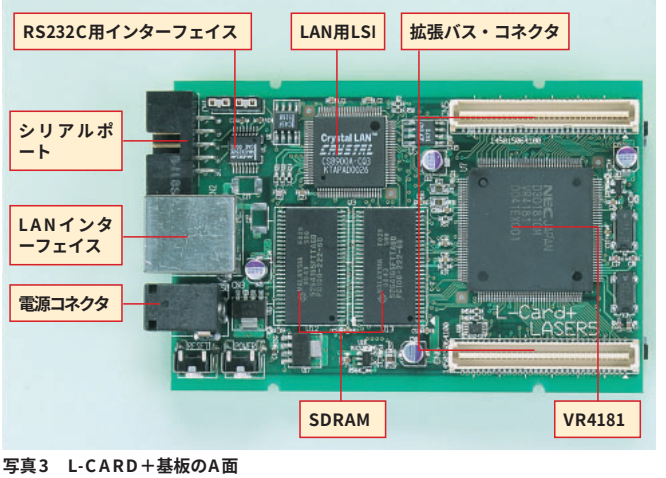


写真3　L-CARD＋基板のA面
VR4181やSDRAM、コネクタ類などが載っている。

PCのシリアルコネクタと同じD-SUB 9ピンのコネクタに変換するアダプタが付属しています。

電源コネクタの下のほうには、2つのプッシュスイッチがあります。「RESET」はプロセッサをハードリセットできるリセットスイッチです、「POWER」はプロセッサがソフト的にパワーダウンしたときに、再スタートするスイッチです（**表2**）。

シリアルコネクタの横には、2つのジャンパスイッチがあります。「JP1」はMIPS16命令の使用許可／不可を設定します。「JP2」はFROMにライトプロテクトをかけるためのもので、ジャンパを挿した状態でプロテクトがかかります（**表3**）。

VR4181の上下にあるコネクタが拡張バスで、64ピンのコネクタが2組あります。シリアル、イーサネット、LED以外で入出力を行うには、このコネクタから信号を取り出さなくてはなりません。

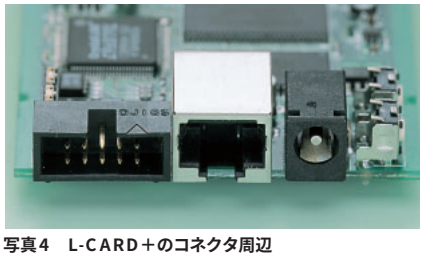


写真4　L-CARD＋のコネクタ周辺
小さな基板に、シリアル（RS-232C）、イーサネット（10BASE-T）、電源のコネクタが載っている（CONN）。

基板裏側（B面）は、**写真5**のようになっています。CF用のスロットの下に、16MビットのFROM（2Mバイト）が1個あります。左側にある、少し厚みのある部品は、イーサネット用のトランスです。真ん中に3つ並んでいるのはバッファ用のICで、高速なSDRAMと、その他の遅いデバイスの間で干渉を起こさないために1個、そしてCF用のバッファ用に2個使われています。

下のほうに並んだ白っぽくて小さな部品がLEDです。LANに接続すると点灯するのですぐにわかりますが、ちょっと見た目には、これが光るとは思えない感じの部品です（**写真6**）。

これらを図にすると、**図1**のようになります。プロセッサは2.5Vと3.3Vの2電源が必要で、周辺のデバイスは3.3Vで動作します。また、スリープなどでパワーダウンさせることができるように、3.3V系の電源を2系統持っています。

L-Card＋で何ができるの？

L-Card＋本体には、イーサネットインターフェイス、シリアルインターフェイス、ユーザーが使用できるLEDが5個ついています（**表4**）。ただし、シリアルインターフェイスはシステムコンソールとして使用するので、ほかの用途には使いにくいようです。

イーサネットが付いているので、Webサーバや、メール処理などによさそうです。また、LEDをON／OFFできるので、アクセスする側から、サーバ（L-Card＋）管理者に、何か伝えるようなことができそうです。もちろん、インターネットに接続されていれば、外からLEDをON／OFFすることもできそうです。

L-Card＋のスペック表には書かれていませんが、VR4181が持つ機能を利用することも可能です。これについては、コラム「VR4181の機能と制限」にまとめておきました。L-Card＋本体でサポートできない機能は、拡張バスを使うことで使用できます。

名称	接続端子	詳細
RESET	#RSTSW	VR4181リセット信号
POWER	POWER	VR4181起動信号

表2　スイッチの内容

ジャンパ	内容	セット	オープン
JP1	MIPS16命令の使用許可／不可	使用可	使用不可
JP2	フラッシュROMのライトプロテクト	する	しない

表3　ジャンパの設定

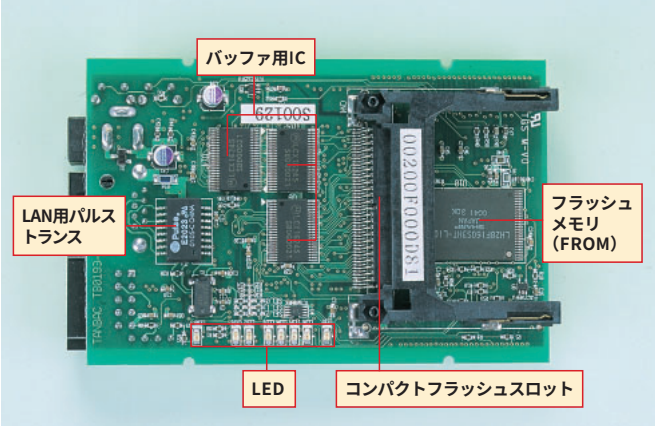


写真5　L-CARD＋基板のB面
CFスロットやフラッシュROM、LEDなどが載っている。

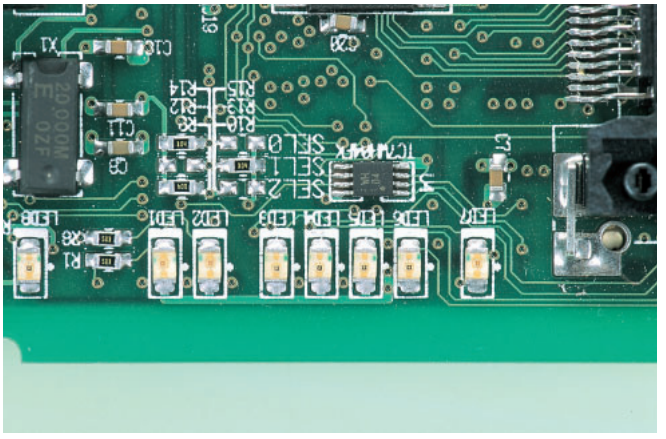


写真6　表面実装部品のLED
8個並んでいる白い部品がLED。

拡張バスに取り付けられそうなものは、最大320×320ドットのLCDユニットとタッチパネル、サウンドの入出力、IrDAです。なかでも、サウンド入出力は使えそうです。MP3の再生や、インターネット電話端末にすることができるかもしれません。

制御用としては、アナログ入力3ch、デジタル入出力5ビット（最大9ビット）も使えます。これだけあれば簡単な制御くらいはできます。やっぱり、L-Card＋は、拡張バスに付加装置を付けてこそ、面白そうなことができるといえるでしょう。

また、ISAライクなバスも持っているので、市販のISAバスカードで使えるものも多くありそうです。もちろん、自分でI/Oインターフェイスを作成することも可能なので、手の込んだ制御にも使えます。カメラを接続してWebカメラなどを作ってみると面白そうに思えます。

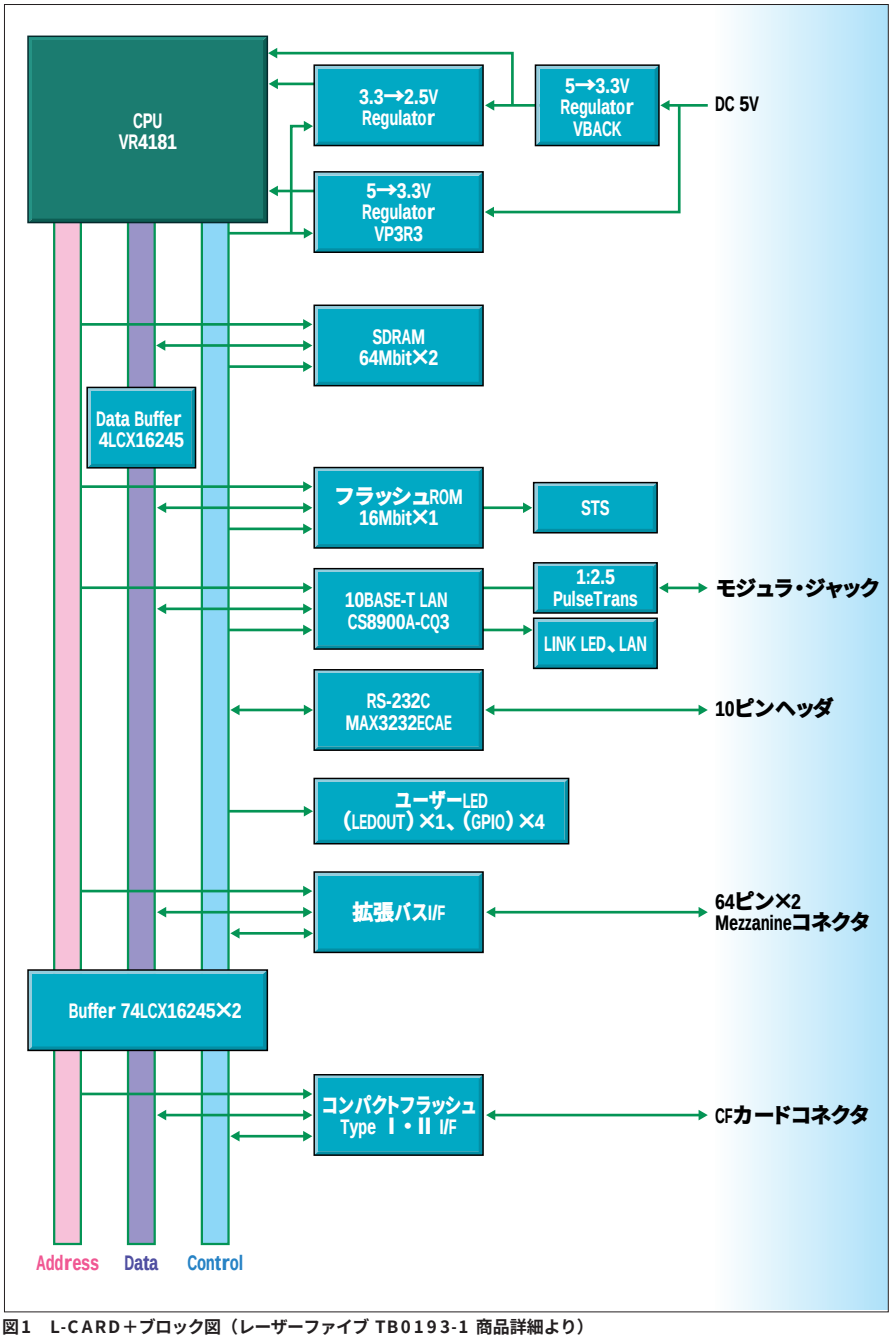


図1　L-CARD＋ブロック図（レーザーファイブ TB0193-1 商品詳細より）

L-Card＋を使ってみよう

L-Card＋のパッケージには、**写真7**のような付属品が付いています。動作を確認するためには、このほかに64Mバイト以上のCF、付属のCD-ROMの内容をCFにコピーできる環境、そして、RS-232Cインターフェイスで115200bps対応の通信端末が必要です。

CFは付属しないので、自分でフォーマットからユーザーランドのコピーまでをしなくてはなりません。これには、Linuxが走るマシンで、CD-ROMとCFが扱える必要があります。Windowsベースのパソコンからはコピーできませんので、Linuxで使えるCFスロットがなければなりません。標準でCFスロッ

LED	色	種別	詳細
LED1	緑	LAN	LINK接続時に点灯
LED2	緑	LAN	LANデータ転送時に点灯
LED3	赤	USER	GPIO 29で制御可能
LED4	赤	USER	GPIO 31で制御可能
LED5	赤	USER	GPIO 8で制御可能
LED6	赤	USER	GPIO 2で制御可能
LED7	緑	USER	LEDOUTで制御可能
LED8	赤	FLASH	アクセス時に点灯

表4　LEDの役割

Column

MIPSⅢとMIPS16

MIPSには幾つかの命令セットアーキテクチャ（ISA）があります。この拡張機能としてMIPS16があります。MIPSⅢは32ビット固定長の命令フォーマットで、MIPS16は16ビット長の命令を使用したフォーマットになっています。このため、MIPS16のほうが少ないメモリでプログラミングできるという特徴があります。

トを持っているのは最近のノートパソコンくらいでしょうか？　最近では、USB接続のCFリーダ／ライタが市販されていますが、Linuxで使用できる製品は少ないようです。PCカードスロットしか持たないノートパソコンや、PCカードアダプタを取り付けたデスクトップパソコンなら、CFアダプタ（**写真8**）を使えばCFを扱うことができます。CFアダプタには機種依存性は少ないようですし、数百円～千円くらいと価格も安いので1つ用意しておくと便利です。

レーザーファイブによると、256Mバイトまでの、SUNDISK製とメルコ製のCFの動作確認ができているそうです。自宅で行ったテストでは、ハギワラ製のCFも扱うことができました。L-Card＋は3.3Vのみの電源しか持っていませんので、5V／3.3V兼用型か、3.3V用のものであれば使えそうな気がします。また、1Gバイトの容量を持つCF互換ハードディスクである、IBMのスマートドライブも使用可能とのことでした。

CFをL-Card＋で利用するための準備は、「L-Card＋で使うCFの作り方」を参照してください。

■いよいよ起動

まず、115200bpsで通信できる端末を用意しなくてはなりません。Windowsのハイパーターミナルなんかでも構いません。ポートの設定は、通信速度＝115200bps、データビット＝8ビット、パリティ＝なし、ストップビット＝1ビット、フロー制御＝なし、とするとよいでしょう。

L-Card＋に付属のシリアル変換ケーブルを付け、通信端末とL-Card＋の間をシリアルケーブルでつなぎます。このケーブルは、D-SUB 9ピンのメスが両方についたクロスケーブルです。ストレートケーブルは使えませんので注意が必要です。

L-Card＋は、CTS／RTSを使ったハードフローには対応していますが、DSR／DTRは使用されていません。使用するクロスケーブルによって、フロー制御線の配線が異なるようなので、CTS／RTSが有効なケーブルを使用する場合は、通信端末の設定を、RTS／CTSフローにするとよいでしょう。

はやる気持ちを抑えつつ、CFをL-Card＋にセットし、シリアルケーブル

写真7　L-CARD＋の付属品
これ以外に、64MバイトのCFとCFが扱える環境を持ったLinuxマシンが必要だ。

写真8　コンパクトフラッシュアダプタ
PCMCIA（PCカード）スロットでCFが使えるようになるアダプタ。CFが扱えないパソコンで使うときには必須。

Column

VR4181の機能と制限

VR4181は実に多くの機能を持っていますが、L-Card＋本体で使用しているのは、この中の、ほんの一部の機能だけなのです。しかし、L-Card＋でも拡張バスを使えば、これらの機能を使うことができるように設計されています。

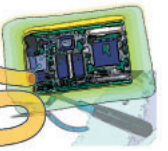
VR4181が持つインターフェイス（コントローラ）には、LCDコントローラ、コンパクトフラッシュコントローラ、キーボードインターフェイス、シリアルインターフェイス（2ch）、IrDAインターフェイス、タッチパネルインターフェイス、A/Dコンバータ（8ch）、D/Aコンバータなどです。このように、多くの機能を持ったVR4181ですが、ピン数の制限から、同じピンに複数の機能が割り当てられています。このため同時に使えない機能もあるのです。たとえば、コンパクトフラッシュコントローラとキーボードインターフェイスは、どちらか一方しか使えません。L-Card＋はコンパクトフラッシュ用になっているので、キーボードインターフェイスは使えないことになります。

汎用I/Oポートも本来は32個持っているのですが、機能と重複するため、拡張バスから使用できるのは5個しかありません。しかし、LCDを使わなければ、もう4個増えます。

シリアルインターフェイス1も、全部で7本の制御線が用意されているのですが、L-Card＋では、DTR、DSR、DCDをLEDの点灯などの用途に使用しているので使えません。また、シリアルインターフェイス2はIrDAと共用なうえ、CTSなどの制御線がLED用に使われているため、拡張バスに信号が出力されていません。

A/Dコンバータは10ビットの分解能を持つものが全部で8個あります。このうち4個はタッチパネルインターフェイスに、1個はオーディオ入力用に割り当てられています。残りの3個が汎用として使えます。また、D/Aコンバータはオーディオ出力用になっています。

これらから考えると、拡張バスに接続して使えるインターフェイス、コントローラは、LCD、タッチパネル、オーディオ入出力、IrDA（またはフロー制御無しシリアルインターフェイス）、汎用アナログ入力（A/D）が3ch、汎用デジタル入出力（GPIO）が5個のようです。ほかには、ISAライク（ISAバスのように使える）バスが使えますので、独自のインターフェイスを取り付けたり、市販のISAバスカードを使える可能性があります。



を繋ぎます。端末の通信ソフトの準備ができたら、電源を入れてみましょう。電源を入れ、画面に「waiting...」の文字が表示されれば成功です。しばらく待っていると、見慣れたLinuxの起動メッセージが出てきます。これを見ると、カーネルのバージョンは2.4.0-test9のようです。PCMCIAカードサービスを組み込んだあとCFが認識されますが、CFが挿さっていなかったり、フォーマットやユーザーランドのコピーに失敗していると、カーネルパニックで起動に失敗します。うまく認識されれば「login:」が表示されますので、rootでログインしてみましょう。このときパスワードは必要ありません。

■ネットワークからtelnetしてみよう

デフォルトの状態では、起動時にイーサネットに接続されていないと「eth0」がアクティブになりません。ネットワークから使用するときには、モジュラコネクタにイーサネットのケーブルを接続してから電源を入れます。この状態で起動すると、Linux起動時のメッセージの最後に「eth0」を割り当ててくれます。

ifconfigで見るとわかるように、IPアドレスは、192.168.128.104が割り当てられています。もし、テストするLAN環境のネットワークアドレスが、192.168.128.XXでなかったら、うまく通信できないかもしれません。すでに構築されているLAN環境に接続する場合は、コンソール（シリアルインターフェイス）からL-Card+のIPアドレスを変更します。

ノートパソコンなどと、1対1（ピアトゥピア）で接続テストをする場合は、パソコン側のIPアドレスを変更してしまったほうが簡単かもしれません。この場合は、クロスケーブルが必要です。L-Card+のIPアドレスを変更するには、/sbin/etherupを書き換えます。接続試験程度で、特に必要がなければデフォルトゲートウェイは設定しなくてもよいでしょう。

うまくIPアドレスが設定されたか、ホストマシン側からL-Card+に向けてpingでテストしてみます。192.168.1.10に変更したなら次のようにします。

```
$ ping 192.168.1.10
```

pingで確認したら、いよいよtelnetしてみます。以下のようなログインメッセージが表示されれば成功です。

```
Linux 2.4.0-test9 ((none)) (ttyp0)
(none) login:
```

■L-Card+を使ってみると

今度はCFの中を見てみることにしましょう。64Mバイトの CFの使用状況を

▼L-Card+で使うCFの作り方

最初の仕事はCFのフォーマットです。L-Card+は、CFをLinux標準のext2フォーマットで使います。そのため、CFをLinuxでフォーマットしなくてはなりません。CFはデジタルカメラやWindowsパソコンで使うことが多いために、通常はDOSのフォーマットになっています。そのため、Linuxのfdiskで、DOSパーティションを削除し、新たにLinuxのパーティションを設定し、ext2のフォーマットに変更します。

```
# fdisk /dev/hde      ← CFのディスクプリタ“hde”は環境によって異なる
# mkfs.ex2 /dev/hde1  ← ext2フォーマットにする
```

次は、CD-ROMの中にあるユーザーランドをCFへコピーする作業です。Linuxマシンで作業用ディレクトリを作り、CD-ROMをマウントします。ユーザーランドの圧縮ファイルを作業ディレクトリにコピーし、そのディレクトリの中に解凍します。

```
$ mkdir CFwork                                  ← 作業用ディレクトリを作成
$ mount /mnt/cdrom                              ← CD-ROMをマウント
$ mv /mnt/cdrom/CL+0216V01CDimg/boot_laser5_20010302.tar.gz CFwork
$ cd CFwork                                     ← 作業ディレクトリに移動
$ tar zxvf boot_laser5_20010302.tar.gz          ← 圧縮ファイルを展開する
```

今度は、先ほどフォーマットしたCFをマウントし、展開したファイルをディレクトリ構造ごとコピーします。

```
$ mount /dev/hde1 /mnt/floppy                   ← CFをマウント
$ cd boot_laser5_20010302
$ cp -dpR * /mnt/floppy
```

最後に、CD-ROMとCFをアンマウントして終了です。

```
$ umount /mnt/cdrom
$ cardctl eject     ← これを実行し、アクセスランプが消えるまでCFを抜いては駄目
```

dfコマンドで見ると、約43Mバイト使われています。ワンボードマイコンの割には、たくさんのものがインストールされているように思えます。

あちこち,ディレクトリを見ていくと、/usrにはX11R6ディレクトリがあります。この中には、しっかりとX Window System関連のファイルが詰まっていました。L-Card＋自体はLCDユニットを持っていませんが、外付けしたときにいつでも使えるような環境は整っているようです。逆に言うと、LCDを使う予定がなければ、X関連のファイルは全部削除してしまっても問題はなさそうです。/usr/X11R6の中身を削除してみたところ、使用量は約28Mバイトとなり、32MバイトのCFに納めることができました。

試しにTCPのポートスキャンをかけてみたところ、telnetやFTP、SMTPなど、14種類のサービスが動いているようです。思っていたよりも多くのサービスが起動していました。

shutdownコマンドを使うと、L-Card＋を停止させることができます。停止したあと、「POWER」スイッチを押すと、再起動しそうに思うのですが、なぜか二度と起動しません。「RESET」スイッチも効きませんので、ACアダプタをいったん抜いて、電源を再投入するしかありません。

```
$ shutdown -h now
```

ちなみに、FROMに書き込み中でなければ、いつでも好きなときに電源を抜いたりリセットしてしまっても問題ないそうです。となれば、わざわざshutdownさせなくてもよいのでしょう。

■オプションのいろいろ

L-Card＋にはいくつかのオプションが用意されています。まず、**写真9**のコネクタ（型番　LC＋KRCN1S）は、拡張バス用に使われている64ピンのコネクタの対になるコネクタです。京セラエルコ製の型番「24 5015 064 105」というものなのですが、一般では、なかなか入手しにくいようなので、レーザーファイブから販売されています。L-Card＋で何かを制御したり、サウンドを扱ったりさせるならば、拡張バスにこれらの回路を付加するために必ず必要になるものです。ただし、このコネクタは基板に半田付けするタイプで、さらにピッチが細かく半田付けの難易度が高いものです。おまけに、このコネクタにピッタリ合う拡張用基板は売られていません。残念ながら、自分で基板を設計し製作できないと活用できません。このコネクタを取り付けたL-Card＋用のユニバーサル基板が発売されることを期待しましょう。

写真10の基板は、L-Card＋に搭載されたFROM上のプログラムを開発するのに便利な「ROM Board for LC」というものです。起動時に、L-Card＋上のFROMではなく、この基板上のEPROMのプログラムを実行できます。この基板があれば、FROM上に有効なプログラムがなくても、EPROMにモニタプログラムを書き込み、それを使ってFROMに有効なプログラムを書き込むことができます。なんらかのミスでFROMの内容を壊してしまったときにも便利です。また、EPROMの代わりにROMICE（ROMエミュレータ）を使うこともできます。FROMの書き換えは、この基板がなくとも、RAM上にモニタプログラムを置くことでできますから、普通に使っていれば必要になることはないでしょう。L-Card＋に、別のOSを載せてみるとか、もっと高度なことをするときにあると便利な基板です。

■開発環境

L-Card＋は、CPUにx86系ではないVR系のデバイスを使用しています。このため、x86系のプログラムをバイナリでインストールすることはできません。ソースをコンパイルしてインストールする必要があります。付属しているCD-ROMの中には、CFのユーザーランドだけでなく、L-Card＋で使われているカーネルのソースと、x86系の環境で動くVR用クロスコンパイラが入っています。このクロスコンパイラを、開発用のパソコンの中にインストールし、この上でクロスコンパイルしたものを、L-Card＋に転送して実行することになります。もちろん、カーネルだけでなく、一般に配布されているソースもコンパイルして使います。開発環境のレポートについては、次回、詳細を紹介しようと思います。

L-Card＋は、小さいながらも大きな可能性を秘めたワンボードマイコンで

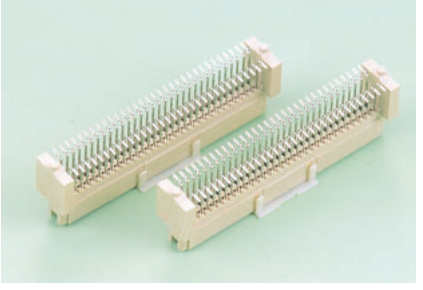


写真9　オプションの拡張バス用のコネクタ
35mmの横幅に64本もの端子がある高密度コネクタ。

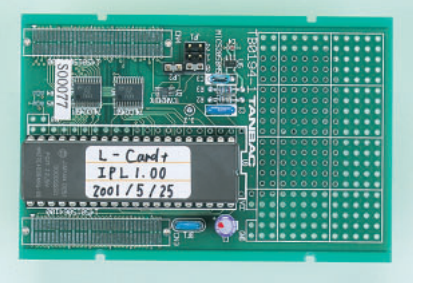


写真10　オプションのROM Board for LC
パワーユーザー用の開発ツール（ROMは付属しない）。

す。消費電力が少なく電池運用も可能、もちろん発熱も少ない。マイクロプロセッサの機能を使うだけでもさまざまなことができそうだし、ISAライクバスを使えば外部の装置も接続できそう。とても面白そうな素材です。ノートパソコンやマイクロPCは拡張性が悪いし、デスクトップパソコンを使うには、なんか大げさすぎてやる気にならない。ちょっと遊んでみようかなと思うには、ちょうどよいサイズとスペックではないかと思います。

Column

VR4181ってどんなマイクロプロセッサ？

L-Card＋に使われているマイクロプロセッサは、NEC製のVR4181（μPD30181）というデバイスです（図）。これは、MIPS社が開発したRISCアーキテクチャを採用しています。Windowsパソコンでお馴染みのインテル系CPUであるx86系のアーキテクチャとはまったく異なるものです。NECのVRシリーズのプロセッサは、NECのモバイルギア2などのWindows CEマシンで使われています。これらのマシンにLinuxを移植しようという計画が、Linux-VRプロジェクトです。L-Card＋は、このノウハウを利用して設計されています。このため、Linuxで主流のx86系のバイナリコードをそのまま使用することはできず、手軽なバイナリインストールはできません。しかし、ソースからクロスコンパイルしてVR用にすることは可能です。

■VR4181の特長

VRシリーズはRISCアーキテクチャを採用しています。RISCの良いところは、CPUの内部構造を簡素化できることです。簡素化することで、チップの面積が減り、値段が安く作れ、さらには消費電力も減ります。大量生産できる安価なポケットPCには好都合です。

VR4181は、ポケットPC向けに作られたデバイスです。そのため、LCDコントローラや、コンパクトフラッシュコントローラ、タッチパネル用インターフェイス、シリアルインターフェイスなどの機能を内蔵しているのが特長です。基本的には、メモリと入出力デバイスを付けるだけで、ポケットPCが完成します。

同じVR4181を使用したPDAに「Agenda VR3」というのがあります（写真）。Palm対抗ということのようですが、これはOSにLinuxを搭載し、フラッシュメモリを書き換えて環境を変更してしまうことができる異色のPDAです。LCDとタッチパネルを載せていますが、コンパクトフラッシュやLANは付いていません。L-Card＋とは方向性がまったく違う製品ですが、使いこなすノウハウはどちらも共通しています。

写真　Agenda VR3
Linuxが搭載された異色のPDA「Agenda VR3」には、L-CARD＋と同じVR4181が使われている。

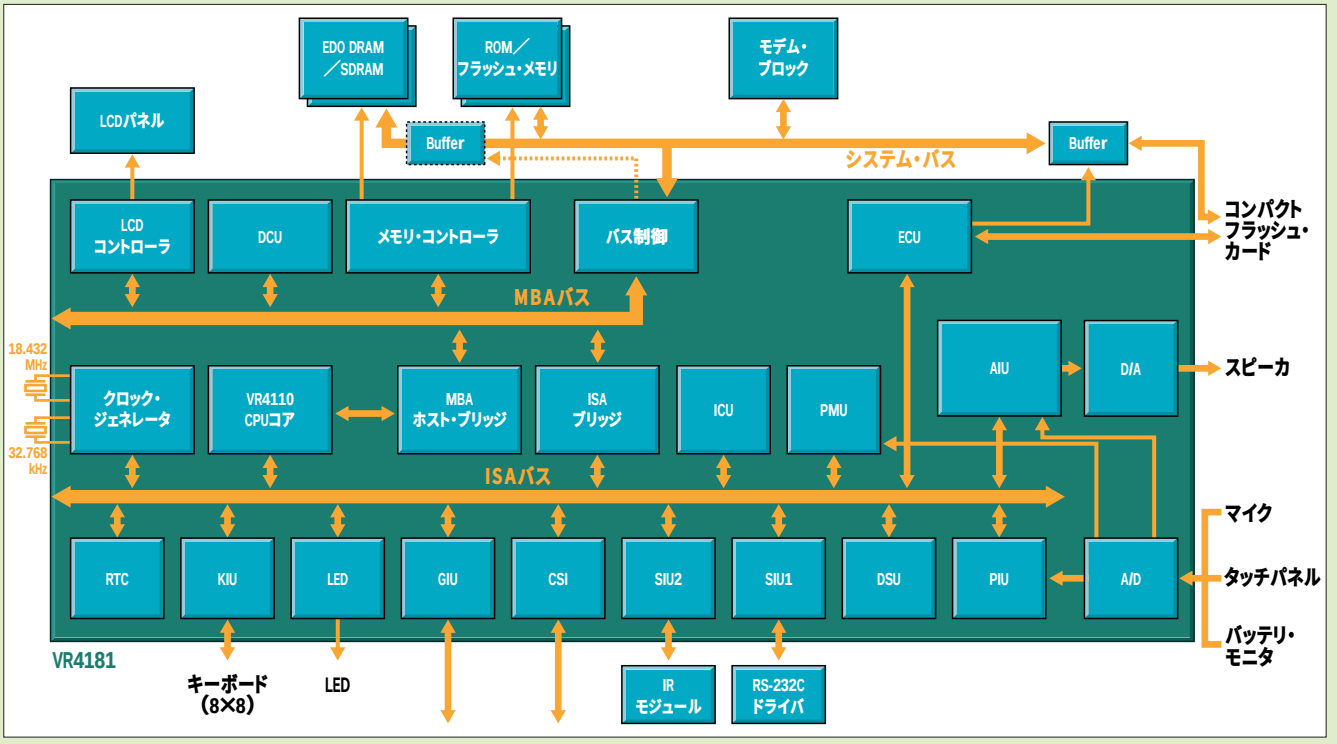


図　VR4181の内部ブロック図（NEC VR4181ユーザーマニュアルより）

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8回　自分たちのスタイルを開発する

Linuxが多くの開発者によって開発される「バザール形式」というスタイルで開発されてきたことは、Eric.S Raymond氏が論文、「伽藍とバザール（山形浩生氏の訳がhttp://cruel.org/freeware/cathedral.htmlで読める）」を1997年に発表して以来、既に公知の事実と言ってよいでしょう。この論文では、従来の中央集権的でトップダウン的なアプローチを「伽藍」に、多数のプログラマーとベータテスターたちがわいわいがやがやとコーディングしたり改良点を見つけたりデバッグをしたりするアプローチを「バザール」にたとえ、それぞれの特徴、特に「バザール」形式で、なぜソフトの開発がうまくいくのかを実証例とともに解説しています。Linuxで開発されたのはOSだけではなく「開発スタイルでもある」と言っても過言ではありません。

もともと開発スタイルは1つではありません。大規模プロジェクトでは、一般に「上流→中流→下流」「ハードウェア→ソフトウェア」といった工程順に開発を行って次の段階へ渡してゆく「ウォーターフォール方式」や、それを途中のトラブルや変更に対応しやすいように変形した方式、短期間でのプロトタイピングを何回も行いながら螺旋状に開発を進めてゆく「スパイラル方式」が採用されることが多いのですが、どのような開発スタイルにも長所と短所、そして限界があります。

それに、従業員数万人の大企業で強固な体制を作っての開発と、従業員数十人～数人の中小ベンチャー企業で、体制の柔軟さを活かしながらの開発では、その企業・その人員・その場面にとってベストといえる開発スタイルは、自ずと異なってくるのが当然でしょう。「話題になっている開発スタイルをありがたく頂戴して現場にはめ込めば、良好な開発効率と成果物が得られる」と信じ込んでいる技術者や管理者を見かけることは現実には珍しくありませんが、開発の現場はそんなに単純ではありません。

今回は「子羊ルータ」で有名なワイルドラボの設立から現在までの経緯、そして「子羊ルータ」の開発とその後のストーリーを通じて、一ベンチャー企業が自分たちにふさわしいスタイルを開発してきた経緯を追ってみたいと思います。

ツールに惚れ込んでの起業

ワイルドラボは1998年に設立されましたが、もともとルータの開発を目的とした起業ではありませんでした。設立者である柳沼由貴子氏・森源明氏の2人が、それまでの勤務先で扱っていたクライアント／サーバ（C/S）システムの構築ツールに「惚れ込み」、そのツールを扱い続けるために会社を設立したのでした。

設立当初は、データベースを中心とした業務ソフトウェアの受託開発が主要な業務でした。通常、設立間もない

SOHOは仕事の確保に苦心し、懸命の営業活動を展開するものですが、ワイルドラボの場合、設立者たちがC/Sシステム全般に明るかったこと、それまでに蓄積された人的コネクションが豊富だったことから、仕事に困ることはまったくなかったそうです。こうして、ワイルドラボは設立当初からSOHOとして順調に実績を積んでゆくことができました。

自分たちの必要から生まれた「子羊ルータ」

ある程度の規模以上の業務ソフトウェアの開発は、複数の企業が協力して開発するのが普通です。このような開発スタイルにはインターネット環境が必須となります。ワイルドラボも、'98年の設立当初からISDNを導入していました。もっとも、現在の「フレッツISDN」のような常時接続サービスは当時存在せず、常時接続といえば、1カ月で3万円以上かかるOCNエコノミーくらいしか選択肢がありませんでした。そこで、「テレホーダイ」や「タイムプラス」といったサービスを活用し、「昼間はメールチェックのみ、夜はWebサーフィン」というスタイルでインターネットを利用し、通信コストの削減に努めていたそうです。複数のマシンからのインターネット利用は、Windows NTのプロキシソフトウェアでIPアドレスを共有することで辛うじて実現していました。

'99年、周辺のケーブルテレビ業者がインターネットサービスを始めるという朗報が舞い込みました。1カ月数千円で、メガバイト単位の帯域が常時利用できるという魅力的なサービスです。これはブロードバンドが当然になりつつある現在でこそ珍しくありませんが、当時としては画期的なことでした。インターネットサービスの開始を待ち焦がれていたワイルドラボの2人が、早速利用を申し込んだのは言うまでもありません。

ところが、複数のマシンで1つのIPアドレスを共有するために利用できる機器は、当時ほとんど業務用の高価な製品に限られていました。SOHOで現実的に導入できるものとして、1台6万円程度の製品が国内に1つだけ存在しましたが、それを購入して使ってみたところ、もともと学校・ホテル・病院といった組織を前提として開発された製品であるということもあり、従業員2人のSOHOでの利用には何かと不便が多かったということです。

当初、その製品をエンドユーザー向けに販売してサポートするというビジネスも考えていたワイルドラボの2人は、「個人やSOHO向けに適切な製品が存在しないのなら、自分たちで作ろう」と考えました。ヒット製品「子羊ルータ」はそこから誕生しました。

2週間で固まった「子羊ルータ」の初期コンセプト

「個人・SOHO向けルータ」という製品を開発しようとワイルドラボの2人が考えたのは、'99年6月のことでした。2人はソフトウェア開発の片手間にマーケティングを行い、2週間ほどで製品コンセプトを固めてしまいました。

といっても、スタートは「ごく普通のパソコンにLinuxをインストールし、NICを2枚装着してIPマスカレードを実現する」という、ごく普通の地点でした。ワイルドラボでLinuxを使ってみたのはこの時が初めてだったとのことです。2人ともUNIXの経験はゼロではありませんでしたが、現役の技術者としては数年のブランクができていました。

この時にインストールしたLinuxディストリビューションは、インストールの容易さに定評のある、メディアラボの「LinuxMLD」でした。「ほかのディストリビューションだったら、インストールで挫折していたかも」とワイルドラボの柳沼氏は述べています。ともあれ、LinuxとIPマスカレードで目的のルータ機能が達成できること、安定稼動の実績、ロイヤリティフリーであること、リソースを消費しないことなどを評価して2人はLinuxの採用を決定しました。

一方でハードウェアの検討も必要です。森氏は趣味で電子工作をやってきたので、ハンダゴテには抵抗がなく、電子工学科を卒業しているのでハードウェアの素人というわけではないのですが、それまで組み込みシステム開発の経験はまったくありませんでした。工業用の小型ボードの存在を知ったのもそのころ、それも一般パソコン誌の記事を見てのことでした。森氏はそれから「組み込みシステム展（ESEC）」を視察して各種ボードを見たり、組み込みシステムでは一般的な「PC104」という規格について知ったりするようになりました。

さらに重要なのは、パーツのメーカーを探すことです。一般的にボードなど部品のメーカーは、過去に実績のない取引先、数量の見込めない取引先との取引を好みません。森氏は目的に見合う構成のボードを販売しているボードメーカー数社にメールで打診してみたのですが、返事をくれたのはそのうちたった1社だったそうです。また、半導体部品などのパーツの入手にも苦労したそうです。

ともあれ、2週間ほどの間で製品コンセプトはだいたい固まりました。「IPアドレス共有のための、小型・低消費電力型イーサネットルータ」で、「PC/AT仕様、i486ベースの産

業組み込み用ワンボードマイコンを利用する」というものになりました。

他者の支援を得て実質2カ月で開発

その後、コストとサイズを抑えるための若干の構成の変更を経て、'99年8月から本格的に「子羊ルータ」の開発が始まりました。といっても、本来の業務であるソフトウェア開発を行いながらの開発です。「子羊ルータ」のために投入できる労力はそれほど大きくありません。

ハードウェアは森氏が担当しましたが、自前で行ったのはプロトタイプの構築、回路設計、部品の選定と調達、筐体の設計、試験などです。実際に製品の形となるにあたっては、基板の設計や部品の実装、筐体の製造などが必要になりますが、これらはそれぞれ、CAD・金型などのプロフェッショナルに委託しました。

ソフトウェアは柳沼氏が担当しました。こちらは主に動作中の電源断への対応と、ストレージとして採用を決めた16Mバイトフラッシュメモリに収まるサイズのシステム構築です。ここではLinuxディストリビューターであるメディアラボの支援が得られ、またJFやLinux Users MLも活用したということです。このあたりの開発スタイルは、リーナス・トーバルズ氏が緩やかな統括と方向付けをするものの、多数のプログラマーが自分のできる仕事をそれぞれのペースで行うことによって開発されるLinuxの開発スタイルとの類似を感じるところです。

「子羊ルータ」の試作機は'99年9月末に仕上がり、10月には予約受け付けを開始、10月末に出荷を開始しました。その後、シリアルポートの追加やカーネルのバージョンアップなどを経て、現在に至っています。

小さいという強みを活かす

筆者がワイルドラボを取材していて感じたのは、従業員二名という「小ささ」をフル活用しているという点です。ワイルドラボの場合、人員的にも資金的にも限られた中で自分たちに適したビジネスのあり方を模索してゆき、結果として、いや必然的に、それが製品の適正規模でのヒットにつながっています。

たとえばワイルドラボは、「子羊ルータ」を大きく宣伝してはいません。本格的な宣伝・広告には多額の費用が発生するからという理由もありますが、ワイルドラボでは「子羊ルータ」へのMACアドレスの焼き付け、梱包・出荷、サポートを自分たちで行っているため、製品をそれほど多数出荷することが不可能だからです。このことは結果的に、良心的な製品とサポートを、求める顧客に確実に提供することにつながっています。

変化するマーケットへの対応

マーケットは時々刻々と変化します。'99年当時、ケーブルテレビ会社のインターネットサービスを利用する個人やSOHO向けにルータとして発売された「子羊ルータ」は、機能・価格とも画期的な製品でしたが、現在ではいくつもの競合製品が出現しています。'99年の発売当初と比較すると、ルータとしての商品価値はかなり減少していることは否定できません。

ワイルドラボでは、小型のPCとして「子羊ルータ」で遊ぶLinux/BSDフリークをメインの顧客層として、この7月11日、これまでi486ベースだったCPUをAMD Elan520（x86互換CPU。AMD Am5x86-133MHz相当）に変更したモデルを発売しました。変化するマーケットへの柔軟な対応は、従業員が2人だからこそやれることと言ってもよいかもしれません。

あえて拡大を目指さない方向性

柳沼氏は「豊かになることは拒まないけれども、そのせいで楽しくない仕事をすることは避けたい」と言っています。また、「会社を大きくしたいという欲はあまりない」とも言っています。これらは一般的なベンチャー企業の方向性とはかなり異なっています。

しかしながら、人の性格がそれぞれであるのと同様、企業の性格もそれぞれです。結局はそれぞれの企業が自分たちに適した方向性や方法を見つけ、それでビジネスを継続させてゆくことが、さまざまな形での「成功」ではないかと筆者は思います。

柳沼氏と森氏は現在、「これから、何をやろうかな」と考えています。現在はソフトウェア開発の傍ら、「子羊ルータ」の改良をやっていますが、「子羊ルータ」を最初に開発したときに「こういう製品が欲しい」と考え、開発・販売にこぎつけるまでに湧き上がったエネルギーは大変なものだったそうです。現在は、その時のように熱くなれるものを探しているところだということです。

今後、低成長、もしかすればマイナス成長となりうる日本経済の中で、個人や企業が生き延びるための正解は案外、「自分が無理なくやれる」「自分が熱くなれる」方法を探すことかもしれません。

自分のスタイルは自分で作るしかない

ワイルドラボを取材して、筆者は自分自身のことと引き比べずにいられませんでした。筆者は以前、企業に勤務して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、いろいろな事情からその仕事の継続が難しくなりました。仕事自体は自分の「天職」と呼びたいようなものだと感じていた筆者にとって、これは非常な痛手でした。同時に、「企業や大学など圧倒的に男性が多い組織の中では、女性のキャリア形成が難しい」ということも自分の経験から痛感していました。また、「フリーランスの研究者という職業を成立させてみたい」という希望を持つようになり、勤務先を退職し、著述活動を広い意味での「研究」ととらえて現在に至っています。

筆者の場合、まず「フリーランス」というライフスタイルに慣れるまでに苦労しました。企業で他人・他部署にマネジメントをしてもらうということに慣れてしまっていたので、すべてを自分でやるというライフスタイルに慣れるまでには少し時間がかかりました。また、一人でやっていることの強みに気付くまでにも相当の時間がかかりました。ともすれば、「自分一人だけ」「不安定」といったネガティブな要因ばかりに目を取られてしまい、「自分ですべてを決められる」「仕事の量やライフスタイルを自分でコントロールできる」というメリットになかなか気付かなかったのです。

会社員といえども身分や生活が安定しているとはいえなくなってきた現在、自分たちのスタイルは自分が開発するというスタンスは、誰にとっても必要なものではないでしょうか。かつてLinuxとともにLinuxの開発スタイル「バザール形式」が開発されてきたように。

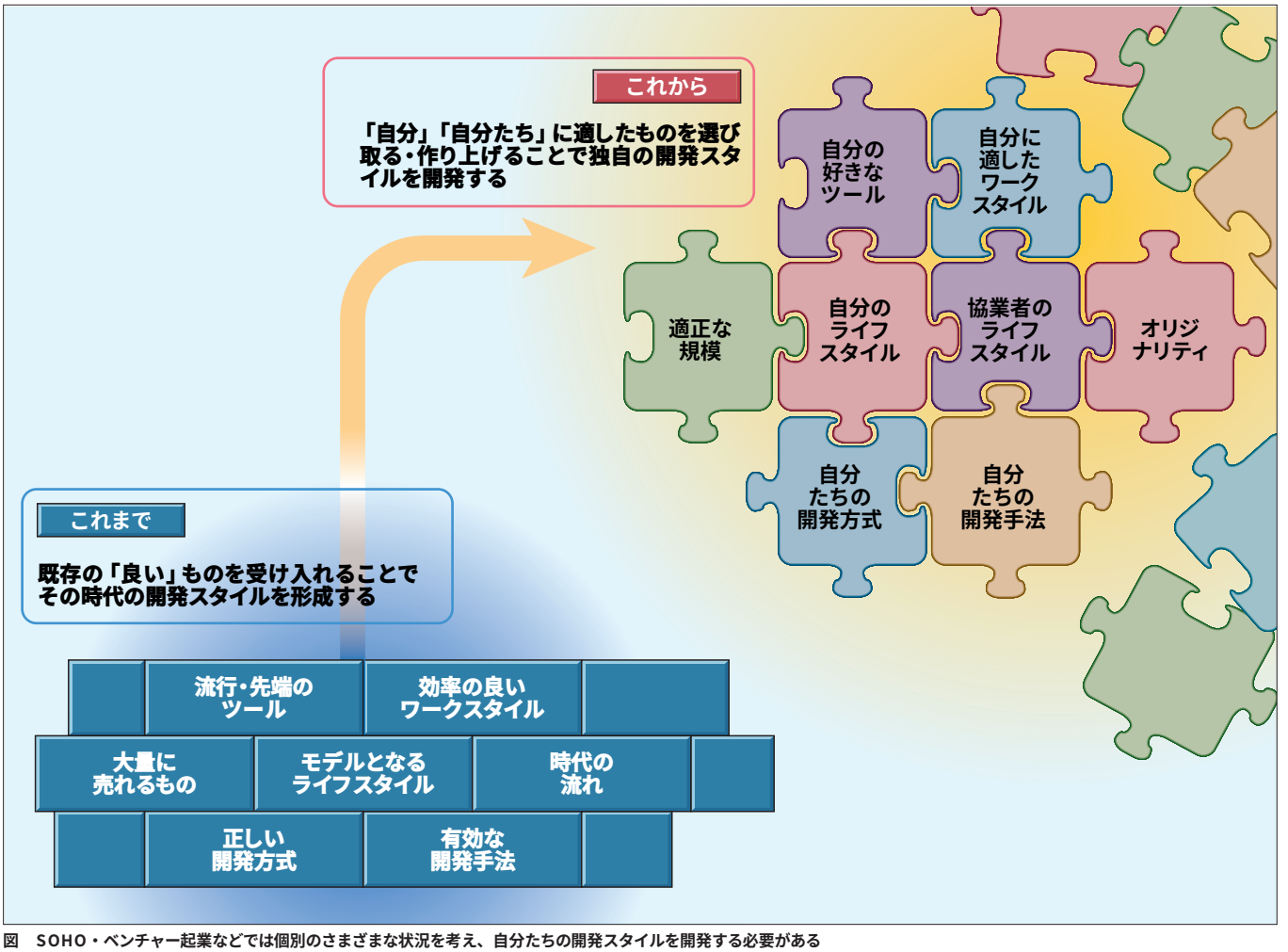


図　SOHO・ベンチャー起業などでは個別のさまざまな状況を考え、自分たちの開発スタイルを開発する必要がある

「子羊ルータ」生みの親。ワイルドラボ代表取締役柳沼由貴子氏、取締役森源明氏に聞く

Interview

マイペースで無理のない開発・経営スタイルでSOHOビジネスを継続しているワイルドラボ。今回は全取締役かつ全社員であるお2人に、「子羊ルータ」を中心に語っていただきました。

――はじめまして。情報機器の名前として「子羊ルータ」というのは面白いですね。どこから命名されたんですか？

森：もともと、この人（柳沼さん）が本当に羊が好きで。

柳沼：ええ、柳沼だから「ヤギ」なんですけど、メールアカウントやハンドルなども昔から「lamb」にしていたんです。

森：製品の大きさを見て、イメージができたんだよね。

柳沼：そうそう、子羊色にしなくっちゃ、って。ピンクとオフホワイトと2つアイデアがあったんだけど、実際に塗装してみるとオフホワイトの子羊色のほうがよくて。

――ピンクもあったら可愛いと思いますけれども、このオフホワイトだと、オフィスにあっても家庭にあっても違和感がなくていいですね。キャラクタデザインもご自分たちでされたんですか？

写真　柳沼氏（左）と森氏（右）

柳沼由貴子氏（やぎぬまゆきこ）

生物学者になりたかったが挫折、ソフトウェア会社に拾われプログラマーとなる。ほどなく自費出版がDTPでできることを知り、Macintoshを手に入れる。アップルコンピュータの「ヒューマンインターフェイスガイドライン」に感動し、世の中GUIだけでなく、人間関係や会社経営もこれで行こうじゃないかと心に誓う。最近の楽しみは、近所の無愛想なシベリアンハスキー犬に尻尾を振らせること。

森源明氏（もりげんめい）

コンピュータとの付き合いは、中学一年のときに電話級アマ無線の合格を条件に、父に当時30万円近くした8ビットマイコンをねだったのがはじまり。以来、はまったのはSONYのCP/Mマイコン、EPSONのハンドヘルド、白黒MacintoshとPowerBook。現在は、「ペン入力ができるモバイルPCはもう出ないのかしら」と嘆きつつ、BIBLO/MCをしつこく愛用中のWindowsユーザー。ノートといえば、実家の3人の無線LANノート使いたち（母、姉、妹）も、もちろん子羊を愛用中。

森：いえ、知り合いのデザイナーに「こんなキャラクターで売り込みたい」という話をして作ってもらいました。自分たちの販売していくスタンスを伝えて、「可愛くて賢そうできりっとしてて優しそうな羊」とか「飛んでいるところがいい」とかいろいろ注文して、最終的に仕上がったのが現在の子羊のキャラクタです。

――楽しんで作っていらっしゃる感じがしますね。

柳沼：そうですね。もともとワイルドラボの本来の業務は業務ソフトの受託開発やコンサルティングなどでしたので、このルータは息抜き的、趣味的に開発を始めたんです。

――起業されたのは'98年6月ということですが、それ以前はどういうお仕事をされていたんですか？

森：2人ともある商社にいたんですけれども、その商社で彼女のほうが関東方面でC/Sシステム構築ツールのセミナーの講師を、私のほうはそのツールの技術営業のようなことをやっていました。

――その時に技術を蓄積されたんでしょうか？

柳沼：今につながるものはあまりありませんね。けれども、セミナーの講師をやることでお客様と直接接触することができて、エンドユーザーと接触することの面白さに目覚めましたね。それは今につながっているように思えます。その商社の前はソフトハウスに長年いまして、UNIX上での開発をしていたんですけれども、'99年に「子羊ルータ」を開発した時には5年ほどのブランクがありました。

森：僕は「子羊ルータ」ではハードウェアを担当しているんですけれども、現在30代前後の方って、世代的にハンダゴテに抵抗がないじゃないですか。

――私もその世代なので、感覚的にわかります。

森：学生のころから電子工作をやったりしていましたし、電子工学科を卒業しているのでハードウェア設計には抵抗がなかったです。

柳沼：私にとってはハードウェアは別世界なんです。私はメモリやハードディスクを挿すくらいしかやってなくて。

――お2人とも、ソフトの仕事に就かれたのは最初に就職したときからですか？

柳沼：私は大学は生物学科だったんですけど、コンピュータがそのころ花形だったのでソフトハウスに就職して、Macintosh、PC-9801、Windows、EWS（エンジニアリング・ワークステーション）とひと通り見てきました。

森：僕はメインフレームのSEをやったり、Macintoshの販売店で営業をやったりといろいろやってきましたね。

――起業されたきっかけは何だったんでしょう？

柳沼：商社であるツールを扱っていたんですけど、2人ともそれに惚れ込んだんです。コンセプトといい、機能といい。

――どういう製品だったんでしょう？

森：ご存じないでしょうね。クライアントでアプリケーションを開発して、データベースサーバと接続するというツールなんです。

柳沼：今だったらブラウザを使うところですけど。これの凄いところは、クロスプラットフォームで、Macintoshで作ってもWindowsで作っても同じように動くということでした。

――そのまま会社で続けるという選択肢はなかったんでしょうか？

柳沼：時代がブラウザ、アプリケーション、データベースの3階層になってしまったので、だんだんそのツールの需要がなくなってきたんです。

――そういう時期の起業には勇気が必要だと思うのですが。

森：すでにそのツールが入っているところには、開発の人間は必要でしたし……。

柳沼：C/Sシステムのアプリケーションに関しては広く扱えるという自信があったので、仕事さえあれば大丈夫という感じでしたね。

――それで'98年に会社を設立されて、その後の展開はどうでしたか？

森：この人（柳沼さん）が非常に求心力があって、以前いた会社の人たちとのおつきあいがずっと続くようなタイプなので、仕事が集まってくるという感じでしたね。それは非常に助かりました。知り合いの人が仕事を分けてくださった感じなんで、やりやすかったですね。営業するには、それなりに実績を見せられないと難しいと思いますけど。

柳沼：そういえば営業ってしたことないね。

森：ない（笑）。

――20代の若い方が起業する場合とはまったく違う感じを受けますね。若い方だとまず営業が必要で、モノを開発しても売るチャネルがないのでリセラーを通さなくてはならなかったりしますし。

森：「子羊ルータ」のような少量生産だと、人間関係といいますか、ほかの方に助けていただかないとそもそも成立しないんですよ。

柳沼：ホームページにも書いていますけれども、皆さん根気よく付き合ってくださった方ばかりで……。

森：そうそう。こういったプロダクトを作るのは初めてだったので、業界用語がわからなくて行き違いがあったりして……。

――そういえば、ホームページに「子羊ルータ」協力者の全員のお名前を書いていらっしゃいますね。

森：隠す必要もありませんし、「あそこで見たよ、と仕事の依頼が来たよ」なんて感謝されることもあるんです（笑）。

柳沼：「子羊ルータ」には特許的なところがありませんし。協力者の方々には感謝の気持ちがすごくあるんです。3カ月足らずできっちりしたモノができたのは、その方々のおかげとしか言いようがありませんし……。

森：素人作品ではない、完成度を誇れるプロダクトになったのは、本当に周囲の人のおかげでしたね。

――豊かな人間関係をフルに活かしていらっしゃるという感じですね。ところでお2人の協力関係はどういう感じなんでしょうか？

森：「子羊ルータ」はハードウェアが僕でソフトウェアはこの人という分担なんですけれども、性格的にまったく違うので2人でバランスしているという感じがあります。

――どういうことでしょう？

森：僕は「貧乏してても技術的に面白ければいいじゃないか」と思うし、新しい技術はどんどん入れたいわけです。イケイケなんです。

柳沼：私は、事業として成立させなくては……と思いますし、新しいものを入れたために何かトラブルがあったらどうしよう……と考えてしまいますね。慎重派なんです。

――なるほど、見事なバランスですね。ところで、ホームページを拝見していますと、BBSを作ってこまめにご自分たちでサポートされていて、ユーザーサポートを楽しまれているように見えますが。

柳沼：そうですね、エンドユーザーの顔が見えない商売はしたくないんです。

森：ユーザーサポートもそうだけど、もともと「自分たちがやりたいからやっている」って感じですね。

柳沼：気楽にやれるように在庫も持たないようにして、できるだけ手作りで。

――だからこそ、「子羊ルータ」という名前も出てくるんですね。

森：そうですね。これが「IPルータ1」とかって名前だったら、そんなに売れなかったんじゃないかな（笑）。

――それにこの外形、単純な直方体がいいですね。どこにでも安定して置けますし、上に物を置くこともできますし。

森：開発当時、iMacはもう出てたかなぁ……？　最初はああいったトランスルーセントで曲線的なデザインも考えたんですよ。

柳沼：結局は金属筐体にしたんですけど、ある程度の重量があるおかげでケーブルに引きずられることがなくなるというメリットがありました。

――金属筐体でもこのサイズだと可愛いものですね。それにこの色がなんともいえません。

柳沼：そうですね。「子羊ルータ」は、キャラクタ、ネーミング、デザインがうまく噛みあった感じです。

森：お客さまも、「子羊さんはよく働いてくれます」とか、「子羊くんの調子が悪いんです」とか擬人化してくださいますしね。

柳沼：子供のようにして作ったものが、客先で可愛がられているようすがとても嬉しいです。ここまで受け入れられたのはお客さまやメディアの方のおかげで、心からお礼を言いたいです。

――確かに、話題の製品であるわりに、広告はあまりされていらっしゃいませんね。

柳沼：ええ。取材していただいたり、あとはお客さまの口コミで広がっていきました。BBSでも活発にやりとりしてくださって。

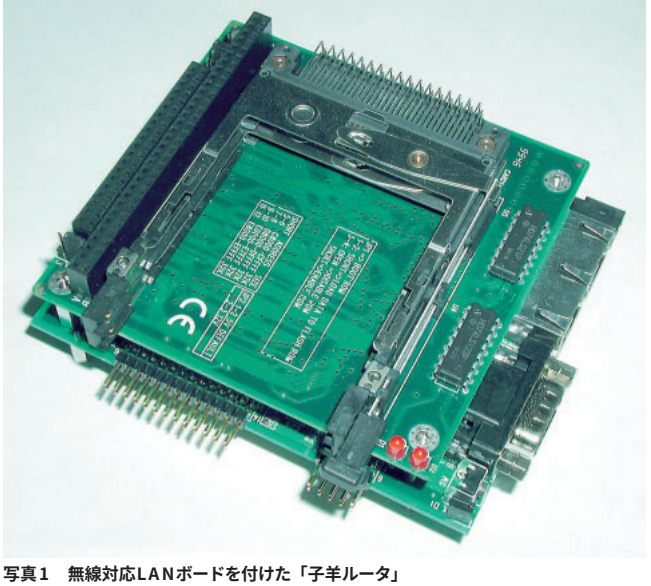


写真1　無線対応LANボードを付けた「子羊ルータ」

――それはやはり、製品自体の魅力があってこそのことでしょう。

森：そうですね。「お客さまにとって魅力的なプロダクトに」という努力はしています。たとえば、最初のモデルはコンソールがなかったんですが……。

――ルータとしての利用にはまったく問題ないですね？

森：ええ。ですけど出荷してみますと、Linuxフリークの方がいろいろと遊んでくださって、「やっぱりコンソールがないと辛い」とおっしゃるわけです。それで最初のマイナーチェンジでシリアルポートを付けました。

柳沼：最初から「遊べる」要素は考えていましたね。初期のモデルには16Mバイトのコンパクトフラッシュメモリを搭載しているんですけれど、Linuxで気軽に遊べるように、コマンドは程よく残しつつサイズを小さくするという感じで、ガンガン小さくはしなかったんです。

――その方向性がやはり現在につながっているんですね。

森：そうですね。ルータというよりは小さなLinuxボックスとして扱っていただいているといいますか。あとは制御系ですね。単なるメディアコンバータではなくて、それなりの処理をしてから流せるというところが魅力になるようです。最近も無線LAN対応の話がある企業から来まして、こんなふうに無線LAN対応ボードを上に重ねるんです（**写真1**）。

――応用の可能性はまだまだ広がりそうですね。

柳沼：ええ。7月11日に5x86ベースの新しいモデルを出します。これはもう、ルータとしての利用はメインではなくて、LinuxやBSDフリークの方に遊んでいただくためのものとして考えています。

――最後に、今後の展開について聞かせてください。

森：今、これから何をやろうかな……と考えているところなんです。

柳沼：「子羊ルータ」を作った時には、ケーブルテレビのインターネットサービスが開通してルータが欲しくなって、その時に湧き上がるエネルギーが凄かったんです。

森：開発してたあの3カ月、ホントに凄かったね（笑）。

柳沼：そうそう、ひとつひとつ「動いた！」って喜んでね。今は「子羊ルータ」の改良をやりながら、同じようにエネルギーが湧き上がってくる何かを探しているところです。

――その「次の何か」が楽しみですね。また、「子羊ルータ」が今後、LinuxボックスとしてLinuxフリークやBSDフリークとともに発展していく様も非常に楽しみにしています。今日はありがとうございました。

第4回組み込みシステム展レポート

第4回組み込みシステム展（ESEC Embedded Systems Expo & Conference in Tokyo 2001）レポート

2001年6月26日～29日、東京ビッグサイトで「第4回組み込みシステム展（ESEC）」が開催された（主催：リードエグジビションジャパン（株）、協賛：（社）トロン協会）。組み込みシステムを追っている筆者には見逃せない展示会である。しかも今年から「組み込みLinuxゾーン」が開設されるとのこと。なおさら見逃せない。執筆スケジュールは一杯で、あちらこちらから原稿を催促する声が聞こえている状況ではあったけれども、その合間をむりやり縫って行ってきた。

まだ6月であることが信じられないほど暑い快晴の午後、会場に到着してまず目についたのは、スーツ・ネクタイ姿、しかも、いかにもエンジニアという独特の雰囲気を漂わせた大量の男性と、かっちり決まった服装の少数の女性の姿。かつてエンジニアであった筆者は、「お、懐かしい匂い」と感じてしまった。もっとも、筆者のかつての職場は男女ともユニホーム着用だったので、男性のスーツ姿にはめったに出くわさなかったのだが……。

実はESECと並行して、「ソフトウェア開発環境展」「データウェアハウス&CRM EXPO」「データストレージEXPO」の3つの展示会が、同じ東京ビッグサイトで開催されていた。このため、ソフト・ハードに関係する大量のエンジニアが来訪していたのだった。3日間の来訪者は、4つの展示会で延べ5万3558人ということだ。どの会場も冷房が効いていたが、ソリューションを求める大量のエンジニアの熱気でいっぱいだった。

写真1　今年から新設された「組み込みLinuxゾーン」
組み込みシステムへのLinux利用への注目が高まっている現状を反映してか、熱気を帯びた人の波が途切れることなく続いていた。

注目を集める組み込みシステム

ESEC会場の中は人、人、人、しかも、そのほとんどが男性。筆者は自分よりも背の高い男性に周囲を囲まれ、自分が会場のどこにいるのかを確認するのにかなり苦労した。

ESEC事務局によると、今年の来場者は昨年に比べて31%も増加しているそうだ。ちなみにESECの来訪者は、3日間

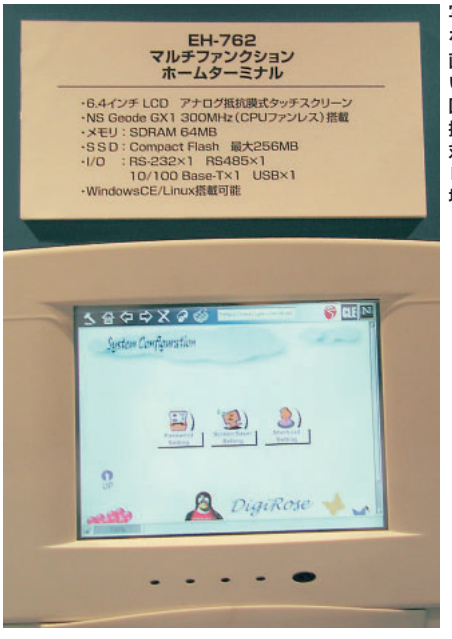


写真2　Linuxを搭載したホームターミナル
画面にペンギンが映っている。これに限らず、今回のESECでは「Linuxを採用しました」「Linuxに対応しました」というフレーズやペンギンの姿を会場のあちこちで見かけた。

で延べ1万6315人で、ソフトウェア・データベースシステムに関連する他の3つの展示会のどれよりも多い。

最近、携帯電話でJavaアプリケーションを動かすなど、地味な存在であった組み込みシステムが一般情報機器ユーザーから見ることのできる、身近な存在になりつつある。ESECの入場者数の増加ぶりは、社会的に組み込みシステムへの注目が集まっていることの反映でもあるだろう。

今年から新設された3つのゾーン

ESECには今年から、「組み込みJavaゾーン」「組み込みLinuxゾーン」「組み込みFPGA/PLDゾーン」の3つが新設された。この背景に、JavaとLinuxが組み込み用途で注目を集めていることがあるのは、説明の必要もないだろう。

ちなみにFPGAとは、Field Programmable Gate Arrayのことで、直訳すると「現場で書き換え可能なゲートアレイ」になる。PLDとはProgrammable Logic Deviceのことで、直訳すると「プログラム可能なロジックデバイス」となる。これらの長所を利用すると、LSIを短期間で開発し、しかも少量でも大きなコストをかけずに生産できるというメリットがある。組み込みシステムではLSIのレベルから、短期間で個別対応した開発を要求される場面が以前より増大しているという背景があり、FPGAやPLDが注目を集めているようだ。

ツールベンダーの出展が目立った「組み込みLinuxゾーン」

「組み込みLinuxゾーン」では、組み込みLinux開発ツールベンダーの出展が目立った。最近、組み込み用途にも注力しているレッドハット、そして世界的に事業展開するモンタビスタ、リネオ、タクシア、国内勢ではA＆Aリナックス、日進ソフトウェアといったところである。

レッドハットがITRONに対応したリアルタイムOS「eCos」を出展し、モンタビスタの「Hard Hat Linux」が「Javaにも対応できる」というメリットをアピールしているのを見て、「Linuxは組み込み用途で利用できる数多くのOSのひとつに過ぎないんだなぁ」という当たり前の事実を改めて感じてしまった。

「Linuxチップ」は不可能だろうか

「組み込みJavaゾーン」では、富士通の「Javaチップ」が気になった。「Javaバーチャルマシンがバーチャルでなくなったもの」といえば最も近いだろうか。Javaアプリケーションをバイナリのまま実行できるプロセッサである。

JavaとLinuxではアプリケーションの実行の仕組みがかなり異なるが、たとえばプロセッサをLinuxカーネルやライブラリと一体化してしまったら、いろいろと面白い応用が考えられるだろうなぁ……と筆者はその場でしばらく妄想を繰り広げた。

名刺より小さなネットワーク対応ボード「Network Engine」

組み込みシステム展で最大のブース数を誇っているのは、ボードコンピュータである。大小さまざま、いろいろな仕様、

写真3　Network Engine
名刺より小さいサイズにサーバ・ルータに必要な機能を組み込んだボード。採用しているOSはNetBSD。Linuxも搭載できるそうだ。

写真4　日立のブースの一部
日立のプロセッサに対応する開発ツール群が「Linux」というカテゴリーにまとめられている。

さまざまな用途（実にいろいろな用途の専用ボードが存在するのである）のボードを見て回りながら「このボードを使ったら何が作れるかなぁ……」とあらぬことを考えながら歩いていた筆者は、小さなガラスケースの中に小ぢんまりと展示されている、名刺よりも小さなボードに目をとめた。一瞬、「もしかして、実はケースのガラスが凹レンズで、ボードが縮小されて見えているのかな？」と思ったくらいであるが、その大きさが実寸法だった。レーザーファイブの「L-Card+」も小さいが、それよりさらに小さい。

それはアイピー・スクエアの「Network Engine」という製品で、名刺より小さなサイズ（67.6mm×50.8mm）にSH3プロセッサ、SDRAM（32Mバイト）、フラッシュROM（4Mバイト）、イーサネットポート×2（100BASE、10BASE各1つ）を搭載したものである。「ネットワーク機器向け組み込みエンジン」と銘打たれており、ほとんど単体でネットワーク機器をそのまま開発できてしまうというボードだ。フラッシュROMの中にはNetBSD 1.5とアプリケーションやコマンド類がインストールされている。もちろんLinuxを搭載することも可能ではあるとのことだ。機会があれば、ぜひ入手して個人的に遊んでみたいボードである。

ハードディスクの騒音・振動問題を解決？ ハギワラシスコムの「FlashDrive」

各種メモリカードやその周辺機器で知られるハギワラシスコムのブースには、「FlashDrive」という製品が展示されていた。「ハードディスクのディスクの代わりにフラッシュメモリが使用されている」といえばイメージを掴んでいただけるだろうか。回転するパーツがないので回転音も振動もない。ハードディスクのように精密でデリケートな機構になっているわけでもないので、衝撃にも強い。もちろんIDE・SCSIの各インターフェイスに対応している。パソコンのハードディスクをこの「FlashDrive」に置き換えれば、ハードディスクの騒音・振動問題は一挙に解決だ。魅力的な存在である。

ただし現時点では、シビアな環境で利用されるFA用途が主なターゲットとなっているようだ。フラッシュメモリの価格や生産コストを考えると、このような製品が個人ユーザーにとって利用可能な範囲の価格で販売される時代が近い将来にやってくるとは考えにくい。

ボードやプロセッサの「Linux対応」

このほか、ボードやプロセッサのブースで「Linux対応」が謳われているのが目についた。

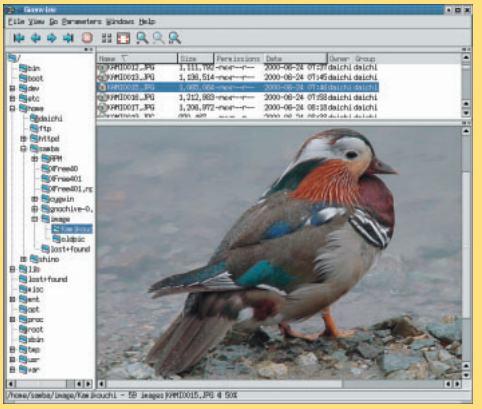
Linuxは標準的なアーキテクチャやプロセッサの多くに対応しているので、わざわざ「Linux対応」とアピールすることはないだろう……と筆者は一瞬思ったが、「Linux対応」であることは現在、ボードやプロセッサのメーカーにとってアピールする価値のあることであり、製品の価値を高める要素と考えられているようだ。数多くのOSが存在する組み込みシステムの世界の中で、Linuxが大きな存在となりつつある時代の流れを感じた。

写真5　組み込みソフトウェアベンダー「エーアイコーポレーション」のブース
LinuxがµITRONと同列に並べられている。ここが扱っている「A&Aリナックス」は国産であることを強調し、ペンギンが武士の姿をしている。

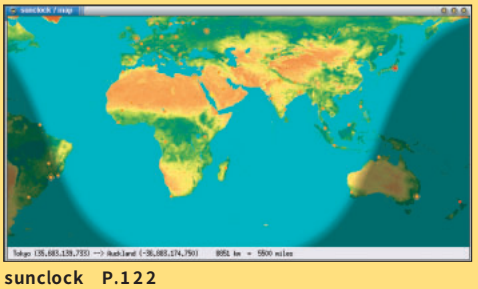
写真6　組み込みソフトウェアベンダー「Lineo」のブース
組み込み向けLinux製品「µCLinux」を提供している同社の組み込みJava向けソリューションの展示の片隅にも、なぜかペンギンが……。

Free Application Showcase

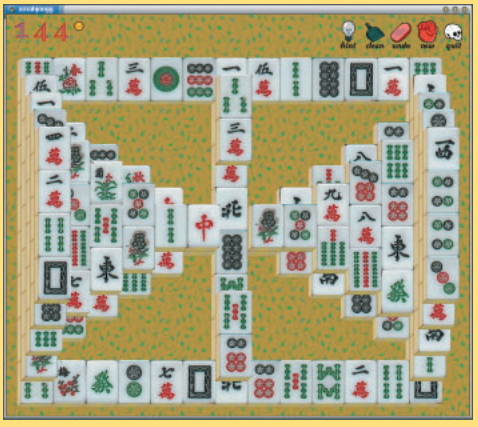
文：出井 一
Text : Hajime Dei



Gwenview　P.120



sunclock　P.122



Xmahjongg　P.130

※紹介したソフトは、すべて付録CD-ROMに収録されています。

科学技術系2D／3Dグラフを描画

SciGraphica

バージョン：0.7.0　　ライセンス：GPL

http://scigraphica.sourceforge.net/
http://gtkextra.sourceforge.net/（GtkExtra）

SciGraphicaは、散布図などの科学技術系グラフを描画するソフトだ。「ワークシート」でのデータの入力から「プロット」でのグラフの描画に関するパラメータ設定まで、あらゆる作業をグラフィカルな環境で行えるのが特徴だ。スケールの異なる複数のグラフを、レイヤーを使って重ねて表示することもできる。動作には同じ作者のGtk+Extraのほか、Numerical Pythonなどのライブラリが必要だ。

●ビルドとインストール

(1) Gtk+Exreaのインストール

Gtk+Extraは、グリッドやプロット、ディレクトリツリーなどのGTK+用ウィンドウ部品（ウィジェットセット）を提供するライブラリで、tarボールのみ配布されている。

tarボールをそのまま利用する場合は、「./configure」→「make」としてビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。

tarボールからRPMパッケージを作成することもできる。「rpm -tb gtk+extra-0.99.15.tar.gz」とすると、/usr/src/redhat/RPMS/i386にバイナリパッケージが作成されるので、「rpm -Uvh gtk+extra-＊0.99.15-1.i386.rpm」としてインストールする。

(2) Numerical Pythonのインストール

Numerical Pythonは、スクリプト言語Pythonに高速な多次元配列を提供するライブラリだ。最新版は20.1だが、国産ディストリビューションで使われているPython 1.5.2では、より古い17.3を利用する必要がある。また、インストールにはDistutilsライブラリが別途必要だ。

どちらもtarボールのみ配布されているので、tarボール展開先のディレクトリで、「python setup.py install」としてインストールする。

(3) SciGraphicaのインストール

SciGraphicaは、tarボールのみ配布されている。「./configure」→「make」としてビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。

●ワークシートに数値を設定

ktermなどのコマンドラインで「sga&」として起動すると、SciGraphicaのメインウィンドウが開く（**画面1**）。上部のタブを使って、データ入力用の「ワークシート」のアイコンが並ぶ[Worksheets]ページと、グラフ描画用の「プロット」のアイコンが並ぶ[Plots]ページを切り替える。

最初は、「Data1」（名前は変更可）というワークシートがひとつだけ用意されている。ツールバーの[Sheet]ボタンを押して新たなワークシートを追加し、複数のワークシートを利用することも可能だ。

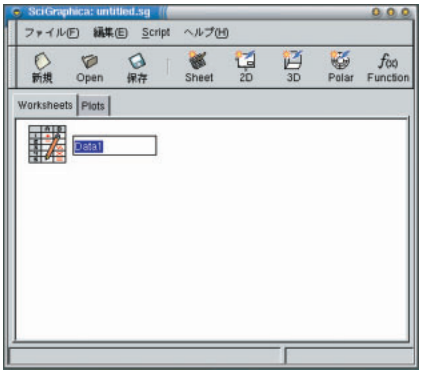
こうしたワークシートのアイコンをダブルクリックすると、内容が別ウィンドウに表示される（**画面2**）。表計算ソフト風のセルが縦横に並んだグリッドに、グラフの表示に利用する数値データを入力する。初期サイズは5列×20行だが、ツールバーのボタンや右クリックメニューにより、行／列の追加や挿入、削除などが可能だ。

●データをファイルから読み込む

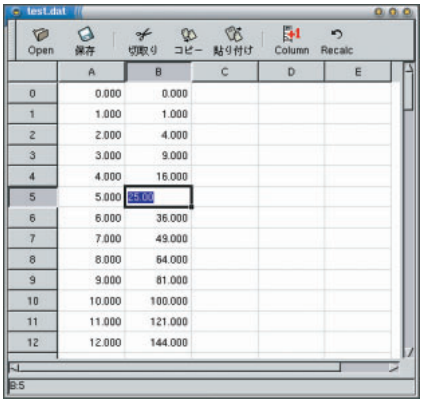
このほか、外部のテキストファイルや別のワークシートからデータを読み込むこともできる。

テキストから読み込む場合は、右クリックメニューの[File]－[ASCII Options]で設定ダイアログを開き、データを区切る文字（デリミタ）や、読み飛ばす先頭の行数などを設定しておこう（**画面3**）。一方、別のワークシートで[保存]ボタンを押してファイルに保存したデータを読み込む場合は、専用のXML形式（拡張子sgw）になっているため、こうした設定は必要ない。

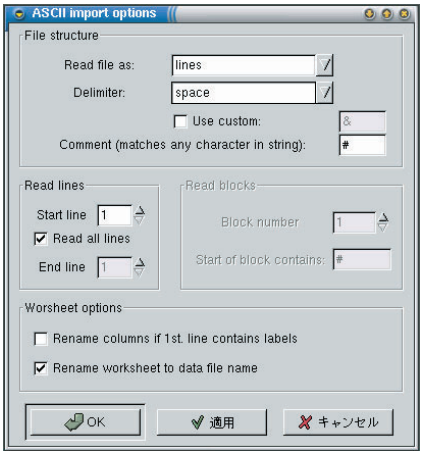
読み込みを行うには、ツールバーの



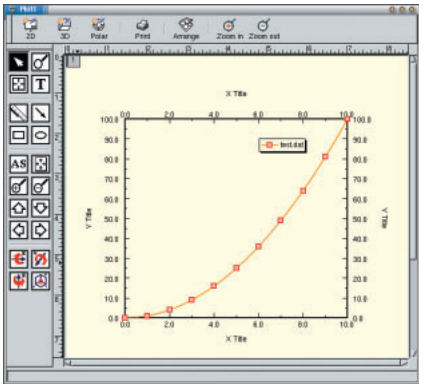
画面1　ワークシートとプロットにページ分けされたメインウィンドウ。



画面2　データの入力は、表計算ソフト風のワークシートで行う。



画面3　テキストファイルから読み込む場合のデリミタなどを設定する。



画面4　プロットウィンドウには、グラフと操作用アイコンが表示される。

では、左側のツリーに並んだ項目をクリックすることで、さまざまな設定のページを切り替えられる。

たとえば、ツリートップの[Plot1]では紙のサイズや向き、その下の[Plot1:1]ではグラフの背景色などを変更できる。なお、「Plot1」はプロット名の初期値で、ユーザーが自由に変更可能だ。以下、[Frame]ではグラフの周囲の枠、[Grids]ではグリッド、[Legends]では凡例、[Axes]では各軸に関する設定をそれぞれ行える。

なお、こうした設定の中には、プロットウィンドウで直接変更できるものも少なくない。たとえば、グラフをクリックして、四隅と四辺に表示される■をドラッグするとグラフの大きさを変更できる。また、各軸の表示範囲やスケールは、ウィンドウ左に並ぶボタンで変更できる。

[Open]ボタンを押し、オープンダイアログでファイルを選択すればいい。テキストファイルを読み込む場合は、[File Format]の設定を「ASCII」に、ワークシートを保存したファイルから読み込む場合は「SciGraphica Worksheet XML」に設定する。

●プロットの作成

データの入力が終了したら、メインウィンドウを[Plot]ページに切り替える。ワークシートと違って、最初はプロットがまったく用意されていないので、以下の手順で追加しよう。

プロットには、直交座標系の「2D」と「3D」、極座標系の「Polar」の三種類があり、それぞれのボタンがツールバーに用意されている。これらのボタンのいずれかを押すと、「Plot1」（名前は変更可）という名前のプロットのが[Plots]ページに追加される。ワークシートと同様に、複数のプロットを扱うことも可能だ。

こうしたプロットのアイコンをダブルクリックすると、内容が別ウィンドウとして開く（**画面4**）。ウィンドウの左には操作用ボタン、右側にグラフが表示される。なお、プロットウィンドウの大きさを変更しても、グラフの大きさは追従しない。[フィット]ボタン（右列の上から5番目）をクリックする必要がある。

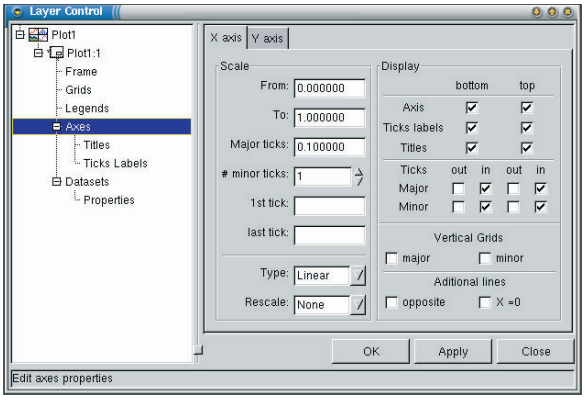
●レイヤーダイアログで設定変更

プロット上のグラフに関する設定は、グラフの左上にある[1]ボタンのダブルクリックで開く「レイヤーコントロールダイアログ」で行う（**画面5**）。ここでは、左側のツリーに並んだ項目をクリックすることで、さまざまな設定のページを切り替えられる。

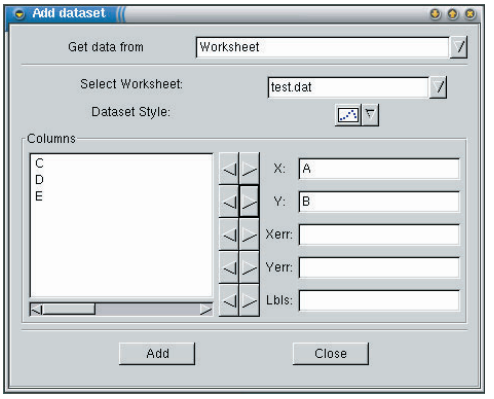
●データセットとは

ワークシートのデータをプロット上のグラフに表示するには、ワークシートのどの列（カラム）をグラフのX軸やY軸の値として利用するかを登録する必要がある。この組み合わせを「データセット」と呼ぶ。

基本的に、ワークシートとデータセットは1対1で対応するが、ひとつのワークシートから複数のデータセットを登録することも可能だ。もちろん、複



画面5
データセットの登録などを行うレイヤーダイアログ。



画面6
データセットでは、ワークシートの列とX,Y軸を関連付ける。

数のワークシートが存在する場合は、ワークシートごとに個別のデータセットを登録できる。

さらに、今回詳しく触れることはできないが、数式からデータを生成する関数（function）や、Pythonスクリプトの変数値・計算値などもデータセットとして利用可能だ。

レイヤーコントロールダイアログの[Datasets]ページの中央上にある[Add]ボタンを押すと、データセットの追加ダイアログが開く（**画面6**）。まずは、上部の[Get data from]で、データセットの種類（ワークシート・関数・Pythonの変数値）を選択しよう。

●ワークシートからの登録

ワークシートからデータセットを登録する場合、[Get data from]で「Worksheet」を選択した後、[Select Worksheet]のリストから、利用するワークシート名を選択する。

続いて、[Dataset Style]の下向き三角ボタンを押し、散布図・折れ線・棒といったグラフのスタイルを選択する。線の種類や色使いなどの細かな設定はあとで行う。

最後に、[Columns]に一覧表示された列の中から、X軸やY軸の値として利用するものを選択し、[X:]や[Y:]の横のボタンで関連付ける。

[Add]ボタンを押してデータセットを登録すると、レイヤーコントロールダイアログにデータセットの一覧が表示される。[Apply]ボタンを押して、プロット上にグラフが描画されることを確認しよう。もし、X軸やY軸の範囲が不適切な場合は、グラフの左にある[AS]（オートスケール）ボタンで自動調整するといい。

また、レイヤーダイアログのツリーの[Properties]をクリックすると、データセットごとのグラフ属性設定ページに切り替わる。

ここでは、データ間の線の描画方法やデータ点の記号、ラベルのフォント、各部の色などを柔軟に設定可能だ（**画面7**）。

●複数のレイヤーを使用する

プロット上に複数のグラフを並べたり、スケールの異なるグラフを重ねて描画するには、複数のレイヤーを利用する必要がある（**画面8**）。

プロットのツールバーにある[2D][3D][Polar]ボタンのいずれかをクリックすると、対応する種類のグラフが新たなレイヤーとして追加される。グラフ左上のレイヤーボタン（[1][2]…）をダブルクリックしてレイヤーコントロールダイアログを開き、データセットの登録などを行おう。

ダイアログ中では、各レイヤーは[プロット名:レイヤー番号]という形式で表示される。枠や軸、データセットの設定はレイヤーごとに独立しているので、さまざまなスケールのグラフを表示できるわけだ。

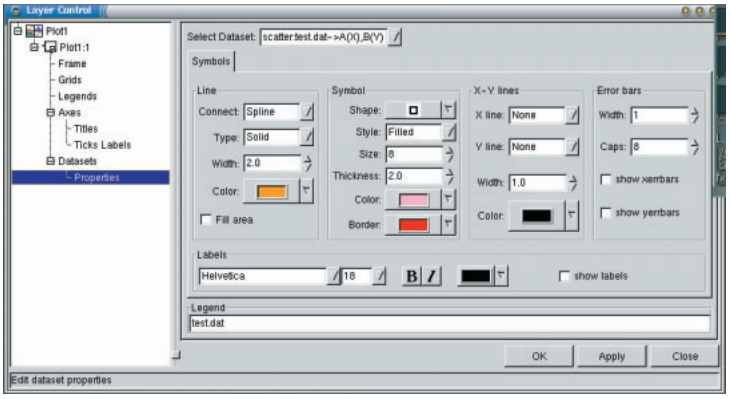
なお、異なるスケールのグラフを同じ位置に描画する場合は、軸のラベルや数値の表示が重ならないように注意しよう。これらは、各レイヤーの[Axes]で設定する。レイヤー番号が大きいものほど画面の手前に位置することに注意されたい。

一方、複数のグラフを別の位置に表示するには、グラフを直接操作したほうが簡単だ。移動したいグラフをクリックで選択し、そのままドラッグで移動すればいい。

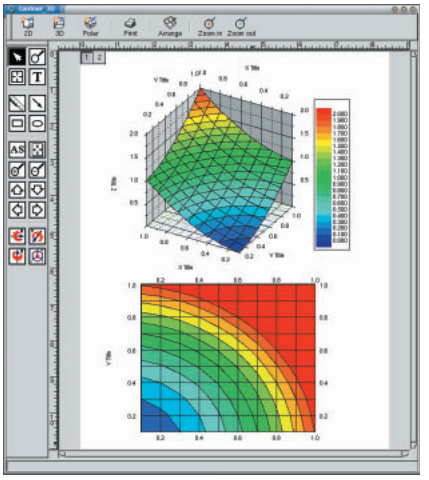
●グラフの保存と印刷

ワークシートの内容やプロットの設定を含む全データは、ひとつのプロジェクトとしてまとめられ、専用のXML形式（拡張子sg）でファイルに保存できる。メインウィンドウのツールバーの[保存]ボタンを押して、ファイル名を指定すればいい。

一方、作成したグラフを他のアプリで利用したり、プリンタで印刷したい場合は、グラフのツールバーの[Print]ボタンを押してPostScript形式で保存する。



画面7
データセットごとに線の種類やデータ点の記号などを設定できる。



画面8
複数のレイヤーを使えば、スケールの異なるグラフを重ねられる。

KDE 2用のグラフィックビューア

Gwenview

バージョン：0.7.0　　ライセンス：GPL

http://gwenview.sourceforge.net/home/

Gwenviewは、KDE 2上で動作するグラフィックビューアだ。ディレクトリツリーとファイルリスト（サムネイル）、イメージという3ペイン構成で、さまざまなディレクトリにある画像ファイルを素早く閲覧できる。それぞれのペインを独立したウィンドウとして表示したり、フルスクリーンで表示するなど、柔軟なレイアウトで使えるのが特徴だ。また、キーボードだけでも快適に操作できるように作られている。

●ビルドとインストール

Gwenviewは、tarボールとRPMバイナリ、ソースパッケージがそれぞれ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。

RPMパッケージは、Red Hat 6系ディストリビューションで利用できる形式になっている。バイナリパッケージを「rpm -Uvh gwenview-0.7.0-1.i386.rpm」としてインストールするか、ソースパッケージを「rpm --rebuild gwenview-0.7.0-1.src.rpm」としてリビルドし、/usr/src/redhat/RPMS/i386に作成されたバイナリパッケージをインストールすればいい。

一方、tarボールを利用する場合は、「./configure」→「make」でビルドし、「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、Vine Linuxなど一部のディストリビューションでは、あらかじめconfigureスクリプトの5477行目と5478行目の末尾に「/kde」を追加しておく必要がある。

●柔軟なレイアウトで使う

ktermなどのコマンドラインで「gwenview&」とするか、KDEメニューの[マルチメディア]－[Graphics]－[Gwenview]を選択するとウィンドウが開く。まるでメールクライアントソフトのような3ペイン構成だ（**画面1**）。

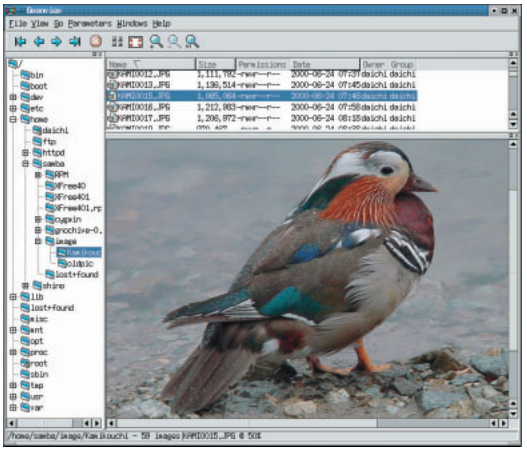
左側のディレクトリツリーにはルートディレクトリ以下のディレクトリ構造がツリー表示され、クリックで選択したディレクトリ内の画像ファイル一覧が右上のファイルリストに表示される。リスト中の画像ファイルをクリックすると、実際のイメージを右下のペインで確認できる。

なお、各ペインはいずれもdockedウィンドウなので、ペイン上部の薄い「グリップ」をドラッグすることで、ウィンドウ内での配置を柔軟に変更できる（**画面2**）。いくつかのペインを隠したり、独立したウィンドウとして表示することも可能だ（**画面3**）。なお、独立したウィンドウを元の状態に戻すには、グリップ右端の矢印ボタンをクリックすればいい。

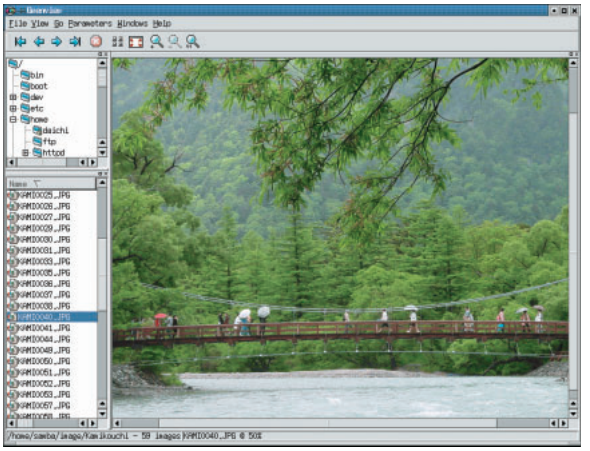
このほか、ツールバーの[Thumbnails]ボタンかCtrl－Tキーを押すことで、ファイルリストの表示がサムネイル（縮小画像）に切り替わる（**画面4**）。サムネイルのサイズは縦横とも60ドットだが、[Parameters]－[Configu

スペース	次の画像に切り替え（Forward）
BackSapce	前の画像に切り替え（Back）
Home	先頭の画像に切り替え（First）
End	末尾の画像に切り替え（Last）
Ctrl－T	サムネイル表示に切り替え（Thumbnails）
Ctrl－F	フルスクリーン表示に切り替え（Full screen）
Ctrl－＋	画像を拡大（Zoom in）
Ctrl－－	画像を縮小（Zoom out）
F5	画像をコピー（Copy）
F6	画像を移動（Move）
F2	画像の名前を変更（Rename）
Delete	画像を削除（Delete）
Escape	処理の中断（Stop）

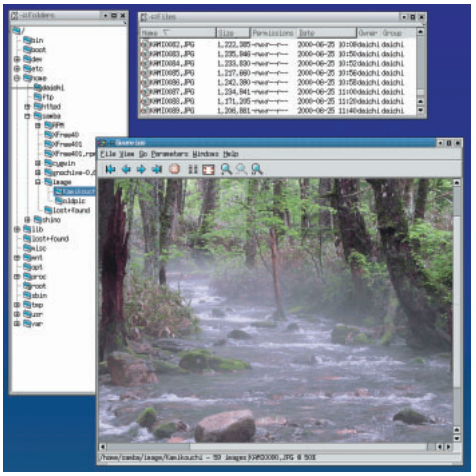
表1　Gwenviewのキー操作



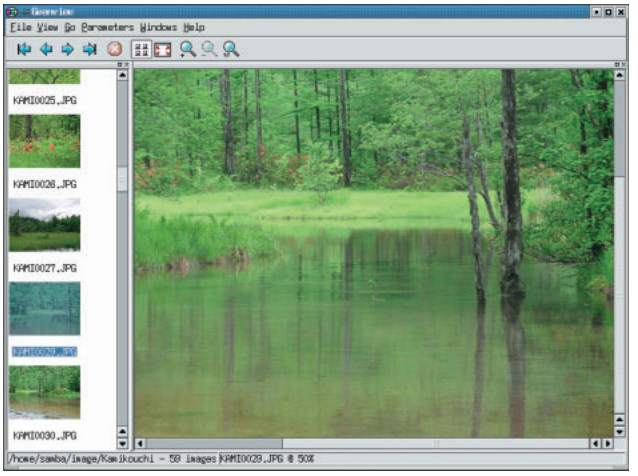
画面1　ウィンドウはメールクライアントのような3ペイン構成だ。



画面2　このように、各ペインのレイアウトは柔軟に変更できる。



画面3　各ペインを独立したウィンドウとして表示することも可能だ。



画面4　ファイルリストはサムネイル（縮小画像）表示に切り替えられる。

ration]で開く設定ダイアログで、20～100ドットに変更可能だ（**画面5**）。

●前後の画像に切り替える

多数の画像が含まれるディレクトリでは、いちいちファイルリストの画像をクリックして選択するのは面倒だ。Gwenviewでは、マウスやキーボードを使って簡単に前後の画像に切り替える方法が用意されている。

マウスを利用する場合、次の画像には「左ボタンを押したまま右ボタンのクリック」、前の画像には「右ボタンを押したまま左ボタンのクリック」で切り替わる。

この操作方法は目新しく、慣れるのにちょっと時間がかかるもしれない。ちなみにフリーなWebブラウザのOperaでも採用されている。

一方、キーボードを利用する場合は、スペースキーで次の画像、BackSapceキーで前の画像に切り替わる。このほか、Homeキーで先頭、Endキーで末尾の画像に飛ぶなど、いくつかのキーが用意されている（**表1**）。こうしたキー割り当ては、[Parameters]－[Keys]で表示されるキー設定ダイアログで変更できる（**画面6**）。

まずは上部のリストから変更したいアクションを選択し、下部の[Custom key]をクリックする。キーの設定は、修飾キー（Shift／Ctrl／Altキー）と、それ以外の通常キーに分かれている。通常キーの設定では、右下のボタンをクリックして押された状態にしてから、変更後のキーを押せばいい。

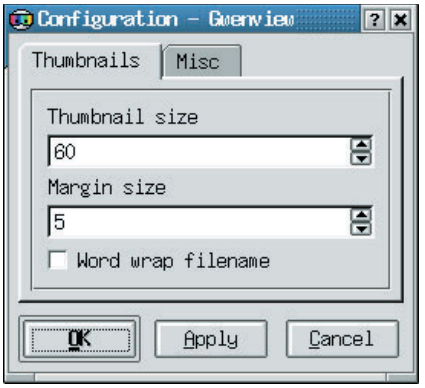
●画像の拡大／縮小など

現時点では、画像のサイズが右下のペインより大きな場合でも、自動的なスケールの調整は行われない（将来改善される予定）。こうした画像に対しては、画像上でのドラッグにより表示されていない部分をスクロールして眺められる。

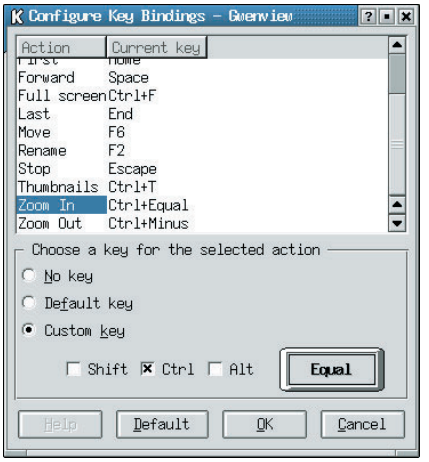
また、ツールバーの[Zoom out]ボタンやCtrl－－（マイナス）キーを押すと、50%のサイズまで縮小可能だ。逆に、[Zoom in]ボタンやCtrl－＋ボタンで400%まで拡大することもできる。

さらに、[Full screen]ボタンやCtrl－Fキーを押してフルスクリーン表示に切り替えると、他のウィンドウが一時的に隠された状態で画像を眺められる。この状態でも、マウスやキー操作による画像の切り替えは有効だ。再度Ctrl－Fキーを押すと元のウィンドウ表示に戻る。

Gwenviewには、画像ファイルを整理するのに便利なファイル処理機能もいくつか用意されている。たとえば、F5（F6）キーでは、現在表示中の画像ファイルのコピー（移動）、Deleteキーでは画像ファイルの削除（確認付き）が可能だ。ただし、ドラッグ&ドロップには対応していない。



画面5　サムネイルのサイズなどを変更できる設定ダイアログ。



画面6　キー割り当てはこのダイアログで柔軟にカスタマイズできる。

太陽の位置や昼夜を表示する世界時計

sunclock

バージョン：3.46　　ライセンス：GPL

http://frmas.free.fr/li_1.htm

sunclockは、世界地図に現在の太陽の位置や昼夜のようすを表示する時計ソフトだ。マップの一部を拡大したり、世界各地の都市の現在時刻や、日出／日没時刻を調べたりすることも可能だ。また、世界地図の表示には、標準ではベクトルマップが使われるが、別配布のマップパッケージの導入により、さらに見栄えのする各種のビットマップ画像を利用できる。

●ビルドとインストール

sunclockとマップパッケージは、どちらもtarボールとRPMバイナリパッケージが配布されているので、使用しているディストリビューションに合わせていずれかを選択しよう。

RPMバイナリパッケージは、Red Hat 6系ディストリビューションでそのまま利用できる形式だ。「su -」としてrootになった状態で、「rpm -Uvh sunclock-3.46-1.i386.rpm sunclock_maps_package-1.0-1.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、展開先のディレクトリで、「xmkmf」→「make」としてビルドし、「su」でrootになってから「make install」としてインストールする。続いて、rootになった状態のまま、マップパッケージのtarボールを「tar zxvf sunclock_jpg_maps.tgz -C /usr/share/sunclock」として展開しよう。

●クロックウィンドウの操作

ktermなどのコマンドラインから「sunclock&」として起動すると、小さなウィンドウが開いて、世界地図と現在の日時が表示される。この状態をsunclockでは「クロックウィンドウ」と呼んでいる。他のソフトの邪魔にならないように、ウィンドウサイズの初期値は小さめになっている。ウィンドウは自由にサイズ変更できるので、画面が広い環境ではもう少し大きくしたほうが見やすいだろう（**画面1**）。

初期設定では、世界地図はベクトルマップで表示され、夜の領域がくっきりと暗く表現される。また、太陽や月の現在位置（それらが真上に見える地点）には、それぞれのアイコンが表示されている。

こうした表示を変更したり、別のマップ画像に切り替えるには、Hキーを押して横長のメニューウィンドウを開く（**画面2**）。このメニューは、マウス操作用のボタンと、ショートカットキーの表示を兼ねている。

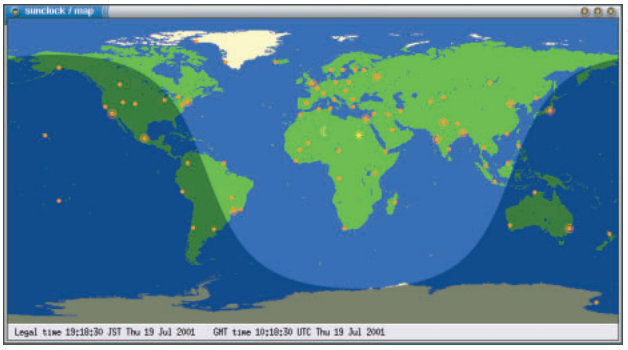
たとえば、画像を切り替える場合には、メニューウィンドウの[F]ボタンを押すか、メニューを開かずに直接fキーを押せばいい。ファイル選択ダイアログで適当なマップ画像を選択すると、クロックウィンドウの表示が即座に切り替わる（**画面3**）。

このほか、表示時間を進め（戻し）たり、夜の部分の表示方法を変更したり、太陽・月のアイコンを消すといった操作も可能だ。ただし、文字が赤く表示されているボタンは、クロックウィンドウでは利用できない。もうひとつの表示モードである「マップモード」に切り替える必要がある。

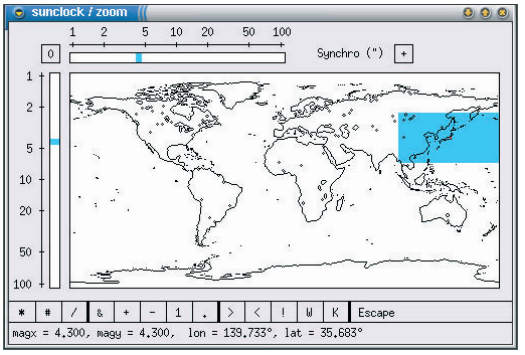
画面1　小さな世界地図に昼夜のようすが描かれるクロックウィンドウ。

画面3　表示に使うマップ画像を「depthmap.jpg」に切り替えてみた。

画面2　Hキーで呼び出すメニューウィンドウ。キー操作も可能だ。



画面4　世界各地の都市が円形アイコンで表示されるマップウィンドウ。



画面5　マップウィンドウのズームはこのウィンドウから制御する。

●マップウィンドウに切り替える

クロックウィンドウからマップウィンドウに切り替えるには、メニューウィンドウの[!]ボタンを押せばいい。また、[W]ボタンを使うと、クロックウィンドウはそのままで、マップウィンドウが新たに開く。

マップウィンドウの初期サイズは、クロックウィンドウよりも大きく、世界各地の都市の位置が円形のアイコンで表示されている（**画面4**）。マップ画像の設定はクロックウィンドウとは独立しているので、こちらも[F]ボタンを使って、見栄えのするマップ画像に切り替えておくとよい。

マップウィンドウを右クリックするか、メニューウィンドウの[Z]ボタンを押すと、ズーム制御用のウィンドウが開く（**画面5**）。ここでは、表示範囲や拡大率を変更して、マップの一部を拡大表示できる（**画面6**）。

なお、初期設定では、設定変更後に[＊]ボタンを押すまで設定が有効にならないが、左上の[Synchro]を「＋」に変更しておけば、即座に設定がマップに反映されるようになる。

●マップウィンドウの5つのモード

マップウィンドウでHキーを押してメニューウィンドウを開くと、クロックモードでは無効だった機能を含む全機能を利用できる。

以下では、マップ上の都市やその他の地点をクリックで指定したときの動作が異なる5つのモードを紹介しよう。先頭の文字のボタン（またはキー）を押すとモードが切り替わる。

[L] リーガルタイムモード

既定のタイムゾーンの日時とGMT日時が並んでマップ下に表示される。このモードのみクロックウィンドウでも利用可能だ（表示形式は異なる）。

[C] 座標モード

マップ上の都市（円形アイコン）をクリックすると、その都市の緯度と経度、現地時間、日出／日没時刻がマップ下に表示される。

[S] 太陽モード

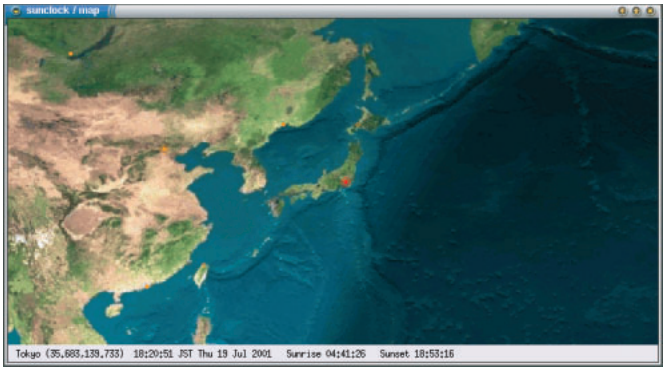
マップ上の任意の地点（都市以外でもよい）をクリックすると、その地点の太陽時と日照時間がマップ下に表示される。

[E] 時間拡張モード

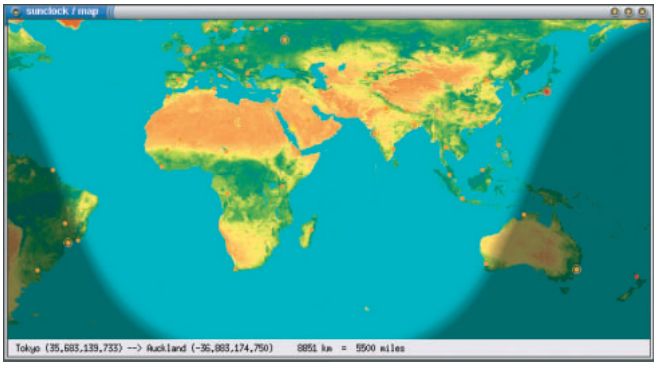
現時点での世界各地の太陽時（0～23）がマップ下に表示される。

[D] 距離モード

マップ上の任意の2点をクリックで指定すると、それらの間の距離がマップ下に表示される。たとえば、東京（日本）とオークランド（ニュージーランド）間は8851kmある、といったことが簡単に調べられる（**画面7**）。



画面6　ためしに、日本とその周辺を拡大表示してみた。



画面7　距離モードを利用して東京・オークランド間の距離を調べる。

キー操作で効率的なファイル処理を行う

vshnu

バージョン：1.0010　　ライセンス：GPL/QPL

http://www.cs.indiana.edu/~kinzler/vshnu/

vshnuは、コンソールや端末画面にファイル一覧を表示して、キーボードのみで効率的にファイル処理を行うビジュアルシェルだ。英字キーでファイルを選択できるbag機能が特長で、ファイルの名前（拡張子含む）や種類に応じて処理を変えることも可能だ。ファイル一覧の表示方法を切り替えたり、キー割り当てを柔軟にカスタマイズすることもできる。実行にはPerl 5.002以降が必要だ。

●本体とモジュールのインストール

vshnuはPerlのスクリプトとして記述されているため、通常のビルド作業は不要だ。「perl Makefile.PL」→「make install」としてインストールすればいい。

なお、vshnuの実行には2つのPerlモジュール（Term::ANSIColor、Term::Screen）が必要で、存在しない場合はvshnuの起動時にエラーが表示される。これらはvshnuのtarボールにも含まれているので、以下のようにしてインストールしよう。

まず、tarボールの展開先で「su」としてrootになった状態で、「cp -r libperl/Term /usr/lib/perl5/site_perl/5.005」としてコピーする。続いて「cd /usr/lib/perl5/site_perl/5.005/Term」でカレントディレクトリを移動してから、「mv ANSIColor ANSIColor.pm」「mv Screen Screen.pm」とすればいい。

●起動と画面構成

Xを起動していない画面（コンソール）や、X上のktermやrxvtなどの端末画面で「vshnu」とすると、カレントディレクトリのファイル一覧が表示される（**画面1**）。このとき、lsコマンドのように（環境変数LS_COLORSに基づいて）ファイルがカラー表示される。

ファイル一覧は、L（大文字）キーで詳細表示（**画面2**）に切り替えられる。元に戻すにはCtrl－Lキーを押せばいい。また、F（大文字）キーでは、fileコマンドで調べたファイルの種類が表示される。

画面右上には、ファイル列の表示範囲と総数を示す数字が「1-4/8」といった形式で表示されている。範囲の末尾の数字が総数より小さい場合は、スペースキーで次のページに切り替えられる。Ctrl－[、Ctrl－]キーで1列ずつシフトさせることも可能だ。

vshnuを中断せずに、シェルコマンドを実行したい場合は、:キーか;キーを押す。:キーはコマンドラインや実行結果が画面に残り、;キーでは残らない。このほか、複数のコマンドを連続実行できる$キーも用意されており、「V」のみ入力するとvshnuに戻る。vshnuを終了するには、Ctrl－Qキーを押してからQキーを押せばいい。

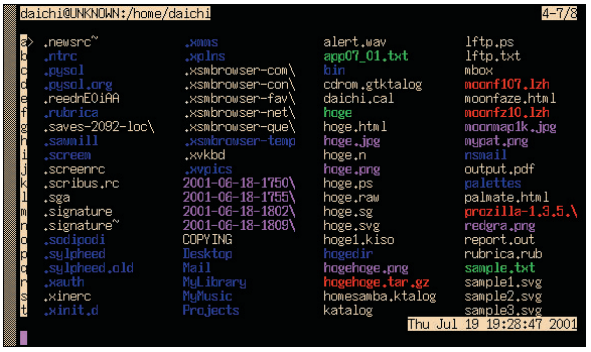
●「bag」によるファイルの選択

ビジュアルシェルでは、さまざまな処理に先立つファイルの選択操作が使い勝手を左右する。一般には、反転表示の「カーソル」をカーソルキーで移動させ、スペースキーなどで選択するといった操作が用いられる。

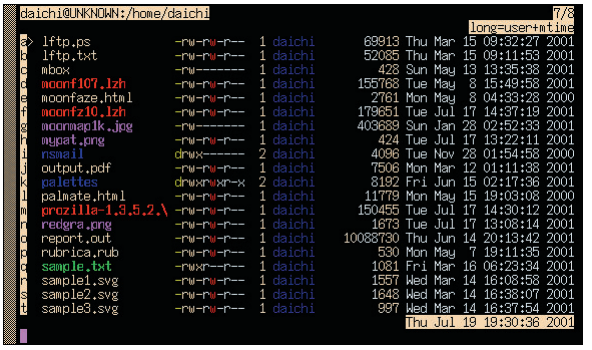
これに対し、vshnuでは「bag」と呼ばれる特異な方式を採用している。これは、ファイル一覧の列のひとつに英小文字のラベルを付け、英字キーを押すだけで対応するファイルが選択されるという仕組みだ。

初期状態では、bagは左端の列に付けられているが、Tab（またはCtrl－I）キーで右、BackSpace（またはCtrl－H）キーで左にひとつずつ移動する。これで、ホームポジションから手を動かすことなく、すべてのファイルを選択できるわけだ。

起動直後のメインモードでは、ディレクトリを選択するとカレントディレクトリが変更され、ファイルを選択し

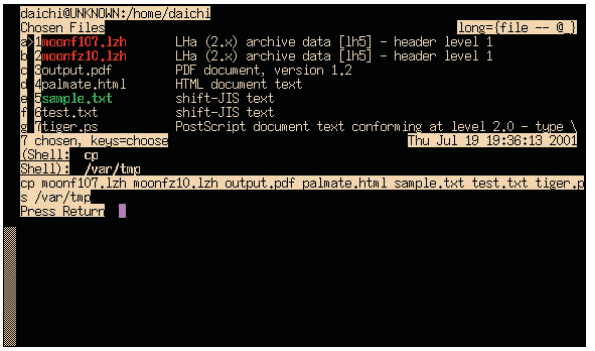


画面1　カレントディレクトリのファイル一覧が表示される。



画面2　Lキーを押すと、「ls -l」と同様の詳細表示に切り替わる。

画面3　Ctrl－Cキーを押すと、選択状態のファイルだけが一覧表示される。



画面4　選択モードで:キーを使えば選択したファイルのコピーが可能だ。

た場合はエディタ（環境変数EDITORに設定）が起動してそのファイルが編集状態になる。

複数のファイルを選択した後でファイルのコピーを行ったり、ファイル名のパターンに応じて処理を変えたりするには、後述のキー操作でキーコマンドモードやファイルアクションモードを変更する必要がある。

●選択モードで複数ファイルを処理

/キーを押してキーコマンドモードを選択モードに切り替えると、画面下に「keys=choose」と表示される。元のメインモードに戻るには、もう一度/キーを押せばいい。

選択モードでは、bagを使ってファイルを選択すると、ファイル名の左に選択番号が付き、複数のファイルを選択できるようになる。また、Ctrl－Cキーを押すと、選択したファイルだけが一覧表示される（**画面3**）。

さらに、!キーに続けてPerlの正規表現パターン「/文字列/」を指定して、マッチするファイルを一括選択することも可能だ。たとえば、「/^\./」というパターンを指定すると、ファイル名が「.」で始まるファイルをまとめて選択できる。

このモードでは、メインモードのキー操作の動作の一部が変更されるほか、選択したファイルを対象とするキー操作がいくつか追加される。

シェルコマンドを実行する:キーは、コマンドラインの入力が2回に分割され、選択したファイル名が両者の間に挿入された状態で実行される。たとえば、選択したファイルを/var/tmpにコピーする場合、:キーを押した後で、「cp」と「/var/tmp」をそれぞれ入力すればいい（**画面4**）。

一方、選択モード専用のキー操作としては、chmod／chown／chgrpコマンドを実行するCキー、ファイルを削除するDキーやRキー、画像を表示するXキーなどがある。

なおこれらのキー操作は、bagのファイル選択操作と区別するため、いずれも大文字で入力することに注意されたい。

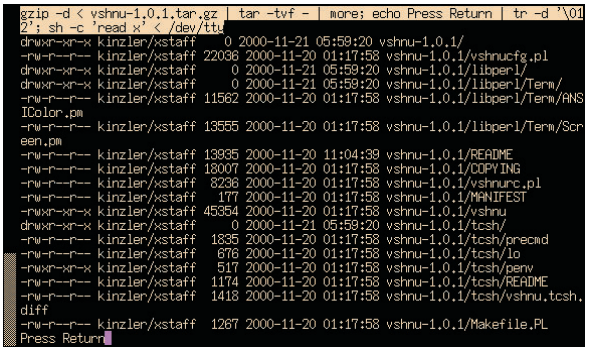
●ファイル名に応じた処理を行う

?キーを押してファイルアクションモードをdoモードに切り替えると、画面下に「types=do」と表示される。再度?キーを押すと、元の編集モードに切り替わる。

doモードでは、選択したファイルの種類やファイル名パターンに応じて異なる処理を行う。たとえば、tarボールなどのアーカイブは内部のファイル一覧が表示され（**画面5**）、画像ファイルはイメージが表示され、MP3ファイルは再生される。

こうしたファイル名パターンとコマンドの組み合わせは、＆キーで確認できる（**画面5**）。ただし、Perlの正規表現パターン「/文字列/」や、ファイルテスト演算子（「-d」でディレクトリチェックなど）がそのまま表示されるので、Perlを使った経験がないと読み難いかもしれない。

こうした設定は、/usr/lib/perl5/site_perl/5.005/vshnucfg.plに記述されている。コマンドを差し替えたり、ファイル名パターンを追加・変更したければ、このファイルを書き換えればいい。また、ホームディレクトリの「.vshnurc」というファイルで設定を追加することもできる。



画面5　doモードでtarボールを選択すると内部のファイル一覧を表示。

RPMパッケージを簡単に作成してインストール

CheckInstall

バージョン：1.4.1　　ライセンス：LGPL

http://mayams.net/~izto/checkinstall-en.html

CheckInstallは、tarボールから簡単にRPMパッケージやSlackwareパッケージを作成できるシェルスクリプトだ。同梱のツール「Installwatch」を利用して、「make install」でインストールされるファイルを調べ、RPMバイナリパッケージ化してインストールしてくれる。このため、「rpm -e パッケージ名」とするだけで簡単にアンインストールできる。また、インストール前のファイルをtarボールにバックアップする機能も備えている。

●ビルドとインストール

CheckInstallは、tarボールとRPMバイナリパッケージが配布されている。RPMバイナリパッケージはRed Hat 6系ディストリビューションでそのまま利用できる形式だ。

以下では、CheckInstallの機能の実演も兼ねて、tarボールからインストールすることにしよう。まず、tarボールの展開先で「make」としてビルドする。CheckInstall自体はシェルスクリプトなのでビルドする必要はないのだが、InstallwatchがC言語で書かれているためだ。

続いて、「su」としてrootになった状態で、「make install」として、CheckInstallとInstallwatchを通常の手順でインストールする。CheckInstall本体は/usr/local/sbinに、Installwatchは/usr/local/binにそれぞれコピーされる。

●CheckInstallをRPMパッケージ化

これで、CheckInstallが利用できる状態になったので、手始めにCheckInstall自体をRPMパッケージ化してみることにしよう。

「su -」（「-」に注意）としてrootになった状態で、CheckInstallのtarボール展開先ディレクトリに移動し、「checkinstall」とする。なお、「su」としてrootになった場合は、/usr/local/sbinがPATHに含まれないため、「/usr/local/sbin/checkinstall」として起動すること。

CheckInstallが起動すると、「make install」でインストールされるファイルがInstallwatchによって調査された後で、利用するパッケージの種類を尋ねてくる（**画面1**）。RPMパッケージの場合は「r」を入力すればいい。

続いて、RPMパッケージの作成に必須のデータ（パッケージ名称やバージョン番号など）が一覧表示され、修正するかどうか尋ねてくる（**画面2**）。ここでは修正の必要はないので、そのままEnterキーを押そう。

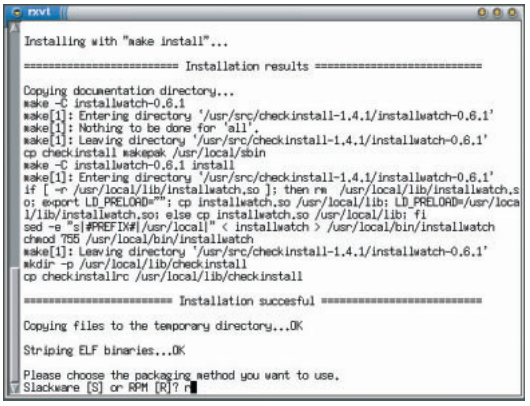
あとは、自動的にRPMバイナリパッケージが作成され、rpmによるインストールが行われて作業は完了する（**画面3**）。なお、/usr/src/redhat/RPMS/i386には、このとき作成されたバイナリパッケージのファイルが保存されている。

これ以後は、「rpm -ql checkinstall」としてインストール済みのファイル一覧を確認したり（**画面4**）、「rpm -e checkinstall」として簡単にアンインストールできるようになる。

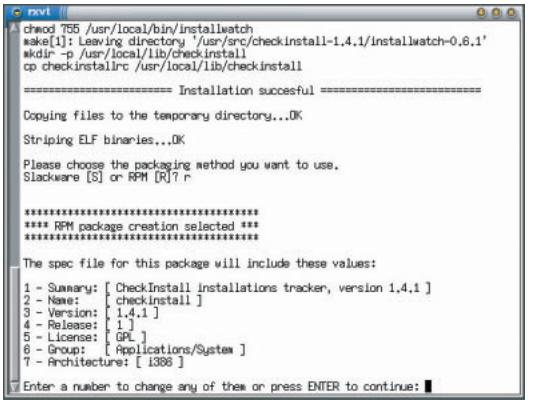
●そのほかのソフトの場合は

そのほかのソフトの場合も、基本的な手順は同じだ。

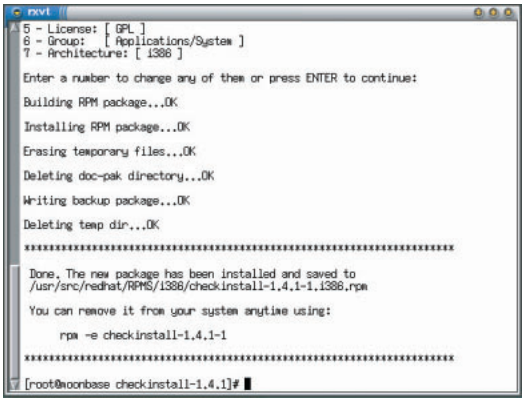
(1) tarボールを展開し、cdコマンドで展開先ディレクトリに移動する。
(2)「./configure」→「make」などの手順でビルドを行う。
(3)「su -」でrootになり、tarボール展開先にcdコマンドで移動する。
(4)「checkinstall」を実行する。なお、特殊な方法でインストールする場合は、「checkinstall setup」といった具合に、インストール用



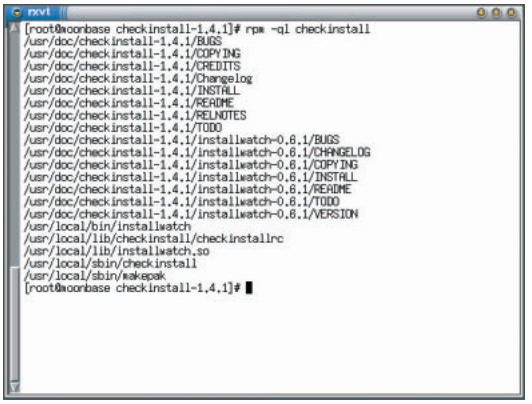
画面1　RPMパッケージとSlackwareパッケージのいずれかを選択する。



画面2　SPECファイルの記述が一覧表示される。修正することも可能だ。



画面3　RPMバイナリパッケージの作成とインストールなどが行われる。



画面4　このようにRPMパッケージとしてインストールされた状態になる。

コマンドラインを引数に指定すればいい。

(5)「exit」で一般ユーザーに戻る。

ただし、CheckInstallの実行に先立って、以下のような準備作業が必要となる。CheckInstallのtarボールは、こうした作業がすでに完了した状態で配布されているのだ。

・doc-pakにドキュメントをコピー

RPMバイナリパッケージに含めたいドキュメント類（READMEやCOPYINGなど）を、tarボール展開先のdoc-pakディレクトリにコピーする。

tarボール展開先で「mkdir doc-pak」としてディレクトリを作成後、「cp README COPYING doc-pak」などとすればいい。

なお、doc-pakディレクトリを作成せずにCheckInstallを実行すると、定型的なドキュメント類をコピーするかどうか尋ねてくる。ここで「y」を入力すれば、READMEやCOPYINGなどは自動的にコピーされる。

・パッケージの説明を作成

RPMパッケージの詳しい説明（サマリ）を、tarボール展開先のdescription-pakというファイルに書いておく。この説明は、「rpm -qi パッケージ名」とすると画面に表示される。

description-pakを作成しないでCheckInstallを実行した場合、その場で説明文を書くように求められるので、簡潔な英語で数行の説明文を書こう（**画面5**）。行頭でEnterキーを押すと入力完了だ。

●バックアップ用のtarボール

CheckInstallによるインストールと同時に、バックアップ用のtarボールも作成されている。カレントディレクトリに、「backup-日時-pre-パッケージ名-バージョン.tgz」というファイルが作成され、インストールで上書きされる前のオリジナルファイルが自動的に退避されるのだ。

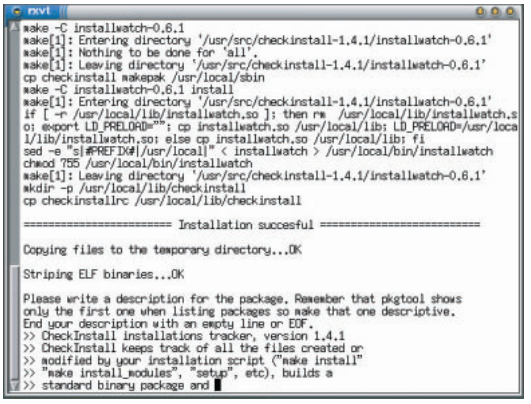
もし、CheckInstallでインストールしたソフトに不具合がある場合は、すぐに「su」としてrootになり、「rpm -e パッケージ名」としてパッケージをアンインストールする。さらに、ルートディレクトリ（/）に移動してから、「tar ztvf バックアップtarボール名」としてオリジナルファイルを書き戻せば、すべてのファイルを元の状態に戻せるわけだ。

●Installwatchの単独利用

Installwatchは、単独で実行することもできる。RPMパッケージによる管理までは必要ないものの、インストールされるファイル情報を把握しておきたい場合に利用するとよい。

使い方は簡単で、ファイルのコピーや削除などを伴なうコマンドのコマンドラインを、Installwatchの引数として指定すればいい。また、ファイル情報が保存されるファイルは-oオプションで指定する。

たとえば、「installwatch -o hoge make install」とすると、「make install」が実行されるとともに、作成・変更されたファイルの情報がhogeというファイルに保存される。



画面5　RPMパッケージの説明文（サマリ）をその場で書くことも可能だ。

さまざまな手法のステレオグラムを作成

Krosseye

バージョン：0.15　　ライセンス：GPL

http://w1.131.telia.com/~u13115035/index3.html

Krosseyeは、ランダムドットやパターンマップなど5種類の手法を利用したステレオグラム（立体視画像）を作成するソフトだ。利用する手法の選択をはじめ、デプスマップやパターンマップといった画像の読み込み、さまざまなパラメータの調整などを、すべてグラフィカルな環境で行えるのが特徴だ。作成したステレオグラムはPNG形式の画像として保存できる。動作にはQt 2.2以降が必要だ。

●GUIでステレオグラムを作成

Krosseyeはtarボールのみ配布されている。ビルドとインストールは、configureスクリプトを利用する一般的な手順でOKだ。

ktermなどから「krosseye&」とすると、左側に操作用のボタンやリスト、右側に画像表示用エリアを持つウィンドウが開く。左上の[Render method]で、ステレオグラムの作成手法を選択できる。まずは、単純な「Ramdom colors」を選択しておこう。

続いて、ステレオグラムの奥行き（深度）情報を与える画像を読み込む。[File]－[Load Depthmap]でダイアログを開き、適当な画像（PNG／XPM／JPEG形式）を選択すればいい。読み込んだ画像は、ウィンドウ右の[Depthmap]ページに表示される（**画面1**）。

こうしたデプスマップ画像はグレースケールが望ましいが、カラー画像でも大丈夫だ。Krosseyeのtarボールにも数個のサンプル（depthmap_＊.png）が含まれている。

[Depthmap]以下の設定で、深度の深さなどを調整できる。なお、初期設定では「交差法」用のステレオグラムが作成されるので、「平行法」で見たい場合は[Invert]をチェックにしておく必要がある。

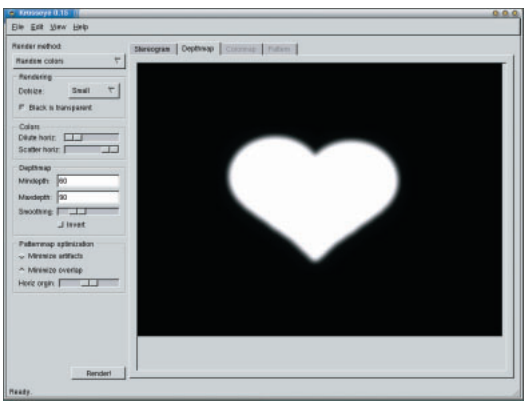
最後に、左下の[Render]ボタンを押すと、右の[Stereogram]ページに、ランダムな色の点で表現されたステレオグラムが表示される（**画面2**）。

●作成手法を変更する

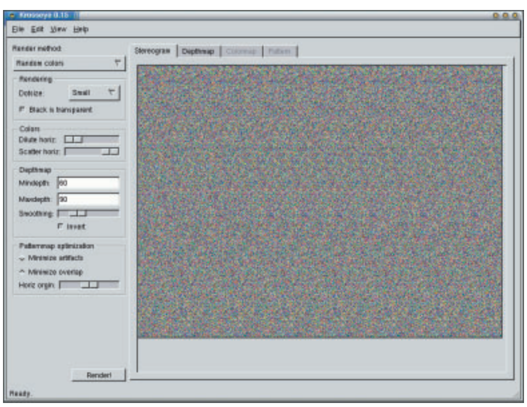
Krosseyeでは、このほかにも、使用する色の種類を限定した「Color mapped」と「Colordotted」、ドットの代わりにテクスチャ系パターンを利用する「Patternmapped」と「Patterndotted」の4手法が用意されている。

これらの手法を利用する場合は、デプスマップ画像のほかに、カラーマップやパターンマップ用の画像を読み込む必要がある。対応する画像形式や、読み込む方法はデプスマップ画像の場合とほぼ同じだ。

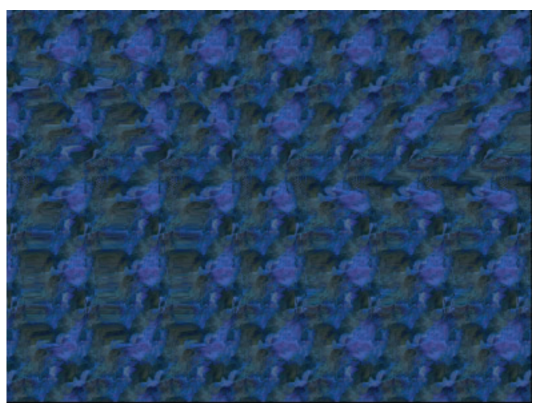
たとえば、tarボールに付属するdepthmap_propellar.pngをデプスマップ、pattern_6a.jpgをパターンマップとして利用して作成したステレオグラムを**画面3**に示す（平行法用）。



画面1
さまざまな手法によるステレオグラムをGUIで作成できる。



画面2
「Random colors」という手法でステレオグラムを作成する。



画面3
テクスチャ系パターンを利用したステレオグラムも作成可能だ。

ファイルを分割して高速ダウンロード

Prozilla

バージョン：1.3.5.2　　ライセンス：GPL

http://prozilla.delrom.ro/

Prozillaは、HTTP/FTPを利用してファイルを高速にダウンロードするコンソール系ツールだ。ファイルを複数に分割して同時に受信するため、ブロードバンド環境では通常の数倍の速さでのダウンロードが期待できる。このほか、もっとも高速なミラーサイトを自動的に検索するFTPサーチ機能や、中断した場所からダウンロードを再開するレジューム機能などを備えている。

●ビルドとインストール

Prozillaは、tarボールとRPMバイナリ、ソースパッケージが配布されている。tarボールからのビルドとインストールは、「./configure」→「make」としてビルドし、「su」でrootになってから「make install」という一般的な手順だ。

一方、RPM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は、tarボールをそのまま利用するか、以下の手順でtarボールからRPMパッケージを作成するとよい。

まず、tarボールを展開して、prozilla.specの3行目の「1.3.3.3」を「1.3.5.2」に変更する。続いて、tarボール自体を/usr/src/redhat/SOURCESにコピーしてから、「rpm -tb prozilla.spec」とする。これで、/usr/src/redhat/RPMS/i386にバイナリパッケージが作成されるので、通常の手順でインストールしよう。

●ファイルを分割ダウンロード

Prozillaを利用するには、ダウンロードしたいファイルのURL（「http://～」や「ftp://～」）をコマンドラインに指定すればいい。たとえば、

```
$ proz http://prozilla.delrom.ro/
tarballs/prozilla-1.3.5.2.tar.gz
```

とすると、Prozillaのtarボールがダウンロード対象となる。

ダウンロード中は、端末画面の背景が黒くなり、ファイルの情報やダウンロードの進行状況などが表示される（**画面1**）。ダウンロードを一時中断する場合はCtrl－Rキー、中止する場合はCtrl－Xキー、画面を再描画する場合はCtrl－Lキーを押せばいい。

なお、初期設定では、ファイルは4つに分割されて同時にダウンロードされる。分割数は起動時の－kオプションで変更可能だ。たとえば、2分割で受信したい場合は

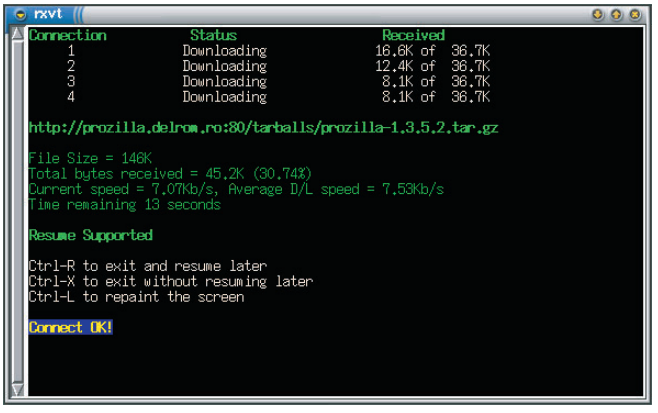
```
$ proz -k=2 http://～
```

などとすればいい。

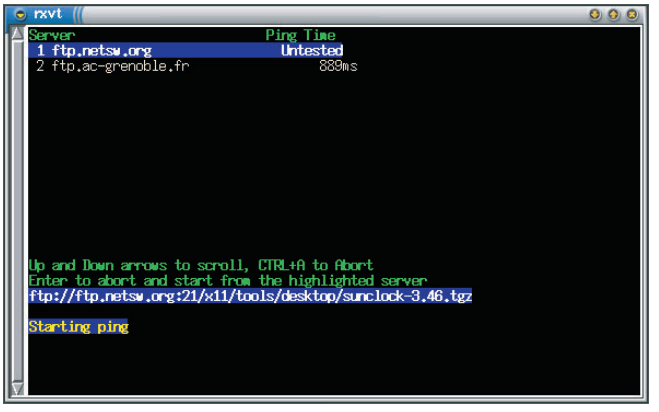
●FTPサーチ機能を利用する

起動時に－sオプションを付けると、指定したファイルを持つミラーサイト一覧を取得し、pingに対するレスポンスがもっとも良いサイトからダウンロードを行う「FTPサーチ機能」が有効になる。ミラーサイト検索中は、各サイトのドメイン名やレスポンス時間などが一覧表示される（**画面2**）。

このほかにも、ファイルの保存先ディレクトリを指定する－Pオプションや、バンド幅の最大値を制限する--max-bpsオプション、同時に張るコネクション数を1に制限する-1オプションなど、さまざまなオプションが用意されている。詳細は「man proz」で表示されるマニュアルページを参照されたい。



画面1
ひとつのファイルを4つに分割して同時にダウンロードする。



画面2
もっとも高速なミラーサイトを探してくれるFTPサーチ機能。

麻雀牌を使ったパズルゲーム

Xmahjongg

バージョン：3.5　　ライセンス：GPL

http://www.lcdf.org/~eddietwo/xmahjongg/

Xmahjonggは、麻雀牌を利用したパズル「上海」によく似たソリティアゲームだ。さまざまな形に配置された麻雀牌の中から、端にある同種の牌を見つけて取り除いていく。取れるペアの数やヒントの表示など、初心者向きのアシスト機能が充実しているほか、さまざまなデザインの麻雀牌（タイルセット）と盤面（レイアウト）が用意されており、普通の上海に飽きてしまった人でも楽しくプレイできる。

●ビルドから起動まで

Xmahjonggは、tarボールとRPMバイナリパッケージの両方が配布されている。国産ディストリビューションでは、tarボールからインストールしたほうがいいだろう。「./configure」→「make」でビルドし、「su」でrootになってから「make install」とする一般的な手順だ。

ktermなどのコマンドラインで「xmahjongg&」として起動すると、色鮮やかな麻雀牌が5段に積まれたウィンドウが開く（**画面1**）。

●基本的な遊び方

ゲームの目的は、同種の牌のペアを取り除いていき、すべての牌を盤面から消すことだ。取り除く牌はクリックで選択する。その際、それぞれの段の左右の端にある牌しかクリックできないので注意されたい。このへんの基本的なルールは、上海と同じだ。

最初の牌をクリックすると、その牌が選択状態（濃い黄色）になり、次に同種の牌をクリックすると、両方の牌が消える。違う種類の牌をクリックした場合は、最後にクリックした牌がかわりに選択状態になる。

ウィンドウ左上には、現在の残り牌を示す数値ともに、取り除ける牌のペアがいくつあるかを示す円形のアイコンが並んでいる。もし、プレイ中にこのアイコンがひとつも表示されなくなったら、それ以上プレイを続けられない（手詰まり）ということだ。

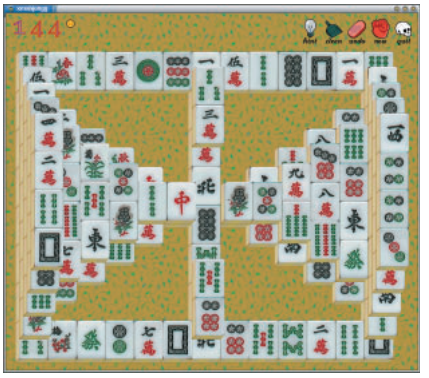
このほか、ウィンドウ右上には、ヒント（取り除ける牌のペア）を表示したり（**画面2**）、新規ゲームを始めたり、Xmahjonggを終了するといった操作のためのアイコンが並んでいる。

●タイルセットとレイアウトの変更

Xmahjonggには、麻雀牌のデザイン（タイルセット）が8種類、盤面のデザイン（レイアウト）が11種類用意されている。これらは、起動時の－t、－lオプションで指定する。たとえば、タイルセット「real」とレイアウト「hourglass」でプレイするには、

```
$ xmahjongg -treal -lhourglass&
```

とすればいい。リアルな麻雀牌と砂時計風の盤面が表示される（**画面3**）。こちらの牌のほうが綺麗なうえに、段の重なり具合も見極めやすいので、お勧めだ。



画面3　タイルセット「real」とレイアウト「hourglass」でプレイ中。

画面1
古典的な麻雀牌パズルゲームをLinuxでもプレイしよう。

画面2
初心者は、取り除ける牌のペアを教えてくれるヒントを活用。

色に着目した落下型パズルゲーム

fblocks

バージョン：0.08　　ライセンス：GPL

http://www.wolsi.com/~dwl/code/
http://www.libsdl.org/（SDL）

fblocksは、さまざまな形のブロックが落下してくるテトリス風のパズルゲームだ。それぞれのブロックは1色から3色に塗られていて、同じ色を4区画以上縦横に並べるとそれらの部分が消えるというルールだ。連鎖的にブロックを消したり、一度に複数のブロックを消したりして「コンボレベル」を上げると、通常よりも高得点になる。実行にはマルチメディアライブラリのSDLが必要だ。

●ビルドとインストール

まずは、SDLのインストールを行おう。最新版の1.2.1はtarボールとRPMパッケージの両方で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する。tarボールからのインストールは、「./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。

一方、fblocksのほうは、tarボールのみ配布されている。configureスクリプトを用いるビルドとインストールの手順はSDLとまったく同じだ。

●同色のブロックを縦横に並べる

ktermなどのコマンドラインで「fblocks&」とすると、ウィンドウが開いてすぐにプレイ開始だ（**画面1**）。上部から落下してくるブロックはテトリスと同じ形だが、それぞれ1色～3色に色分けされている点が異なる。ゲーム全体では、灰／黄／空／紫の4色が使われている。

このゲームでは、ブロックを並べた結果、同じ色の部分が縦または横に4つ以上並ぶと、その色の部分だけが消える（**画面2**）。もちろん、残りの部分がさらに落下することで、連鎖的にブロックが消えることもある。

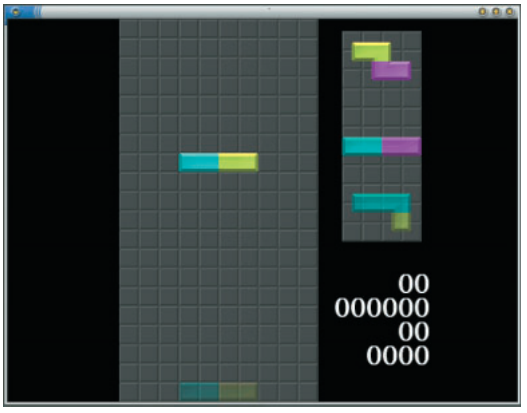
キー操作はカーソルキー系とvi系に大別できる。カーソルキー系では、←キーと→キーで移動、↑キーで回転（左回り）、↓キーで落下だ。vi系では、jキーとlキーで移動、iキーで反時計回り、oキーで時計回りの回転、kキーで落下となる。

このほか、スペースキーで下まで一気に落としたり、aキーで現在落下中のブロックを放棄して次のブロックに制御を移したり（**画面3**）、dキーですべてのブロックを強制的に落下させるといった操作も可能だ。

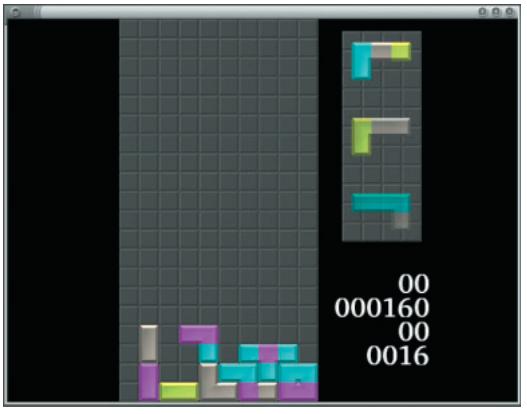
高得点を得るには、ブロックを消した点数を倍増させる「コンボレベル」の数値を増加させる必要がある。コンボレベルは、

- **連鎖的にブロックを消す**
- **消える途中のブロックにさらに同色のブロックをくっつける**
- **一度に複数の色のブロックを消す**

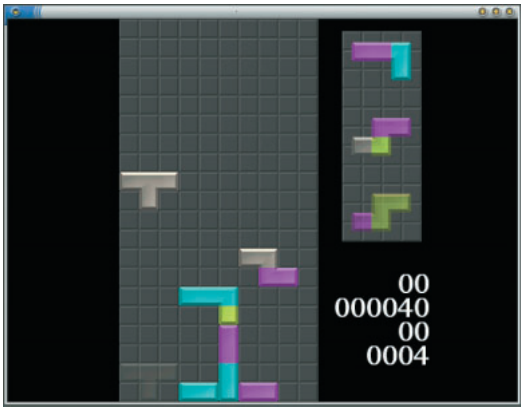
ことで増加し、コンボが途切れると0に戻る。



画面1　プレイが始まると、さまざまな色に塗られたブロックが落下してくる。



画面2　同色のブロックを4つ以上縦横に並べよう。



画面3　複数のブロックを同時に落下させることも可能だ。

Software Catch Up

予測変換機能や連想変換機能を搭載したかな漢字変換システム

Wnn7 Personal for Linux/BSD

長い歴史を持つWnnの最新バージョンが登場した。ワークステーションで培われた技術と、より正確なかな漢字変換エンジンを搭載したWnn7は、Linux上でのかな漢字変換を快適なものにするだろう。

文：塩田紳二
Text : Shinji Shioda

価格	9800円（2ライセンス）
問い合わせ先	オムロンソフトウェア ソフトウェアプロダクト営業部 044-246-6006 http://www.omronsoft.co.jp/

Wnn（うんぬ）は、ワークステーション全盛の時代に、京都大学、オムロン、アステックの共同開発で作られたかな漢字変換システムだ。当初の目標が「Watashino Namaeha Nakano desu」という入力を、正しく「私の名前は中野です」に変換することを目的としていたためこの名称になったと言われている。このWnnは、バージョン2が最初に公開され、前述した3者によりバージョン4までが開発された。その後、Wnnコンソーシアムおよび、オムロンに開発が引き継がれ、Wnn4.03からWnn4.2まで6つのバージョンが公開されたほか、中国語対応のcWnn4、韓国語対応のkWnn4も作られた。

Wnnは、変換／辞書検索部分とクライアント側で動くモジュールが独立しており、ネットワークを介した通信で両者が組み合わさって動作するようになっている。これは、かつてはワークステーションといえども、数Mバイト程度のメモリに数100Mバイト程度のハードディスクで動作しており、マルチユーザー環境では負荷を分散できるというメリットがあったためだ。また、このような構成とすることでマルチウィンドウ、マルチタスクといったUNIXの環境にもなじみやすいものとなっていたわけである。

Wnnは、フリーのバージョンが多くのディストリビューションに含まれているが、これは最終版であるWnn4.2をベースにしたものである。現在では、FreeWnnという名前となりGPLにて配布が行われ、FreeWnnプロジェクト（http://www.freewnn.org/）がメンテナンスを行っている。

これとは別に、Wnn6と呼ばれるオムロンソフトウェアが販売する製品版もある。このバージョンは、多くのクライアントに対応するために辞書検索と変換を別プロセスにした「大規模jserver」版をベースにしたものになっている。

つまり、WnnにはWnn4ベースのFreeWnnとWnn6の2つが存在していたのだが、今回紹介するWnn7は、Wnn6の直系の後継にあたるものだ。Wnn7は、予測変換などの機能を取り入れたもので、オムロンソフトウェアが販売する製品である。

Wnn7の特徴

カタログスペックによれば、今回のWnn7では、

- 予測変換機能（楽々入力）
- 連想変換機能
- 入力補正機能
- 逆引き変換機能
- 辞書強化

といった点が、従来のWnn6に対して追加されている。

予測変換機能（楽々入力）とは、過去に入力した文字列を記憶して、読みのローマ字を入れた段階で同じ読みから始まる文字列をリストとして提示する機能である。これは、ローマ字を入力していくごとに絞り込みが行われるほか、複数文節の入力では適当な文節に分けて記憶されるため、後半が違う入力の場合にも有効となる。たとえば、「毎度お世話になっております」という文章を入力した場合、次に「m」を入れただけで、「毎度お世話になっております」、「毎度」、「毎度お世話」の3つが候補に含まれるようになる。これを使うことで、頻繁に入力するフレーズを選択入力することが可能となり、大幅に入力が簡略化される。

連想変換機能は辞書内に類義語が登

録してあり、それを表示させる機能だ。似た意味を持つ、別の単語に置き換えて変換が可能となる。

入力補正機能はローマ字入力などにありがちなキー入力ミス、たとえば、nを2つ入力してしまい「んん」となってしまう場合や、母音の過不足などにより変換ができない場合にそれを修正して変換する機能である。

逆引き変換は漢字を読みに逆変換する機能である。読み方のわからない漢字などに対して、辞書を引く必要がなくなる。カット&ペーストを使うことで、文字列の入力も部首入力などを併用する必要がなく簡単だ。

辞書強化は従来バージョンの基本辞書を見直し、50万語を基本辞書としたのに加え、分野別の辞書を別に用意した。たとえば、郵便番号辞書や、市外局番辞書（市外局番を住所に変換）などのほか、地名、企業名といった辞書が多数用意されている。

Wnnの変換方式

当初の目的が複数文節からなる文を変換できることだったため、Wnnは文節区切りをなるべく正確に行うことと、同音異義語の処理を行っていた。

Wnn7では、文節区切りは「N文節文節長最長一致法」、「N文節文節数最小法」、「N文節評価値最大法」の3段階で行う。文節の分解は、基本的には辞書引きをベースにして行われるが、さらに助詞や接頭語、接尾語などの判定も行う。このようにしていくと、入力文字列が長くなればなるほど、何通りもの分割が可能になってしまう。このときに、どのような評価方法を使って正しい文節への分割を決定するかが、変換効率の違いになって現れる。

「N文節」とは、処理対象が制限のない複数文節でありことを示す。従来、コンピュータの処理能力の問題もあって、単文節や2文節といったように文節数を限定した処理が行われていた（特に先行していたMS-DOS上のかな漢字変換システム）のだが、N文節はこれらに対しての表現である。

文節長最長一致法は、入力された文字列を文節に分解する際に、個々の文節の長さがなるべく長くなるものを選択することであり、文節数最小法は、文字列をなるべく少ない文節に分割することを意味する。評価値最大法は、対象となる文節間の関係などを評価し、その評価値を最大とするように文節を分割することである。

正しく文節に分解できたとしても辞書引きと文法処理では、一部の場合に同音異義語を区別できない場合がある。これを区別するには、多少なりとも単語の意味に踏み込んだ処理が必要となる。たとえば、「あつい」という単語には「熱い」、「暑い」、「厚い」などがあるが、これらは後続する単語により使い方が決まっている。「暑い夏」ということはあっても「暑い本」にはならないわけである。

こうした処理を俗にAI辞書と呼ぶ。これは、あるワープロの機能名からくる名称なのだが、テレビコマーシャル以来、パソコン用のかな漢字変換システムなどにも取り入れられ、現在では普通に行われている処理である。

この意味に立ち入る同音異義語の処理には、単に組み合わせを辞書に登録する、あるいは単語に意味ラベルなどを持たせて接続可能な組み合わせを制限するなどの方法や、係り受けや格情報などを含めた組み合わせ処理など、さまざまな方法がある。単純に「厚い」と「本」「辞書」「オーバー」といった単語の組み合わせだけを記憶しておいても実用上の組み合わせはさほど多くなく、どのような方法にしても辞書にない単語との組み合わせは判定不可能なので、ある程度の変換効率を達成することは可能である。しかし、「人の良い厚い赤い童話の本の持ち主」といった文章になると、どの語がどれを修飾しているのかなどを判定しなければならない（ただし、すべての場合で判定が必要というわけではなく、組み合わせによっては関係を把握しないと同音異義語が区別できなくなる可能性があるということ）。また、人間でさえその関係を把握できない文章もありえるので、どれだけ精密に分析しても決定が不可能となることもある。

Wnn7では、FI（Flexible Intelligence）変換と呼ばれる機能を持つ。これは、前後の文節の格情報を解析し、文節間の接続状態から、最も使用頻度の高い変換方法（係り受け変換、合成語変換、修飾語変換など7種類）を選択するもので、さらに入力後に確定した文章から文節間の関係を学習することができる。これにより、ユーザーの文章のクセを織り込んだ変換が可能になる。

Wnn7のインストール

Wnn7は、バイナリファイルがCD-ROMで供給される。対応OSは、Linux（Red Hat Linux 7.1、Turbolinux Workstation 6.0、LASER5 Linux 6.4、Vine Linux 2.1.5、Kondara MNU/Linux 2000の各日本語版）、FreeBSD（4.3-Release）である（すべてx86アーキテクチャ用）。

また、オムロンソフトウェアのほかの製品と同じく、ライセンスサーバによるライセンス管理が行われている。なお、同じアーキテクチャとなるため

にWnn7とFreeWnnは共存することができない。すでにFreeWnnがインストールされている場合、インストーラは、FreeWnn（あるいはWnn4）のアンインストールを行う。また、Wnn6に対してはアップグレードを行いWnn6はアンインストールされるものの、ユーザー登録単語などは引き継がれる。

インストール作業はCD-ROMをマウントし、そこからスクリプトを起動することで行われる。日本語での表示も可能だ。ただし、インストール終了後にユーザーによる多少の手作業が必要になる。具体的には、XMODIFIERS環境変数などの指定や、kterm、ネットスケープなどで日本語入力を行うために必要なXリソースの追加修正などである。Wnn7は標準のInput Managerとしてxwnmoを使う。このため、XMODIFIERS環境変数を設定する必要があるのだが、これをどこで指定しているのかはシステムや個人の設定によって異なるため、インストーラでの修正を諦めたものと思われる。

なお、このインストールスクリプトはディストリビューションとそのバージョンを見て処理を行うため、対応ディストリビューション以外では警告が表示され、強行して行うかどうかをユーザーが選択する。筆者の環境では、Kondara MNU/Linux 1.1であったため警告が表示されたが、プログラムをインストールすることは可能であった。

インストール後の手作業の負担を軽くするため、xwnmoの起動など、各ディストリビューションに合わせた設定ファイルなどが用意されている。筆者の環境では、Kondara MNU/Linux 2000の設定を参考にした。Kondaraでは、/etc/X11/xinit/xim.dに各かな漢字変換システムの初期化などを行うファイルを置き、そのどれかを実行することでかな漢字変換システムを切り替えている。しかし、Wnn7のCD-ROMに入っていた設定では、この初期化ファイルの中身に何も記述がなく（ヌルキャラクタしか入っていなかった）、この方法ではxwnmoを起動できなかった。.bashrcなどにXMODIFIERSを指定し、xwnmoを手動で起動することで使えるようにはなるのだが……。

なおxwnmoは、ktermとMule（Emacs）に対応しており、前者は-ximオプションを付けて起動し、後者は専用のwnn7egg（elispスクリプト）を組み込んで利用し、Emacs自体の再コンパイル（xim対応）は必要ない。

Wnn7の附属ツール

Wnn7にはいくつかの附属ツールがある。ここでは、そのうちユーザーインターフェイスを持つユーティリティ類を紹介する。

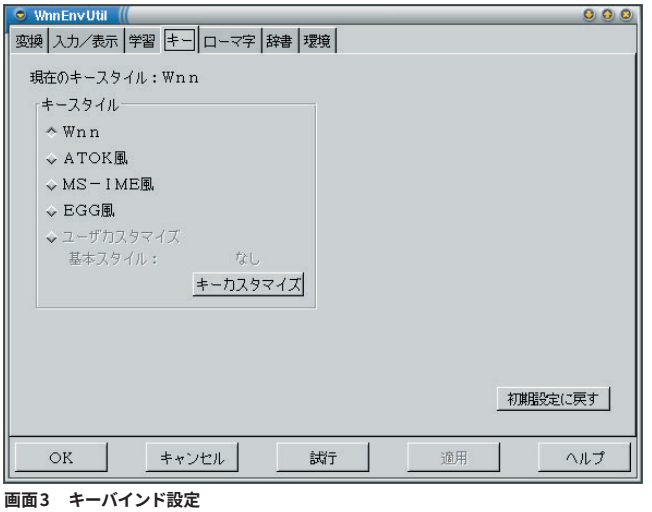
xwnmoはツールパレットを表示し、ここから環境設定ツール（wnnenvutil）および、辞書関連機能の起動が可能だ。

xwnmoのツールパレットは、状態表示（左端）と4つのボタンから構成されている。左端部分には現在の入力文字種やローマ字入力のオン／オフなどが表示される（**画面1**）。その隣のボタンは、ローマ字入力のオン／オフを切り替えるものだ。このほか、ツール類の起動（Wnn7アイコン）、辞書登録（辞書アイコン）、ヘルプ（？マーク）などがある。

環境設定ツールはコマンドラインからも起動可能で、ウィンドウは7つのタ

画面1　xwnmoのツールパレット
xwnmoは起動するとツールパレットを表示する。これがxwnmoが動作している表示となる。

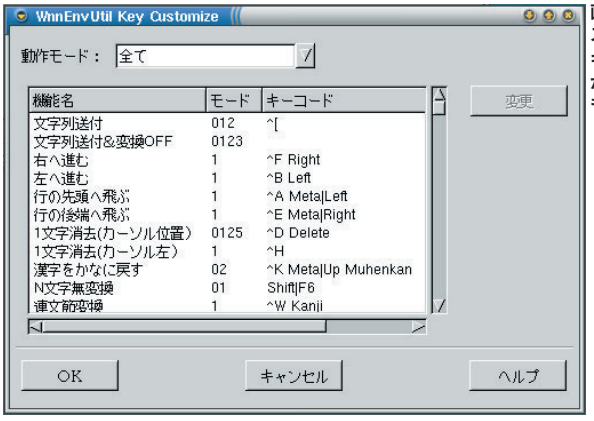
画面2　環境設定ツール
環境設定ツールは7つのタブで項目を切り替えて設定を行える。



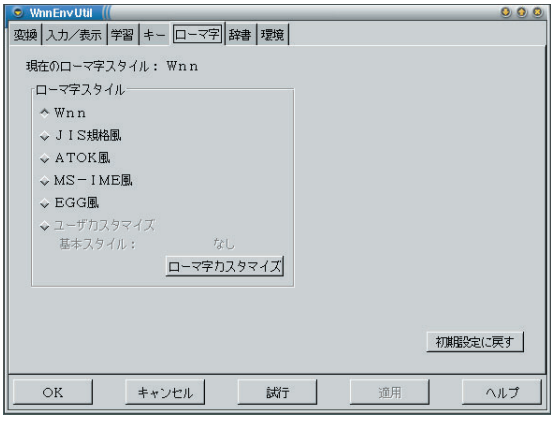
画面3　キーバインド設定
キーバインドは、あらかじめ登録してあるATOKやMicrosoft IME風に設定することが可能だ。

ブに分かれているダイアログボックス形式（**画面2**）になっており、変換に関する設定や学習のオン／オフ、キーバインド、ローマ字変換テーブルの切り替えと設定、辞書設定、その他の設定が可能になっている。なお、キーバインドは、「Wnn」オリジナルの設定以外に「ATOK風」、「MS-IME風」、「EGG風」（**画面3**）と、ユーザー独自指定が選択可能（**画面4**）で、ローマ字入力テーブルについてはさらに、「JIS規格風」が選択可能（**画面5**）だ。

辞書については、xwnmoツールパレットから単語登録や削除をGUIベースで実行可能だが、別途、辞書ユーティリティ（wnndictutil。**画面7**）が附属しており、ここから辞書関連の作業のほとんどが行える。また、ここではATOK（13、14）とMicrosoft IME（97、2000）の辞書ファイルからWnn7辞書への変換、およびWnn7辞書をJIS

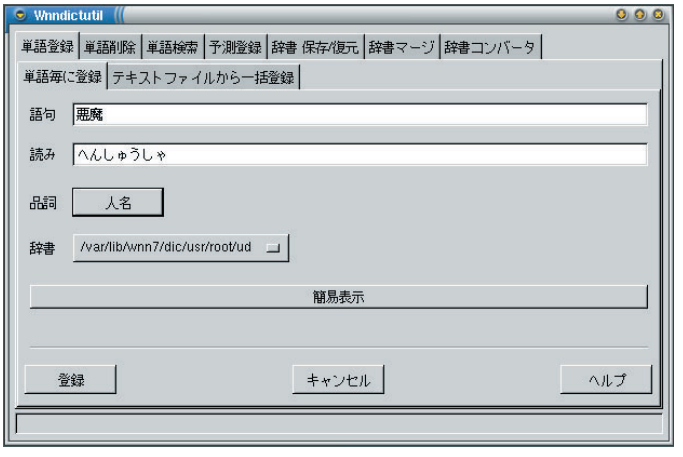


画面4　キーカスタマイズ
キーバインドはユーザーが独自に設定を行うことも可能になっている。

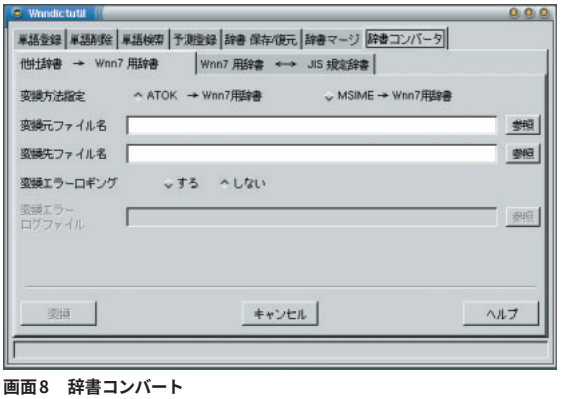


画面5　ローマ字スタイル
ローマ字変換テーブルも、ATOKやMicrosoft IME風のものがあらかじめ定義されている。

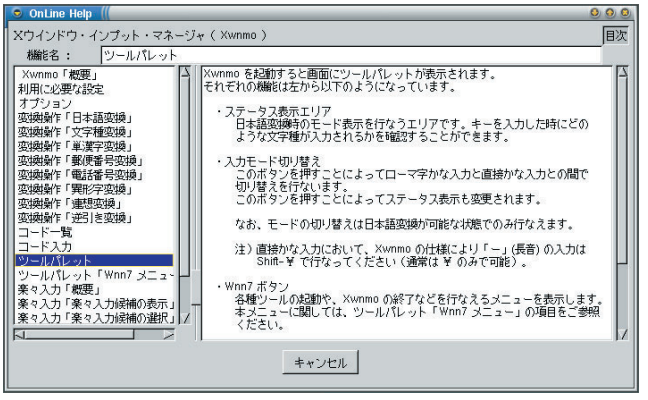
画面6　ローマ字カスタマイズ
ローマ字の変換テーブルをユーザーが自由に設定することもできる。



画面7　辞書ユーティリティ
辞書ユーティリティもタブにより項目を切り替えて利用する。単語登録や削除など、辞書に関連するすべての操作がここから可能だ。



画面8　辞書コンバート
Windows用のATOKやMicrosoft IMEの辞書をWnn7用に変換して利用することもできる。



画面9　ヘルプ
ヘルプには専用のブラウザがあり、リストから項目を選択して読むことができる。

規定辞書への変換が行える（**画面8**）。

ヘルプは、見出し領域と本文領域に分かれた専用のウィンドウに表示される（**画面9**）。また、各ダイアログボックスにあるヘルプボタンを使うことで、該当項目を表示した状態でヘルププログラムが起動する。

Wnn7のGUIと特殊入力

Wnn7はかな漢字変換システムであるため、その操作のほとんどはキーボード経由で行われる。ただし、一部の機能についてはダイアログボックスが表示されて、そこにユーザーが入力などを行うものがある。

キー割付は、前述のように設定の変更が可能であり、今回は標準のままとしWnn7オリジナル割付で評価を行った。以後のキーバインドの表記はこれに基づいたものである。

標準では変換モードのオン／オフは、Ctrl＋\で行う。ktermなどではこの段階では、なにも表示は変化しない。xwnmoの状態表示が「---」から「あr」に変化するので、これで入力状態を判定する（**画面10**）。入力を始めるとローマ字がひらがなに変換されていき、アンダーバー付きで表示される。また、予測入力用辞書にヒットすると、該当の文字列がカーソル下に「予測候補」としてリスト表示される（**画面11**）。このときの最大表示数は10を上限として設定可能である。Ctrl＋Wで変換を行うと、入力文字列の変換が行われる。この段階では先頭文節のみが選択状態となる。ここでさらにCtrl＋Wを押すと、変換候補がある場合、変換候補がリストとして表示される（**画面12**）。

また、ここでShift＋F9で連想変換が行われる。ただし、該当単語について連想語が登録されている場合のみ候補が表示される。

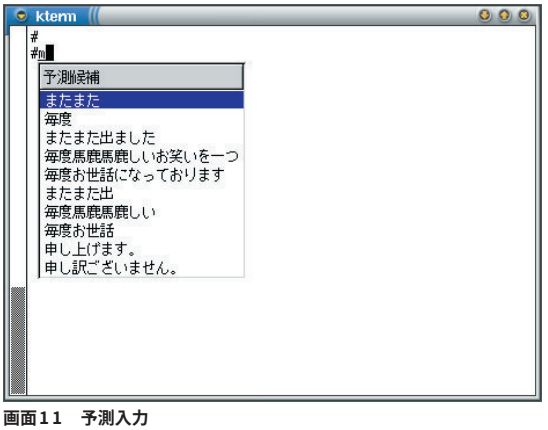
ローマ字入力以外の文字入力方法としては、

- **コード入力（画面13）**
- **コード一覧（画面14）**
- **部首入力（画面15）**

などが用意されている。これらはキー割付されており簡単に呼び出すことが可能だ。ただ、個人的な感覚でいえば、これらの入力はふだんあまり使わないことが多く、従ってどのキーに割り当ててあるのかを覚えられない。ツールパレットから直接指定できるほうが便利だろう。

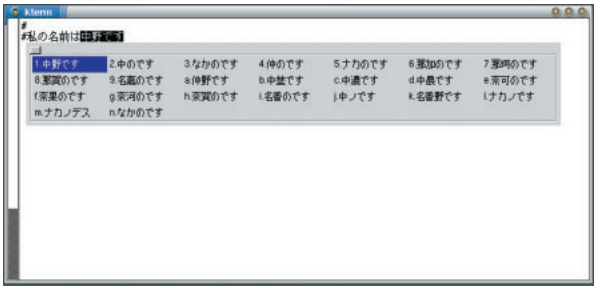
逆変換については、Shift＋F10でウィンドウ（**画面16**）を表示させた後、そこにカット＆ペーストもしくはコード一覧、部首検索などにより逆引きしたい文字列を指定する。これはちょっと面倒だ。というのは、このような難読漢字は入力中のアプリケーション側

画面10　変換オン
キーストロークで変換をオンにすると、xwnmoの左側の領域に現在の入力モードなどが表示されるようになる。



画面11　予測入力
予測入力（楽々入力）は、最初のローマ字を打った段階で同じ文字から始まる候補をリスト表示する。また、一度入力した文字列を文節に分解して記憶するため、最初の文節のみ入力するといった使い方も可能。

画面12　候補の表示
変換後、文節を選択して再度変換キーを押すと、変換候補が下に表示される。



画面13　コード入力
コード入力では、区点コードやJIS、シフトJIS、EUCなどのコードで直接文字の入力が可能。

画面14　テーブル入力
テーブル入力では、文字を表から探して選択することができる。

ではなく、参照しているWebブラウザなどに表示されているわけで、ちょっと使おうと思っても「難読漢字の選択」、「コピー」、「ktermの起動」、「Wnn7のオン」、「Shift＋F10」、「ペースト」といった作業が必要になるからである。できれば、単独のツールとして常に入力ウィンドウを表示するなどしたほうが便利なのではないだろうか。入力作業中にキーストロークで起動可能という点はかまわないが、これしか起動方法がないというのはちょっと疑問を感じる。

Wnn7の使い心地

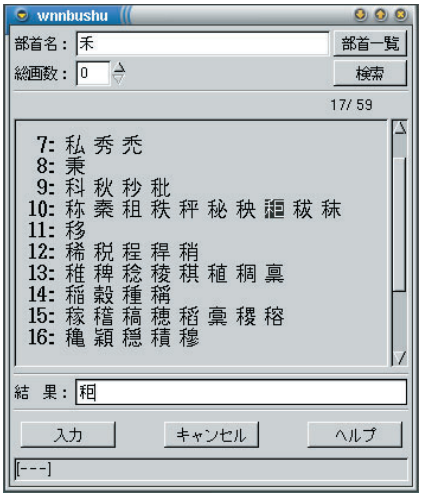
さてこのWnn7の使い心地だが、確かに変換がかなり正しく誤変換が多いという印象はない。十分実用という水準にはあると思われる。ただ、文節の切り直しが必要な場合も多少あるし、同音異義語の選択をすることもないわけではない。短い間の評価なので、辞書学習をかなり行わせればもう少し改善される部分もあるかもしれない。

予測入力は、先頭のローマ字を打った段階でリストが表示されるため、本当に頻繁に入力する文字列、たとえば、ビジネスレターをよく書くユーザーであれば文頭の挨拶などは確かに簡単に入力できる。また、文節に分解されるため、後半の違う文字列なども入力がラクになるのは確かだ。同様の機能は、ATOK14などにも装備されているが、こちらに比べるとWnn7の入力予測のほうがストレートである。ATOK14の予測入力は複雑な動作になっていて、必ずしも思い通りに候補が表示されないことがある。また、常に第一候補だけを表示し、リスト表示させるには別途キー入力が必要な点が違う。

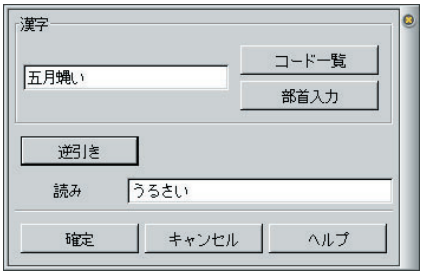
ただ、カーソル位置のすぐ下にリストが表示されるため、ユーザーによってはこれを煩わしいと感じることもあるだろう。これはかなり好みの問題である。筆者も頻繁にウィンドウなどが表示されるのはあまり好きではないし、イベントが連続して処理されつつ、表示がリアルタイムに切り替わっていくような場合に、システムのバグが顕在化しやすく落ちやすくなりそうで、ちょっとした不安を感じる（確証があるわけではないが、スクロールバーをメチャクチャに動かしたりするとハングするソフトに出会ったことは何度もある）。

連想変換は、筆者のように比較的大量に文章を書く場合には便利な機能なのだが、実際に使ったところ、連想変換できない単語に出会う確率が高かったこと、候補リストに抜けを感じたこともあり、辞書をもう少し強化する必要があるのではないかと感じた。たとえば、「不要」という単語に対しては連想変換ができないのに対して、「不用」という単語では連想変換が可能であるが、その候補には「不要」が入っていない。「不用」が「いらないこと、用のないこと」などという意味で古くから使われていた言葉であるのに対して、「不要」は明治期に作られた「必要でないこと。いらないこと」という意味の単語である（単語の意味は広辞苑より引用）。読みが同じではあるが、連想変換でも両方が候補として表示されるべきなのではないだろうか。ほかに例があるかどうかは見つけられなかったが、短い間にひとつでもこうした例を見つけてしまうと、辞書内容にちょっとした不安を感じてしまうのも確かだ。

また、インストール時にFreeWnnがアンインストールされてしまうのもちょっと気になった。Wnn6は同じ会社の製品でWnn7が後継にあたるのだからアンインストールされるのはかまわないが、FreeWnnはベースが同じで同じアーキテクチャを使うとはいえ、いまや別のソフトウェアといえるだろう。実用上、1人のユーザーがFreeWnnとWnn7を使い分けることは少ないので実害はないのかもしれないが、管理主体が別々のソフトウェアをアンインストールしてしまうのもどうかと思う。ポート番号を変えるなどして互いに独立して利用できるようにできないものだろうか。UNIX上のソフトとしての歴史が長いためか、こういったユーザーライクな部分が今ひとつなのかもしれない。今後、改善してほしい点である。



画面15　部首入力
部首入力では最初に部首を選択し、その後、該当部首を持つ漢字を「検索」ボタンで表示させてから漢字を選択する。



画面16　逆変換
逆変換は専用のウィンドウで行う。カット＆ペーストで難読漢字を入力するか、コード、または部首入力で漢字を入力した後、「逆引き」ボタンで読みを表示させる。この逆変換機能はWnn7で復活した機能である。

隠喩としてのコンピュータ

Linuxは老年期のシステムか？

文：豊福 剛
Text : Tsuyoshi Toyofuku
illustration : hmm

人間のライフサイクルに若年期、成熟期、老年期という3つのフェーズがあるように、人間が作るシステムにも3つのフェーズがあるのだろう。人間が作るシステムは、それを作った人間に似るということでもあるだろうし、システムはそれ自体で成長と成熟と老化のサイクルをもつ生き物のようなものといえるのかもしれない。

『UNIXという考え方』（Mike Gancarz著、芳尾桂 監訳、オーム社）は、システムやコンピュータに対するUNIX的アプローチの特徴が9つの定理と10の小定理にまとめられていて、UNIX的思想のエッセンスがわかりやすく説かれている。その定理3では、できるだけ早く試作を作成すること、つまりプロトタイピングの有効性が説かれているのだが、なぜプロトタイピングが有効なのかといえば、それはライフサイクルに対応した3つのシステムがあるから、というわけである。

若年期、成熟期、老年期

第1のシステムは若いシステムである。このシステムを作る人は、追い詰められている。締め切りに追われ、時間がない。切羽詰まった状況に置かれると、人間はいろいろなアイデアをひらめくものでもある。しかし、正しいやり方で作るだけの時間的余裕はなく、重要な箇所だけに集中し、それ以外の細部は後回しにするという、一点突破的なアプローチになる。このシステムは、1人または少数精鋭チームによって作られる。高いビジョンと情熱が共有されていて結束は強い。気分の高揚と達成感、そして疲労困憊。

第1のシステムには無駄がない。最小限のコストで最高の性能を実現する。それは、単純さと速度を最優先する一方で、多機能性と柔軟性を犠牲にしたからだ。このシステムは人々を驚かせる。そして、もしこのシステムに人々の想像力を刺激する面白いコンセプトが含まれていたならば、第2のシステムが生まれる。

第2のシステムは成熟したシステムである。このシステムを作る人は専門家である。第1のシステムのアイデアの成功に魅了され、しかし自分ならもっとうまくやれたはずだと考える。つまり正しいやり方を知っている専門家の登場というわけだ。第2のシステムは、勝ち馬に乗りたいだけの人々も集める。第1のシステムを作るのは、少数精鋭チームであったのに対して、第2のシステムは委員会が設計することになり、公開の場で議論が展開する。しかし、メンバーの価値観や意見の一致を見るのは難しく、そのため、メンバーの貢献をすべて平等に反映することになり、

その結果、象のように巨大で鈍いシステムになってしまう。

第2のシステムには贅肉が付いている。柔軟性、豊富なオプション、拡張性が追求され、単純さと速度が犠牲になる。犠牲になった速度を穴埋めするには、より高速な新しいハードウェアに投資しなければならない。このシステムを世間は鳴り物入りで歓迎する。誇大広告、専門雑誌の発刊、セミナーの開催。第2のシステムは社会現象となる。第2のシステムは王者として君臨する。そして、第2のシステムへの反抗から、第3のシステムが生まれる。

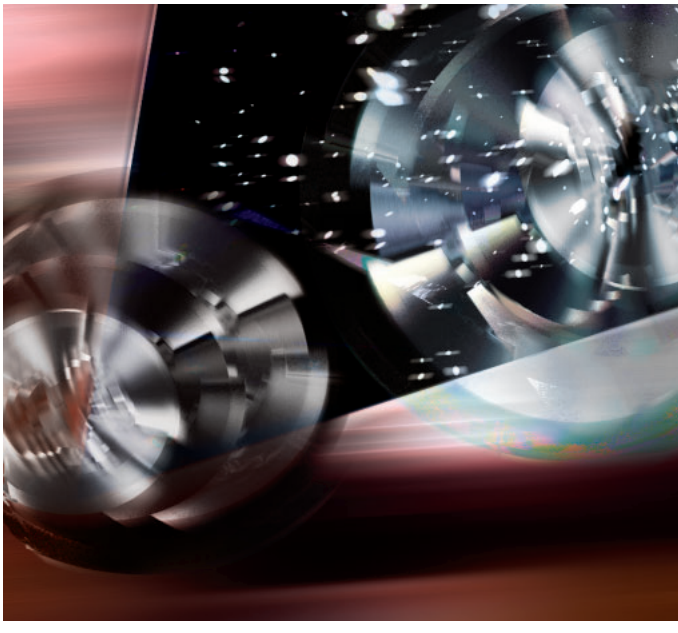
第3のシステムは枯れたシステムである。このシステムを作る人間は、第2のシステムで火傷した人である。第2のシステムは究極のシステムのはずだったのに、時間の経過とともに、それが多くのシステムの中のひとつでしかないことが明らかになる。第2のシステムは、裸の王様だったのだ。火傷した人は、経験の蓄積があり、貴重な知恵を持っている老人だ。しかし、体力は衰え、かつての名声も過去のものとなり、変化を嫌う老人でもある。第1のシステムから第2のシステムへの移行には、希望の翼による高揚感があったのに対して、第2のシステムから第3のシステムへの移行は、沈むタイタニック号から救命ボートに目がけて走るようなものだ。

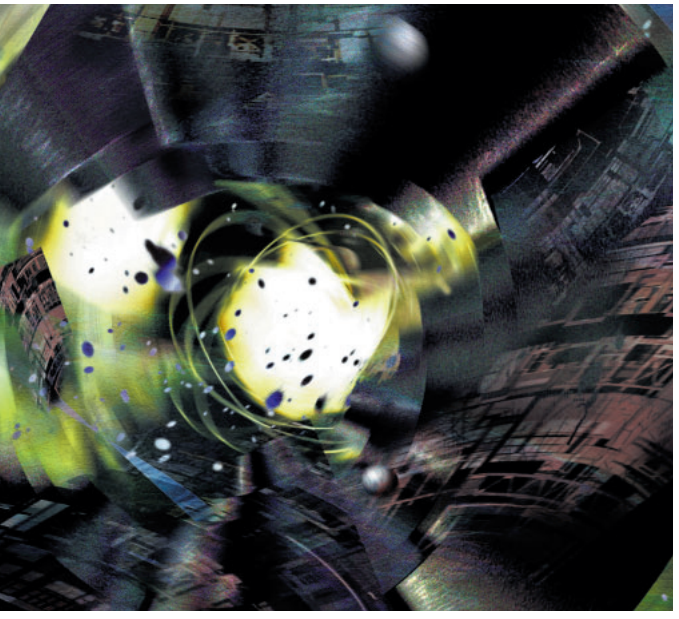
第3のシステムには常識がある。常識とは、その有効性が時間をかけて証明された枯れたコンセプトやアイデアのことだ。第3のシステムは、第1のシステムのもつ高性能と第2のシステムのもつ機能の間に、最適なバランスを模索する。本当に必要な機能だけを残し、それを少ないリソースで実現できるようにする。システムの目的はしっかり把握されている。使用する技術の有効性も証明済みだ。そのため、リスクは最小限である。正確な予算と正確なスケジュールが可能になる。第3のシステムは、ハードウェアに収めたり、ROMに焼くことも可能になる。もはやそのシステムの本質が容易に変化することはない、というわけだ。

Linuxは第3のシステムか？

最終的な目標は、第3のシステムを作ることである。しかし、第3のシステムは、第1のシステムを作り、次に第2のシステムを作ったあとでなければ作れない。それでは、最初の2つのシステムの作成にかかる時間を、できるだけ短いサイクルで実現すれば、それだけ早く第3のシステムに到達できる。そのための有効なアプローチがプロトタイピングである、ということを定理3は説いているのである。

それはさておき、この3段階論は、世の中のシステムを





考えるうえで、なかなか面白い視点を提供していると思う。

たとえば、Linuxは第3のシステムに達していると考えられるが、その一方で、第2のシステムの段階にあるとも考えられる。おそらく、第1のシステムから第3のシステムに至るサイクルは、直線的なものではなく、螺旋状に展開していく、循環的なものなのだろう。あるいは、Linux全体は第2のシステムであるが、その膨大な資産の中から、目的やニーズに合わせて最適のパーツを選択することで、第3のシステムが個別に実現できると解釈してもいいだろう。

そもそも、ケン・トンプソンがPDP-7でスペース・トラベルというゲームプログラムを実行させようと思って誕生したときのUNIXは、第1のシステムそのものであるが、同時に、それはMULTICSという第2のシステムに対する反抗から生まれた第3のシステムでもある。

また、リーナス・トーバルズが386のPCで動くカーネルを作ろうと思って誕生したときのLinuxは、第1のシステムそのものであるが、同時に、それはMinixという第1のシステムよりも、より機能を追加した第2のシステムを実現しようとして誕生したものだともいえる。

資本主義の精神とは

それにしても、なぜプログラマーはプログラムを作り、システムを作るのだろうか。ましてや、フリーソフトウェアやオープンソースとなると、金のためではなく、何か別の価値のために、というわけだが、その別の価値というのは一体何なのだろうか。

『リナックスの革命（原題The Hacker Ethics）』（ペッカ・ヒマネン著、安原和見・山形浩生 訳、河出書房新社）は、このあたりの問題の核心部分を考察した一冊である。興味深いのは、副題の「ハッカー倫理とネット社会の精神」が、マックス・ウェーバーの『プロテスタンティズムの倫理と資本主義の精神』をもじったものである点だ。

西欧近代の資本主義の初期の担い手には、合理的な組織や経営の能力、市民的で自立的な経済活動といった特性が見られる。これは、それ以前の時代における、権力と癒着した政商的な、あるいは、道徳を無視した冒険商人的な経済活動とは、明確に区別することができる。ウェーバーは、初期資本主義の精神の中心に世俗内禁欲というエートス（倫理的生活原則）があり、このエートスは禁欲的プロテスタンティズムに由来すると分析した。

ウェーバーによれば、キリスト教的禁欲の特徴は、設定した目的に対して、精神的にも肉体的にもエネルギーを集

中し、それ以外の事柄については、エネルギーを使わないところにある。そうしたキリスト教的禁欲は、中世には主に修道院で展開していったのだが、この禁欲を修道院という世俗外ではなく、世俗内で展開したのが禁欲的プロテスタンティズムである。

宗教倫理と職業労働観の結合

プロテスタンティズムの思想は、職業における倫理や義務の考え方に大きな影響を与える。ドイツ語のBerufという語は、「呼び寄せる、任命する」という意味の動詞berufenから派生した語であって、呼び寄せ任命する主体は、もちろん神なのである。宗教改革で有名なルターは、聖書をドイツ語に翻訳するとき、このBerufを「職業」の意味で初めて用いたらしい。そこには、世俗内のあらゆる職業はすべて神によって与えられた天職であり、そのため、職業に誠実かつ勤勉に励むことが、神への信仰であるという考え（職業召命観）が込められている。

その一方で、人がどんな身分にあっても救いに到達できるのであれば、職業のあり方を重視するのは無意味である、人は必要を満たすために働けば十分であり、必要を越えた営利の追求は、神の恩恵のもとにはない、とルターは考えた。

プロテスタンティズムにおける宗教倫理と職業労働観の結合は、カルヴァンの宗教思想を受け継いだピューリタニズムにおいて、営利追求を肯定するものに展開していく。そこでは、富を目的とした追求は邪悪の極致とされるのだが、天職である職業労働の結果として富を獲得することは、神の恩恵だと考えたのである。そして、富を所有し享楽することで、怠惰、肉欲、高潔な生活からの乖離がもたらされるのは邪悪であり、そのような時間の浪費は重い罪になる。このような世俗的禁欲は、享楽や奢侈的消費を圧殺する一方で、厳しく絶え間ない労働による営利の追求を、神の意志に添うものとして正当化したのである。

しかし、資本主義の誕生において中核にあった世俗内禁欲のエートスは、資本主義の確立とともに、営利そのものが自己目的であるエートスに変質していく。資本主義の精神は、資本主義が誕生する前には存在していたにも関わらず、資本主義の確立とともに喪失されてしまったというわけだ。

それでは、高度情報化資本主義において、あたかもAIDSのように誕生したハッカー倫理は、労働観と宗教倫理の新たなる結合を意味しているのだろうか。

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。

オープンソースとボランティア

文：安田幸弘
Text : Yukihiro Yasuda

十年ぐらい前なら、「ボランティア」なんて、何か特別な人がすることというイメージがあったかもしれない。しかし、95年の阪神淡路大震災あたりから、日本でのボランティア活動に対するイメージは大きく変わった。

あまり知られていないかもしれないが、今年（2001年）は国連の「ボランティア国際年」なのだ。あの地震でボランティアが大活躍したのを見て感動した日本政府の役人が、国連総会で「2001年をボランティア活動の振興の年にしよう」と提唱して決まった話らしい。そんなわけで、今年は日本国内でもずいぶんいろんなボランティア関連のイベントや活動が開かれ、多くのボランティア・グループがボランティアへの理解の増進や活動への参加を呼びかけている。

だけど、ボランティア精神で国を作ってしまったアメリカなんかは別として、そういう呼びかけが必要というのは、やっぱりまだまだボランティアという活動が、まだ十分社会に根付いていないからなんだろうね。一般的にはやっぱり、「へー、ボランティアなんてやってんの？ 物好きだねえ」みたいな反応が多いような気もする。

確かに物好きなのかもしれない。しかしそれが人間の心理ってやつで、仕事でやれ、と言われたら誰もやりたがらないようなことでも、ボランティアなら面白がってやってしまうのだから面白い。そして、そんな物好きなボランティアたちの無償の貢献が、時としてまさかと思われるようなことをやってしまう。たとえば、世界中のコンピュータを結ぶインターネットだとか、すべてのソースコードが無償で公開される高性能なオペレーティング・システムを作ってしまうとか、ね。

オープンソース・ボランティア

世間的な意味では、インターネットやオープンソースの世界での活動は、いわゆるボランティア活動というイメージではないのだけれど、「ボランティア活動とは××をすること」という定義があるわけじゃない。一般的には、何らかの社会的な活動をするのがボランティア活動だと考えられることが多いのだが、あえて定義するとしたら、自分の意志で自分が興味を感じているコミュニティの活動に参加すれば、それがボランティア活動だ。そこで「何をするか」なんてことは、とりあえず二の次である。ましてそれを世間がボランティア活動と呼ぶかどうかなんてことはどうでもいい話なのかもしれない。

それでも、ぼくは今のオープンソースやフリーソフトウェアを支えるボランティアの活動が、もっと世間的な評価を得られるようにしなきゃいけないんじゃないかと思うのだ。たとえば、いわゆる「ボランティア活動」的な活動、たとえば施設での介護ボランティアだとか、被災地の救援ボランティアなんかだったら、会社によってはボランティア休暇を認めてもらえるかもしれないし、ボランティア保険やら経費の補助が支給されることもある。しかしボランティア休暇の制度がある企業でも、「オープンソースの開発に参加するから」という理由でボランティア休暇を取るのは、相当難しいのではないだろうか。

おそらく、オープンソースの世界でのボランティア活動が、世間的な意味でのボランティアと見なされにくい理由は、オープンソースのコミュニティが現実社会での公共のコミュニティではなく、特殊な私的コミュニティだと考えられているからじゃないかと思う。

しかし、そうなのだろうか。

開発者のコミュニティは特殊なものかもしれない。だけどオープンソース・ソフトウェアそのものが、一部のコミュニティのものではないのは明らかだ。ITの世界で、オープンソースが無視できない比重を占めるに至った現在、オープンソースの開発に携わる開発者は、情報社会の建設の一翼を担うボランティア活動家というべきなのではないだろうか。

オープンソースへの援助は社会に還元される

もちろん、他人の評価なんて気にしてたらボランティアなんてやれないし、支援がなければできないようなボランティアなんてその程度のものかもしれない。だけど世の中のボランティア活動の中には、そうしたさまざまな社会的な理解と支援によって、草の根ボランティアの力だけでは難しい大きなプロジェクトを成功させているケースは少なくない。オープンソースのムーブメントへの参加が、世間並みの意味でのボランティア活動として認められ、社会的なバックアップが得られるようになれば、オープンソース・ソフトウェアの開発は飛躍的に進むはずだ。

現在でも、Linuxをはじめとするオペレーティング・システムやApacheのような大型のアプリケーションの開発は、草の根ボランティアの無償の貢献だけではなく、企業や団体、あるいは公的な助成などからの有形無形の支援に支えられている。また、企業からのコントリビュートも増えてきた。だが、星の数ほど存在するオープンソース・ソフトウェアの数に対して、十分な支援を得ることができるプロジェクトの数はほんの一握りだろう。言うまでもなく、ほとんどのオープンソース・ソフトウェアは、草の根ハッカーたちの無償の貢献によって支えられている。

もっとも、ソフトウェア企業の中には、すでにオープンソースには十分な公的資金が注ぎ込まれてきたと考える企業もあるらしい。確かに、BSDやインターネットの各種のソフトウェアなどは、相当な額の税金が使われているのは事実だ。ただ、そうした公的な資金によって開発されたさまざまなソフトウェアがあるからこそ、現在のソフトウェア産業が存在するという事実を無視するなら、それは傲慢ってもんだ。それに、いくらオープンソースが普及しても商用のクローズドなソフトウェアが消えてなくなるわけではない。もしオープンソースのせいで消えてなくなるようなソフトウェアがあるとしたら、それは商品性や技術の欠如で消えるべくして消えたのだろう。

コミュニティ側も社会的な視点を

企業があり、行政があり、市民が暮らす現実の世界だって、ボランティアだけじゃ動かせない。行政や企業がすべての住民に満足できるサービスを提供できるなら、ボランティアなんていらない。だけど現実には、自治体や企業にできることは限られている。ボランティアの役割は、そうした隙間を埋めてみんながもっと暮しやすく快適な生活を送れるようにすることだ。ボランティアは、行政や私企業を否定するものなんかじゃないし、フリーソフトウェアにしたって商用ソフトの存在を否定するものじゃない。

世の中ってやつは、行政と企業と市民社会の三つのセクターが協調して動いてはじめてうまく動く。特に、現在のように経済が世界化し、私企業の活動が拡大している時代に、市民社会の働き掛け、つまりボランティア活動はいつになく重要になっている。だからこそ、国連もボランティア年なんてものを作ってボランティアに対する理解と支援を呼びかけているのだから。

しかしオープンソースへの参加が現実の社会から世間並みのボランティア活動として認められるようになるには、オープンソースのコミュニティには技術的興味ばかりでなく、もっと社会的な視野が求められるのだろう。オープンソースで社会をもっと良くしようという発想を持つことができれば、と思う。

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

6・21 東京メタリック通信、ソフトバンクの子会社に
6・28 控訴裁判所、Microsoft独禁法訴訟を差し戻し
7・3 アップル、G4 Cubeの打ち切り決定
7・9 Ximian、Monoプロジェクトを発表

筆者のように自宅で仕事をしていると、夏休み時期は憂鬱である。まず、子供が学校にいかないで家にいる。朝早くから近所の公園でラジオ体操している（俺は寝てるんだよその時間）。それに近所の子供が昼間から外で騒ぐ。家族はどこかに連れて行けとうるさいし、どこにいっても混んでいる。毎朝の通勤がないので、電車が空いているなんてなんの関係もない。

それに熱い、暑いではなく、マシンが熱くて、エアコンの効きが悪くなる。昼間外出していると、その間にパソコンがCPU過熱でエラー音を出したまま止まっていることもある。

Microsoftの裁判はどうなった？

さて、昨年地裁で出たMicrosoftの分割命令を含む判決について、控訴裁判所は結論を出した。簡単に言えば、分割命令は破棄されたものの、Microsoftの独禁法（シャーマン法）違反は認めた。また、ブラウザのWindowsへの統合、いわゆる抱き合わせについては地裁へ差し戻し、再度審理されることになった。結局、分割は免れたものの、Microsoftは「クロ」のままなのである。というわけで、この裁判は再度、抱き合わせ販売と是正措置について争われることになったわけだ。

ちなみに、司法省とともにMicrosoftを訴えていた19州のうち、ニューメキシコ州がMicrosoftとの和解に応じ、残りは18州となった。裁判開始時には20州が訴えており、このうち、サウスカロライナ州は早くにMicrosoftと和解している。

ところで、司法省はなるべく早くMicrosoftを和解の席につかせたいようである。というのも、Windows XPの出荷が迫っているからである。米国の法律によると、控訴裁判所の判決が出たあと、被告、原告には、再審請求を検討するための52日間の猶予期間があり、差し戻された裁判はこの猶予期間が終わらないと始めることができない。つまり、8月19日までは何も話が進まないのである。そこで司法省は、審理再開を早めるように控訴裁判所に対して申し入れを行った。今回の控訴審判決では、7人全員の判事が審理にあたり、全員一致で判決を出した。つまり、Microsoftがこの控訴審判決に対して再審理請求を出したとしても、それが却下される可能性が高いわけだ。また、地裁での審理が始まれば、司法省はMicrosoftに対して仮処分申請を行うことが可能になる。

ここで圧力をかけて、Microsoftに和解させ、司法省のコントロール下に置くか、あるいは地裁での再審理中に仮処分を使って、是正措置を行いたいということなのだろう。ところが、Windows XPの出荷に向けて市場が大きく動き出してしまうと、こうした行動に対する制約が大きくなってしまう。

Microsoftが現在計画中の.NETについても、Microsoftが不法に独占状態を維持しようとしているとなれば、さまざまな制限を課す必要が出てくるだろう。そのためにも、Windows XPについても、なんらかの制限を付けたいというのが司法省の考えなのではないだろうか？ さて、このMicrosoftと司法省の裁判、まだまだ続きそうな感じである。その間に、Microsoftの市場支配ってなんだっけ？ ってことになったりしてね。

そういえば、GNOMEの開発元であるXimianがMonoプロジェクトを発表している。これは、公開された.NET情報を使ってオープンソースプロジェクトとして実現するもの。C#やCLR

（Common Language Runtime）、クラスライブラリなどを提供するそうだ。これらを使ってWindows上の.NETアプリケーションも作成可能なのだとか。Microsoftもこういう「危険性」については十分検討したはず。なんらかの対応策を出してくるのではないかと思われる。やはりCorelにLinux版作らせてリリースかな？

現在のところ、LinuxかWindowsのどちらにするのかは、アプリケーションに何を使うのかによって決まってしまうが、こうしたプロジェクトがうまく成果を出せると、アプリケーションが共通になって、純粋にどっちのプラットフォームがいいのか、という話になってくる。すぐにはこうした事態にはならないものの、1〜2年で、また状況が変わってくるのではないかと思う。

そういえば、Microsoftは、まだXPの価格などを発表してないのに、Amazon.comがXPの価格をはやばやと公開して予約を受け付け始めたのだとか。それによるとアップグレード版が100ドルと従来の98からMeへのアップグレードよりも10ドルほど高いのだとか。いままでWindowsはバージョンアップのたびに、古いOSのまま使い続けるユーザーを生んでいるので、また98／Meに居残るユーザーが増えそう。

インテルは次世代メモリに夢中？

インテルは、報道機関向けに次世代メモリの開発動向についての説明会を行った。それによると、大容量保存用メモリには、PFRAM（Polymeric Ferroelectric RAM）やOUM（Ovonics Unified Memory）を中心に研究開発を行っているという。現在、半導体業界では、携帯電話や携帯機器関連のデバイス、特にフラッシュメモリが好調で、各社とも大きな収益源となっている。携帯電話、デジタルカメラなど、さまざまな電子機器にフラッシュメモリが使われているが、特に携帯電話は市場も大きく、その中心。ところが、携帯電話は毎年のように機能が向上していき、IMT-2000などの第三世代携帯電話ではさらに大きなメモリが必要になるという。ところが、現在は大容量を実現するDRAMとフラッシュメモリの間には、大きな性能差があり、その中間に位置するデバイスがない状態。また、DRAMは大容量を実現できるものの、揮発性（電源を切ると内容が失われる）という欠点があり、携帯用機器には向かない部分がある。

いままでメモリといえば、コンピュータを中心に考えてきたわけだが、もはや、携帯電話を中心にメモリデバイスの開発を考える時代になってきたのである。そこで要求されるのが、「不揮発性」で、ほかの電子デバイスと同じチップに集積しやすいメモリなのである。

そんななか、IBMがFeRAM（Ferroelectric RAM：強誘電体RAM）の研究に力をそそぎ、実用化にかなり近づいていることを学会の発表などでアピールした。携帯電話はパソコンと違って、中身に標準化が必要ないため、新しいデバイスを採用しやすい。つまり、先に出荷したほうが、有利になるのである。そこで、各社ともに次世代メモリを大きく意識し始めたわけだ。もちろん、各社ともまったくサボっていたわけではなく、それぞれ研究は進めていたのであるが、IBMがアピールを始めたとたん、各社一斉に、開発状況を発表するなど、「マスコミ」的に騒がしくなってきたのである。今回のインテルの説明会もその一環であろう。

ただ、世間の目に曝されることで、予算や人員が増え、開発が進むという面もある。各社とも4〜5年で実用化と言っているが、今後の動きが加速され、もう少し早くなるような気もする。

アップルはFreeBSD

アップルコンピュータは、FreeBSDプロジェクトのリーダー、ジョーダン・ハバートを雇ったという。Mac OS Xは、簡単にいうとマイクロカーネル（Mach由来）とAPIセットとしてのFreeBSDである。まあ、たしかに雇うと有利な人材。FreeBSDをサポートしていた、BSDIがリアルタイムOSメーカーのウィンドリバーに買収されたって背景もあるんでしょうね。

しかし、Appleって結局自分のことしか考えてないので、今後FreeBSDコミュニティとの関係がどうなるのかね。ちょっと心配。昔、AppleはLinuxに肩入れしようとしたこともあったし、リーナスの自伝によれば、ジョブスがリーナスを誘ったのだとか。でも、リーナスがタネンバウムとマイクロカーネルで論争したことを考えると、彼がMac OS Xを悪く言うのもうなずける。これに対してジョブスは、NEXTのときからマイクロカーネル派だもんねぇ。話が合うはずないよね。

Monoプロジェクト（http://www.go-mono.com/）

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〜引用符の使い方

Linuxのコマンドラインで「文字列」をデータとして渡したい場合、特別な文字がその中に含まれていると思ったように動かないことがある。これを解決するのが「引用符」である。今回はこの「引用符」の使い方のコツを紹介しよう。

文：竹田善太郎
Text : Zentaro Takeda

拙宅の洗濯機は相当古くて、もう12年近くも使い続けているものだ。買った当時は、家電の世界で**ニューロ**や**ファジー**といった、理屈先行の不可思議なコンピュータ制御が流行っていた時代だが、メカの品質は現在のように悪くはなかったので、12年たった今でもガタつくことなく、問題なく使えている。

同様に、10年近く使い続けている電子レンジもそうなのだが、この時代の家電製品は、**何でもコンピュータにおまかせ**というのが流行り始めたころで、**おまかせボタン**のようなもの1つだけを押せば**人間はなにもしなくてよい**、というつくりになっている。ところが、このコンピュータ制御というのが**なんとなくあやしい**のだ。

電子レンジの場合は**調理センサー**なるものが加熱具合を見張っていて、適温になったら自動的に加熱をやめる、ということになっているはずなのだが、たとえば牛乳を温めようとすると、あるときは沸騰してしまったり、逆にあるときは加熱不足で生ぬるい状態になってしまったりする。

洗濯機の場合も、**汚れセンサー**とか**にごりセンサー**とかいったものが、洗濯物の汚れ具合や洗剤のすすぎ具合を監視して……、と**能書き**には書かれているのだが、実際に動いているのを見ていると、それとわかるような動作の変化は、どうもないような気がする。

結局、電子レンジも洗濯機も、**マニュアルモード**でしか使わなくなってしまったのだが、先日、この洗濯機の制御プログラムの**バグ**らしきものを発見した。この洗濯機は、洗濯動作の途中で一時停止ボタンで動作を中断させた後、再び一時停止ボタンで再開させると、中断していたあいだに洗濯槽の水位が減ることを見越して、設定水位まで注水してから洗濯動作を再開するようになっている。おそらく、少なすぎる水位で運転することによる、モーターへの過負荷を避けるためなのだろう。

ところが、洗濯工程が終了して脱水工程に移る前、排水されて水位が半分くらいになった状態で「一時停止」して、洗濯物の偏りを直すなどの手作業をしてから、一時停止状態を解除すると、水を抜いていた最中のはずなのに、なんと、**いったん、元の設定水位まで注水してから、何もせずに再び水を抜き始める**ではないか。**無駄になる水ももったいない**し、なにより、せっかく洗濯物の偏りを直したのに、水位が上がって洗濯物が浮遊しだすと、**同じ作業をやり直さなければならないはめになる**。

おそらくは、脱水工程の直前に限っては、一時停止解除後の「水位維持」の動作をスキップさせるべきところを、**プログラマーが忘れてしまった**のだろう。そう理解してしまえば、致命的な問題でもないし、**笑って済ませられる**のかもしれないが、予期せぬ動作にはちょっとびっくりさせられたし、ほかに致命的なバグがないかどうか心配になってきた。まあ、12年も問題なしに使ってきたのだから、**いまさら**のような気もするが……。

引用符が必要になる場合

前回まで、大量のファイルを「for」コマンドを使って処理する方法について説明してきたが、その話題の中で、**名前にスペース文字が含まれているファイル**の扱い方について

触れた。コマンドラインでこのようなファイル（たとえば「filename with space.txt」というようなファイル）を扱う場合、

```
$ ls filename with space.txt
```

のように、コマンドラインの引数としてファイル名をそのまま記述しても、

```
ls: filename: No such file or directory
ls: name: No such file or directory
        ⋮
```

というように、エラーが大量に発生する。スペースで区切られた文字列中の「単語」ひとつひとつが、別々のファイル名であると解釈されてしまうからだ（**図1**）。

このような場合には、次のように**引用符**（「 」記号）を使えばよいことを説明した。

```
$ ls "filename with space.txt"
filename with space.txt
```

引用符でスペースを含んだファイル名全体をくくることで、lsコマンドに「filename with space.txt」という文字列が、**1つのファイル名**として渡されるようになるのだ（**図2**）。

引用符が必要になるのは、スペース文字が含まれる場合だけではない。シェルのコマンドラインにおいて**特別な意味を持つ文字**を含んだ文字列を、コマンドに渡したい場合には必ず引用符が必要になる。たとえば、コマンドラインにおいて複数のファイル名を指定するときに使う**ワイルドカード文字**、すなわち、「*」（**アスタリスク**）、「？」（**疑問符**）、「.」（**ピリオド**）などの文字を使うような場合がそうである。

たとえば、「Did you get to the weir?」（「ダムに到着した？」というような意味）という文字列をechoコマンドで表示させるような場合を考えてみよう。

```
$ echo Did you get to the weir?
Did you get to the weir?
```

このように、一見するとなんの問題も起こらないので、引用符は必要ないように思えるかもしれない。しかし、

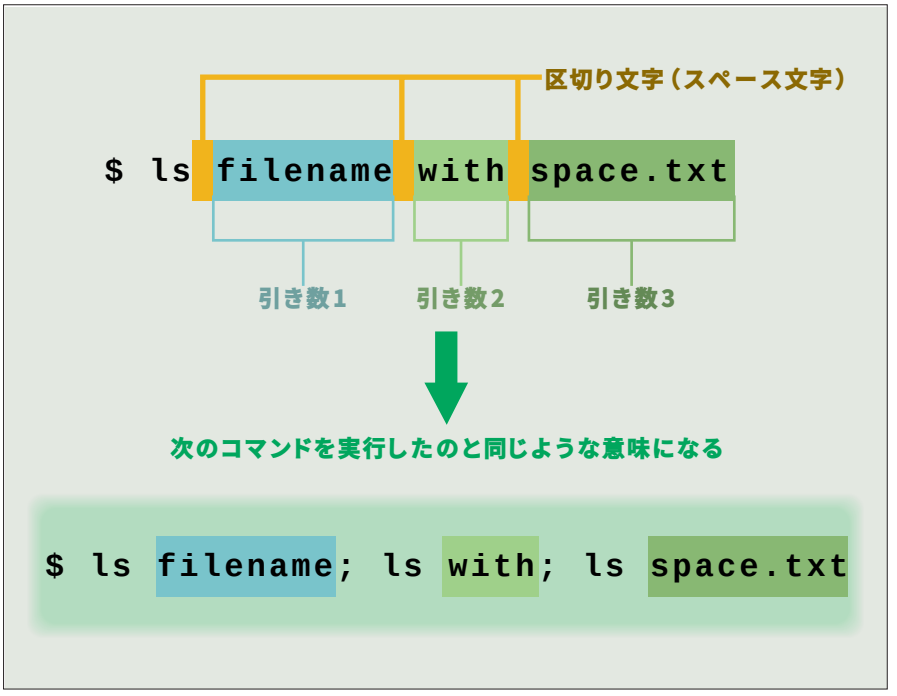


図1　引用符を付けないで、空白文字を含む文字列をコマンドの引数にした場合

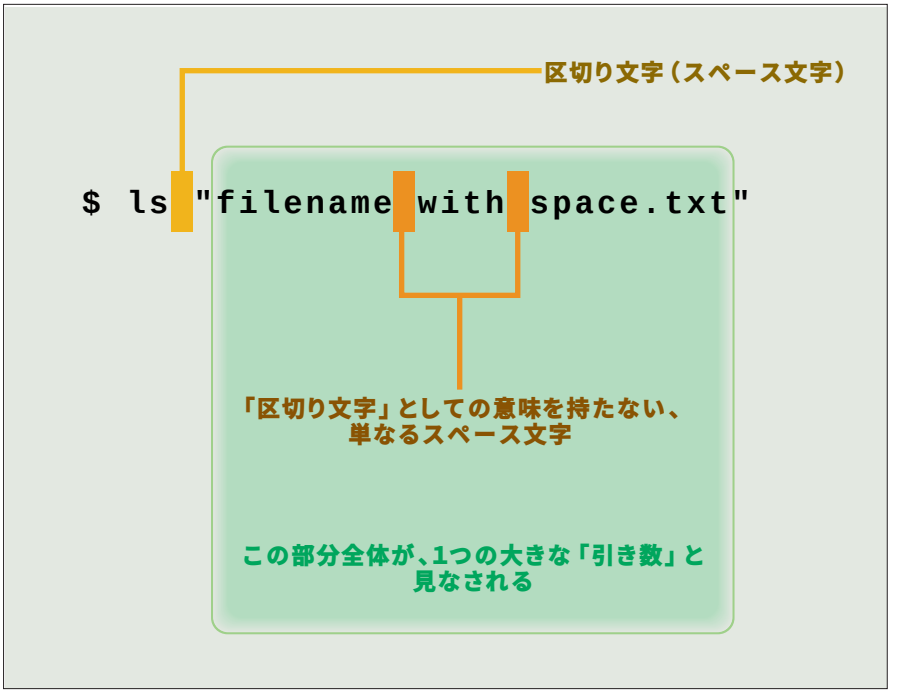


図2　二重引用符で文字列を囲んだ場合

カレントディレクトリに「weird」というファイルが存在した場合、文字列の末尾の「?」が「ワイルドカード文字」であると判断されて、「weir?」が「weird」に変換されてしまう（**図3**）。したがって、

```
$ echo Did you get to the weir?
Did you get to the weird
```

となり、文章が（文法的には正しくないけど）「変人に会えた？」というようなヘンな意味に改ざんされてしまうのだ。

この例はあまりにも極端かもしれないが、コマンドラインでは常に入力した文字列が意図せずに変換される可能性があることを意識していないと、とんでもない結果になってしまう。

このような結果を防ぐために、長い文字列を入力するときは引用符を付けるくせをつけておいたほうがよい。先ほどの例では、

```
$ echo "Did you get to the weir?"
```

のように、文字列全体を「”」で囲んでやれば、文字列中の「？」が勝手に変換されるようなこともない。

1文字だけの「引用符」

ところで、先ほどの「？」のように、文字列中に特殊な文字が1つしかないような場合は、その特殊な文字だけを「引用」扱いにする方法もある。エスケープ文字などと呼ばれる￥（円記号）を使うのだ。この「￥」記号は、日本語以外の環境や、欧文フォントで画面表示しているような場合は、＼（バックスラッシュ）で表示されることがあり、英語版のLinuxやbashの参考書では、すべてバックスラッシュ文字として解説されている。

この「￥」記号を使って先ほどのechoコマンドの例を書き直すと、次のようになる。

```
$ echo Did you get to the weir¥?
Did you get to the weir?
```

カレントディレクトリに「weird」というようなファイルがあったとしても、「？」文字の置換は行われないので、入力したままの文字列がechoコマンドに渡され、表示される。

この「￥」記号は、もちろん「？」以外の特殊文字にも応用できる。たとえば、前回紹介した例では、ファイル名の中にスペース文字が含まれている場合、ファイル名全体を「"…"」で囲む代わりに、「￥」記号を次のように使う方法にも触れた。

```
$ cp This¥ is¥ my¥ favor.txt
my_favor.txt
```

上の例では、ファイル名部分のスペース文字すべての前に「￥」を付けて、そのスペース文字がコマンドライン中の区切り文字ではなく、なんの意味もない単なる文字であることをシェルに伝えている。

もっとも、このような場合、1つ1つのスペース文字の前に「￥」を入力するのは面倒なので、素直に、

```
$ cp "This is my favor.txt"
my_favor.txt
```

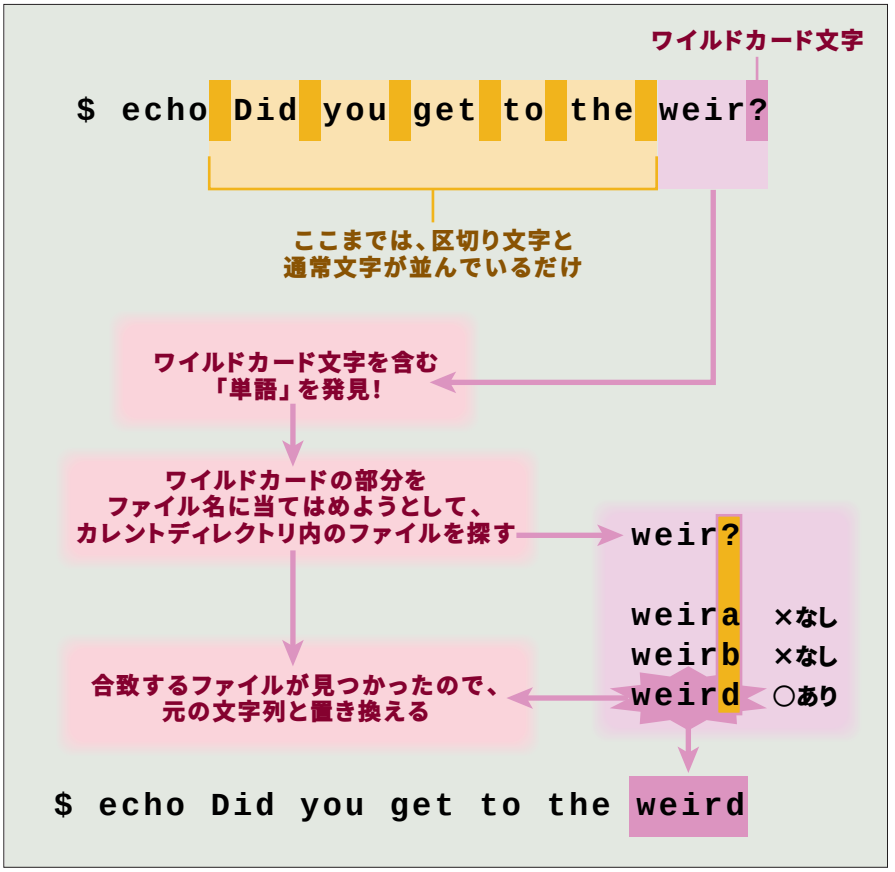


図3　文字列中に「ワイルドカード文字」が含まれている場合
このような「副作用」を防ぐには、二重引用符などで文字列全体を囲めばよい。

のように、引用符で囲んだほうが簡単だろう。

「¥」記号の本当の意味

ところで、今までの説明では「¥」記号は「特殊文字の意味をうち消す」というような意味だと説明してきたが、この文字の本当の意味は**直後の文字を、文字そのものとして扱う**という意味である。したがって、

```
$ echo ¥a¥b¥c
```

というように、ごくふつうの文字「a」、「b」、「c」に¥記号を付けてやっても、それぞれの文字はそのまま文字そのものとして扱われるだけなので、結果は、

```
abc
```

というようになる。

では、¥という文字の前に¥を付ける、すなわち¥を2つ続けて「¥¥」のように入力するととどうなるだろうか。実際に試してみると、次のようになる。

```
$ echo ¥¥
¥
```

この場合、2つ並んだ\のうち、右側の¥は「¥」という文字そのもの、左側の¥は、「右側の文字の特殊な意味をうち消す」という意味になり、結果として「¥」1文字が表示されるわけだ。

さらに、¥文字はほかの引用符として使われている文字を、文字そのものとして扱いたい場合にも使える。たとえば、H"（某PHSのブランド名。「エッヂ」と読むらしい）という名前のファイルをコマンドラインから次のように作成できる。

```
$ touch H¥"
$ ls
H"
```

「"」という文字は「引用符」という特別な意味を持った文字なので、たとえば、

```
$ touch H"
```

と入力しても、

```
>
```

というようなプロンプトが表示されて、目的のファイルは作成できない（このプロンプトは、「引用符が閉じられていないので、続きの文字列を入れろ」という意味）。

もちろん「H"」などというような変な名前のファイルは、あとあとでトラブルの元になる可能性もあるので、**作らないに越したことはない**。だが、なにかの拍子におかしなファイル名をもつファイルが作られてしまうこともないとはいえない。「¥」を使えばこのようなこともできるという例として、記憶の片隅にとどめておいてほしい。

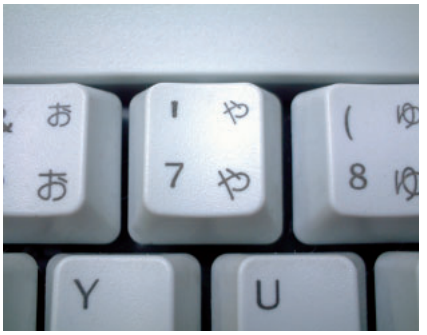


写真1　日本語キーボードでの「'」（一重引用符、シングルクォート）キー

二重引用符と一重引用符

ここまでは、「"…"」と「¥」の2種類の「引用符」について説明したのだが、bashで使える引用符にはまだ別のものもある。たとえば、「'…'」を**二重引用符**あるいは**ダブルクォート**と呼ぶのに対して、**一重引用符**あるいは**シングルクォート**と呼ばれる引用符「'…'」が存在する。

パソコンに慣れていない初心者には、これらの引用符がキーボードのどこにあるかわかりづらいと思うが、一重引用符は、日本語キーボードなら、シフトキーを押しながらメインキーボードの数字の「7」を押せば入力できる（**写真1**）。

英語キーボードの場合なら、「L」キーの2つ右隣のキーがそれである（**写真2**）。こちらの場合は、シフトキーを押さなくても入力できる。

一重引用符は、基本的には二重引用符とまったく同じように使える。

```
$ mv 'long file name.txt' lfn.txt
```

また、一重引用符の中では、「"」という文字は引用符ではなく、単なる文字として扱われるようになる。したがって、先ほどの「H"」というようなファイルも、

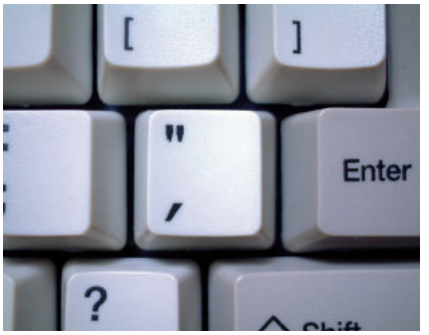


写真2　英語キーボードでの「'」（一重引用符、シングルクォート）キー

```
$ touch 'H"'
$ mv 'H"' edge.txt
```

というように、一重引用符で囲めば、自由に操作できるようになる。

逆に、二重引用符の中では、一重引用符も、ただの文字として扱われるようになる。

```
$ echo "I'm hungry."
I'm hungry.
```

このように、一重引用符と二重引用符は、同じような働きをするものなのだが、実際にはその働きには大きな違いがある。それは、コマンドライン中に含まれている**シェル変数**の扱いの違いである。

シェル変数と引用符の関係

二重引用符の中では、シェル変数を参照して、その値を文字列の一部に組み込むことができる。

```
$ myname="Ichiro"
$ echo "Hello, my name is $myname."
Hello, my name is Ichiro.
```

ところが、上の例の二重引用符を一重引用符に変えてみると、次のような結果になってしまう。

```
$ myname="Ichiro"
$ echo 'Hello, my name is $myname.'
Hello, my name is $myname.
```

このように、シェル変数の部分が置換されずに、変数名が単なる文字列として扱われているのである。

二重引用符と一重引用符では、**文字列中の「¥」の働き**も違う。二重引用符の場合は、引用符で囲まれた文字列中でも、「¥」記号の働きは有効である。つまり、¥の次に来る文字の特別な意味をうち消す。実際には、二重引用符の中で特別な意味を持つ文字は、シェル変数を示す「$」と、「¥」文字くらいなのだが、たとえば次のようにすれば、二重引用符の中でシェル変数が展開されるのを防ぐことができるのだ。

```
$ echo "Hello, my name is ¥$myname."
Hello, my name is $myname.
```

もちろん、このような場合なら、わざわざ二重引用符を使わずに、一重引用符を使えばすむわけだが、たとえば、文字列中でシェル変数を示す「$」と、単なる文字としての「$」を使い分けたいような場合には、二重引用符中での「¥」文字が利用できる。

```
$ total=$((1000+100))
$ echo "Total Charge is ¥$$total."
Total Charge is $1100.
```

ここでは、「ドル通貨記号」としての「$」と、シェル変数「$total」を示す「$」記号の2通りの意味を、1つの文字列中で使い分けている。ちょっと複雑な例だが、自分でいろいろと試してみれば理解できるようになるだろう。

このほかの、二重引用符"…"と一重引用符'…'の違いについては、**表1**にまとめたので参考にしてもらいたい。

「インラインコマンド」

さて、引用符の最後の例は、いままでのものとは**かなり毛色が異なる**。しかし、使い方を覚えれば、コマンドラインの使い道が**数十倍以上**も豊かなものになることは間違いない。

いままでの二重引用符や一重引用符は、引用符の中に文字列を記述して、それをそのままLinuxのコマンドに渡したり、シェル変数の処理をさせるだけのものだった。しかし、bashでは、文字列の内容を**別のコマンドライン**と見なして実行して、その結果として得られる文字列を、元のコマンドライン中の文字列として**再利用**できる仕組みがあるのだ。

このことを、専門用語で**インラインコマンド**と呼ぶこともあるが、用語のことはともかく、実際にどのようなものなのか見てみよう。

```
$ echo 'date'
$ echo `date`
```

上の2つのコマンドラインを実行してみると、結果が大きく違うのがわかるだろう。

```
$ echo 'date'
date

$ echo `date`
Mon Jul 23 20:38:07 JST 2001
```

上のコマンドラインでは、単に

	"…"（二重引用符）	'…'（一重引用符）
シェル変数の扱い	展開する	展開しない
シェル演算子（$((…))）の扱い	計算する	計算しない
`…`（逆引用符、バッククォート）の扱い	囲まれた内部をシェルコマンドとみなして実行する	単なる文字列として扱う

表1　"…"と'…'の主な相違点

「date」という文字列をechoコマンドで表示しただけだが、2番目のコマンドラインでは、なぜか現在の日時が表示される。これは、`（バッククォート）記号で文字列を囲んでいるところに秘密がある。「`」記号は、日本語キーボードでは、シフトキーを押しながら、Pキーの右隣のキー（「@」キー）を押すと入力でき（**写真3**）、英語キーボードの場合は、メインキーボードの数字の1の左隣（**写真4**）を押すと入力できる。

コマンドライン全体を実行する**前**に、「`…`」で囲まれた文字列自体を別のコマンドラインであると見なして実行し、その結果の文字列を元のコマンドラインに**埋め込む**。上の例では、「`date`」の中身の「date」をコマンドラインとして実行する、すなわち、**dateコマンド**を実行して、その実行結果の文字列（Mon Jul 23…）を元の「echo」コマンドの引数として置き換えるのだ。そして、改めてechoコマンドを実行して、結果として、「Mon Jul 23…」という文字列を表示することになる（**図4**）。

前回、MP3ファイルのファイル名を変更する例では、`…`を次のように利用していた。

```
$ for f in `ls -rt *.mp3`
      ⋮
```

ここでは、「ls -rt ＊.mp3」というコマンドラインを実行して、その結果の文字列、すなわち、カレントディレクトリ中にある「＊.mp3」ファイルを、日付の逆順で並べたものを、「for」コマンドの引数として渡していたわけだ。

バッククォートは次のような場面でもよく使われる。

```
$ for f in *.txt
> do
> mv $f `echo $f | sed s/¥.txt/¥.text/`
> done
file1.text
file2.text
      ⋮
```

このコマンドラインでは、forコマンドの3行目で「`…`」を使っている。シェル変数$fに入っているファイル名の中の「.txt」という文字列を、sedコマンドを使って「.text」に置換してmvコマンドに渡し、＊.txtというファイルを＊.textというファイルに名前を変えている。

このように、sedコマンドと`…`引用符を組み合わせる使い方は、ほとんど**公式**のようなものなので、この機会にぜひ覚えておいてほしい。例によって、**付箋紙にメモしてディスプレイの横に貼っておく**とよいだろう。

写真3　日本語キーボードでの「`」（逆引用符、バッククォート）キー

写真4　英語キーボードでの「`」（逆引用符、バッククォート）キー

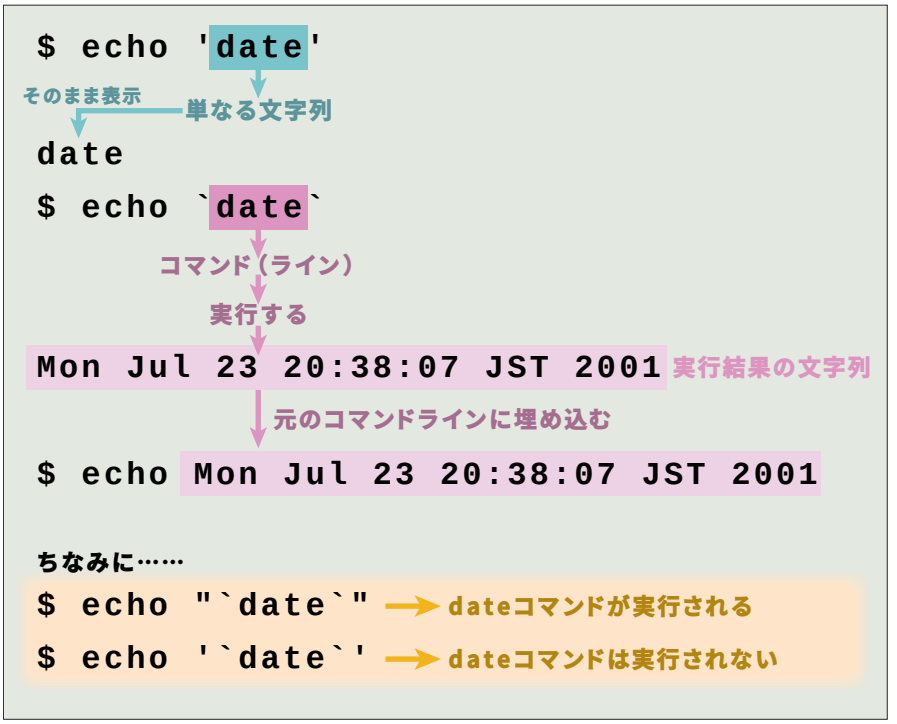


図4　通常の引用符とバッククォート（インラインコマンド）
バッククォートで囲まれた部分は、「コマンド（ライン）」であると判断して、その部分だけを先に実行して、その結果の文字列をコマンドラインに埋め込む。

サーバ百科
基礎から学ぶネットワークの仕組み

バイナリパッケージの普及で、誰でもサーバを動かすことができるようになった。しかし、サーバのメンテナンスやトラブル時の対応には、いくつかの基本的な知識が必要だ。今回は、サーバ運用者に求められる基本的なTCP/IPの知識を解説しよう。

第1回　サーバ運用のための TCP/IPの基礎知識

文：安田幸弘
Text : Yukihiro Yasuda

少し前まで、インターネットのサーバ運用は、コストや技術面での難しさからプロの世界のものだった。しかし、最近になってCATVやADSLを使ったインターネット接続サービスの普及により、個人でも比較的安価に固定IPアドレスの常時接続が可能になりつつある。そして技術面についても、Linuxをはじめとするオープンソースの普及のおかげで、ほとんどサーバ運用の経験がないユーザーでも、とりあえずサーバをインストールして稼動するぐらいのことはできるようになった。

■常時接続でサーバを立てる

自前で運用するサーバは、管理の手間はかかる反面、自由度が高いのが最大の魅力。たとえばこれまでプロバイダが用意しているサーバでWebサイトを構築しようとしても、CGIが使えなかったり必要なモジュールがインストールされていないなどの不便を感じていた人も少なくないだろう。しかし、自分の自由になるマシンでサイトを運用するのなら、そんな制限は一切ない。XMLでもJavaでも、自分が興味のある技術を気兼なく使うことができる。インターネットの技術に興味のある人には、天国のような環境が手に入るのである。

しかし、公共のインターネットに繋がったサーバは、自分だけのものではないということにも十分に注意を払ってほしい。インターネットに接続されたサイトの管理不良によって、何かセキュリティ上の問題が発生したり、トラブルが起きれば、それは自分だけの問題ではなくなる。世界中のIPアドレスは、高性能なポートスキャナで常時スキャンされていると思っておきたい。「まさかウチが……」と思っていると招かれざる客の来訪であわてることになりかねないのである。

LinuxでRPMパッケージを使えば、好みのサーバをインストールし、起動するところまでは誰でも簡単にできる。また、サーバを走らせておくために、細かいTCP/IPの専門知識が必要になることはあまりないと思う。だが、ネットにつながったサーバを運用するからには、たとえ小さなサイトであっても、あなたは単なるインターネットユーザーでもLinuxのオーナーでもなく、サイトの管理責任者だ。そしてサイトの管理者なら、自分のサイトを効率的に運営し、さまざまなトラブルに対処するためにも、基本的なTCP/IPの常識は理解しておきたい。

若葉マークのサイト管理者が知っておきたいTCP/IPの常識について、簡単に全体像を説明しよう。

■インターネットとTCP/IP

よく知られているように、インターネットはTCP/IPと呼ばれるプロトコルで世界中のコンピュータを結ぶネットワークだ。

コンピュータ通信のプロトコルにはさまざまな種類があるが、たとえば電話線とモデムを使ったいわゆるパソコン通信の無手順プロトコルは、基本的に1本の電線で遠く離

れたコンピュータを接続するためのプロトコルだ。原理としては、物理的に電線をたどっていけば相手のコンピュータの場所にたどりつく（もちろん、現在では電話システムがデジタル化されているため、電線をたどっていくことは不可能だが）。このようなネットワークでは、接続先を変えるときは別の電話番号をダイヤルして、電話線を別のコンピュータにつなぎかえてもらうことになる。これが回線交換という考え方だ。

しかし、回線交換では、1本の電話線で一度に1つの相手にしか接続することができない。また、情報のやりとりがない状態でも、電話回線を占有してしまうため、効率が悪く、コストもかかってしまう（**図1**）。

アナログの通信なら、「音が聞こえない」という情報もまた情報だが、デジタル通信では、「情報がない」という情報は必ずしも必要ではない。そこで、デジタル通信の世界では、アナログ情報のように回線を占有するのではなく、複数の情報を1本の回線に同居させるようにするパケット交換という方法が早くから使われてきた。

パケットとは「小包」という意味で、パケット通信とはあるデジタル情報を小包のように適当な単位に分割してやりとりする。回線に空きがあれば、ほかのパケットを送れるので、伝送の効率が上がり、通信コストを低くすることができる。もちろん、情報をばらばらにして送るので、他人の情報とまざってしまわないようにしなければならない。そこで各パケットには送り手と受け手を識別する情報が書き込まれる。また、途中でパケットが壊れてしまったり紛失してしまった場合に備えて、さまざまな管理情報も送られる。このような管理情報がいわゆるパケットのヘッダと呼ばれる情報で、肝心な情報はパケットのボディ部分に書き込まれて送られる。

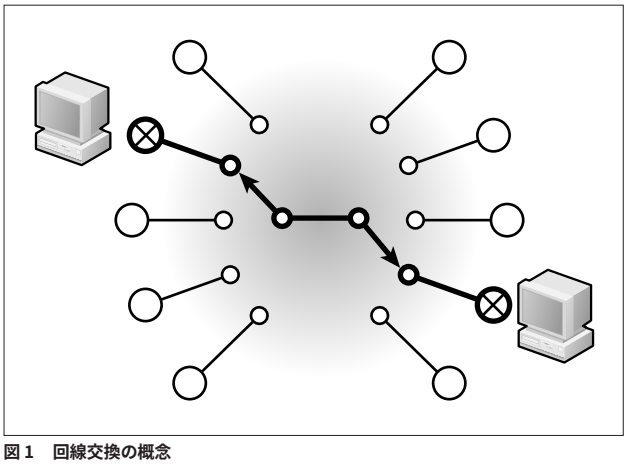


図1　回線交換の概念
回線交換のネットワークでは、1本の回線でコンピュータを接続し、情報がやりとりされる。

回線そのものをつなぎかえて相手を選ぶのではなく、回線を流れるパケットを必要に応じて振り分けることで相手を選ぶ通信方式といえるが、インターネットのTCP/IPをはじめ、現在のデジタル通信はほとんどがこのパケット交換の方法で行われている。

しかし、パケット交換のネットワークにも大きくわけて2種類がある。

ひとつは従来型のオンラインネットワークで使われているコネクション型の通信方法である。これは、基本的な考え方としては前述の回線交換と同じで、コンピュータ制御のパケット交換機は通信が始まったらその通信が終わるまで、その通信状態をがっちりと保持する。そのため、回線のトラブルやパケット交換機の故障によって通信が切れてしまわないように、十分な保守、管理が行われる。

もし無数の回線や交換機のどれかひとつが故障すると、接続が失われてしまうため、回線や交換機の保守管理には相当なコストがかかってしまう。このようなネットワークでは「つながらないかもしれない回線」やら「壊れるかもしれない交換機」などというものは論外で、いくら安くてもそんなものは使えない。

しかし世の中に絶対確実はあり得ない。1960年代、米ソ冷戦の時代に、アメリカの国防総省がソ連の核攻撃によって回線が途絶したり、交換機が破壊されても、可能な限り通信が続けられるようにするネットワークの研究を始めた。それがコネクションレス型の通信方式であるTCP/IP、つまり、「つながらないかもしれない回線」や「壊れるかもしれない交換機」を使っても、何とかつなげられるコンピュータネットワークを誕生させたという。

TCP/IPの最大の特徴はまさに「つながらないかもしれない」、「壊れるかもしれない」という不安定なネットワークでも、ちゃんと通信ができるという点で、必ずしも管理が行き届かない安価な機材を使っても、それなりに通信が成立するようになっている。ただし、いうまでもなく、不安定な機材を前提とする限り、絶対確実ではない。可能な限りさまざまな手段を使って通信の成立に努力するものの、さすがに100％の接続性は保証できない。これを「ベストエフォート」型の通信と呼ぶが、いずれにしてもネットワーク機材に十分な予算を確保することはできなくても安い機材と優秀な頭脳が集まる大学を中心に、「つながれば勝ち」というネットワークの研究が進み、それが現在のインターネットになったわけだ。

TCP/IPは、コネクションレスという言葉からもわかるように、低いレベルでの通信ではコネクションという概念

がない。データと宛先の情報を入れたパケット（データグラム）を作り、ネットワークに放り出すのである。放り出されたパケットは、ルータ（経路制御装置）と呼ばれる機器に送られるのだが、ルータは、互いに情報を交換しあっていて、「何番のネットワークならあっち」という情報を持っている。そこで、ルータはパケットの宛先を見て、パケットを次のルータに送り出す。運が良ければ、送り出されたパケットは次々とルータを経て、最終目的地にたどりつくことになる（**図2**）。

では運悪く、途中でパケットが壊れてしまったりルータが行き先の情報を持っていなかったらどうなるのだろうか。実は、そのパケットは消えてなくなってしまうのである。それで通信が成立するというのかと思われるかもしれないが、TCP/IPはICMP（Internet Control Message Protocol）と呼ばれるプロトコルで、ネットワークの状態を調べ、問題のあるルータを検出したり経路を変更する仕組みを持っている。また、少々壊れたり失われてもかまわないような情報は、処理が単純なUDP（User Datagram Protocol）の形で送られ、破壊や紛失が許されない情報は、送り手のコンピュータと受け手のコンピュータ間でパケットのやりとりに関する情報を交換して、パケットのトラブルを発見できるようにするTCP（Transmission Control Protocol）というプロトコルを使う。TCPは、仮想的なコネクション型のセッションを張ることができ、接続に問題があれば情報の再送信を要求できるようにしている。

以前、TCP/IPの仕様を見たコンピュータネットワークの専門家が「これで本当に通信ができるのか」と驚いたというが、こうした巧妙な仕組みにより、インターネットのTCP/IPは「つながらないかもしれない」安価な回線を使ってもそれなりに通信ができる大衆的なネットワークを実現しているわけだ。

とはいえ、TCP/IPの基本は「ベストエフォート」、つまり可能な限りさまざまな手段を使って相手に接続できるようにするが、場合によっては通信ができないかもしれないのである。

■プロトコルの層

単に安い回線が使える安価なネットワークというだけであれば、必ずしもTCP/IPである必要はないだろう。TCP/IPを基本とするインターネットが世界中に普及したのは、単に低コストというだけではなかった。

当初、大型計算機やUNIXをつなぐところからはじまったTCP/IPは、ミニコンやワークステーション、パソコン、そして最近は各種の情報端末から情報家電まで、さまざまなデジタル機器を同じネットワークに接続することができる汎用性という点も大きな特徴だ。

実は「TCP/IP」と言っても、何か特別な「TCP/IP」と呼ばれるプロトコルがあるのではなく、さまざまなレベルでの通信手順を決めた無数のプロトコルの集合体だ。たとえば、イーサネットでネットワークに接続する場合にはこの手順、接続先のコンピュータにPINGを送るときはこの手順、Webをブラウズするときはこの手順……といったように、いくつもの手順、つまりプロトコルが決められていて、その全体を一般に「TCP/IP」と呼んでいる。

TCP/IPでは、細かいプロトコルまで数えると数百のプロトコルがRFCと呼ばれる文書群にまとめられているが、これらのプロトコルは機種に依存する下位のプロトコルから、実際のインターネットのサービスとして利用者の目に

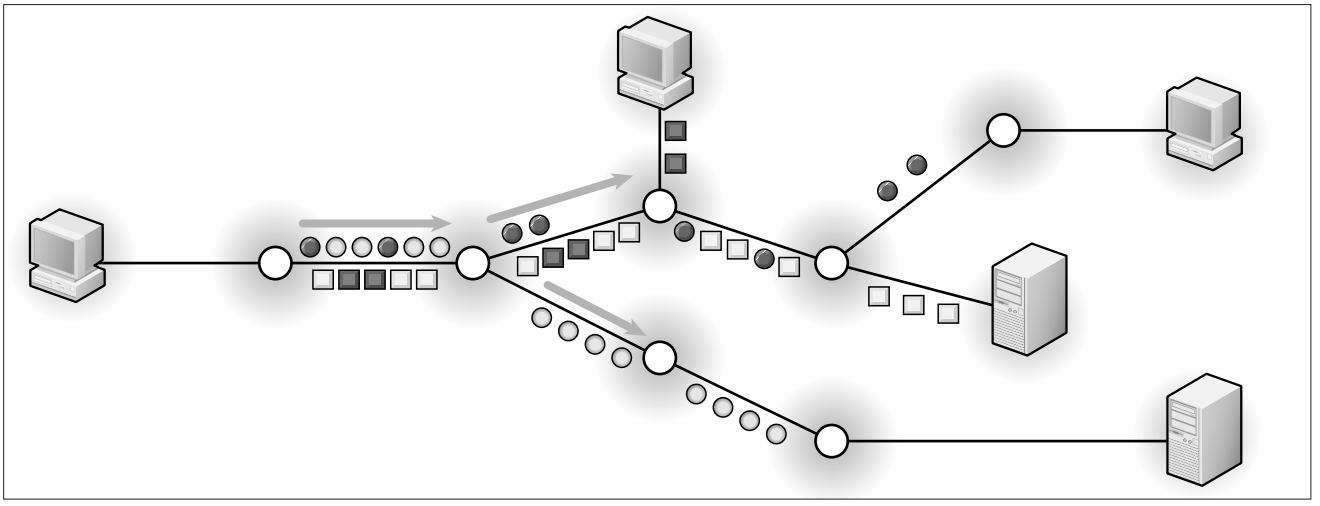


図2　コネクションレスプロトコル
送り出されたデータグラムは、それぞれのルータが最も適当な次のルータに送り出される。

見えるWWWやFTPのような上位のプロトコルまで、いくつかの階層に分けて考えるようにするとわかりやすい。

プロトコルの層分けの枠組みとしては、OSIプロトコルという異機種間接続のためのプロトコルを設計する際に決められた「OSI参照モデル」と呼ばれる7つの層によるモデルが有名で、この話になると必ず**図3**の図が登場することになっている。TCP/IPはOSI以前に設計されたプロトコルなので、必ずしもOSI参照モデルに合致しているわけではないが、基本的な知識としてOSI参照モデルについて知っておくことにしよう。

この図では、右側の層の箱と左側の層の箱が、機種も違い、OSも違うかもしれないが、ネットワークで接続され、相互にTCP/IPで通信ができるコンピュータだ。それぞれの層は、「何をするか」を定義し、それぞれの層でやるべき仕事に関するプロトコルが定義される。そして、それぞれ同じ層では同じプロトコルを実装することで、通信が成立するということになる。

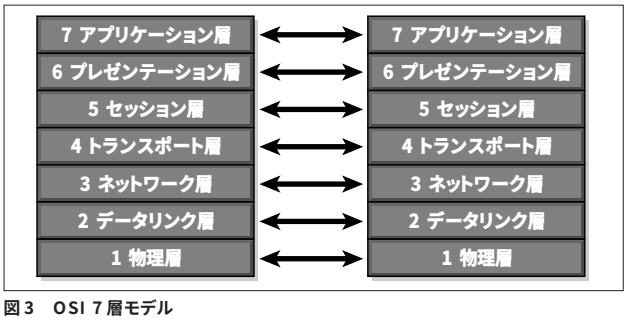


図3　OSI 7層モデル
OSIではネットワークの7つの層を規定している。

たとえば、物理層はコンピュータをつなぐケーブルの物理的な条件を整える役割を受け持っている。おなじみの10BASE-Tの規格は、この層で決められるわけだが、2台の機器は同じ10BASE-Tのケーブルでつなげば少なくとも物理層の仕事、つまり電気的に接続するという点ではちゃんと接続ができることになる。

またそれぞれの層は、隣り合う層との間で仕事の結果の受け渡しをする。たとえば、第2層のデータリンク層はデジタル信号的につなげる仕事を受け持っているのだが、これは第1層が電気的な接続という仕事をこなしているのが前提だ。逆に言えば、第2層では電気的な接続の心配はせずに、第1層が行う仕事を前提に、たとえばチェックサムやCRCなどでデジタル信号としてのエラーを見つけて、必要なら再送信の要求を出すなどして、デジタル信号のレベルできちんと相手とつなげることに専念すればよく、第1層は単に電気信号の部分の仕事に専念すればいいということになる。

このようにして、OSI参照モデルでは7つの仕事を決め

Column

プロトコルと層

コンピュータのネットでは、通信の手順に関する規格をプロトコルと呼ぶ。プロトコルには、複数のコンピュータが間違いなく接続、情報を交換して通信を終了するまでに必要な作業の手順が細かく決められている。

たとえば、どこかに電話をかける場合は、次のような手順になる。

0. 受話器からの発信音で、電話がつながっていることを確認する
1. 電話番号をダイヤルする
2. 接続できたら話をする
3. 接続できなかったら9.に進む
4. 電話が接続したら、自分の名前を名乗り、相手を確認する
5. 間違い電話だったらあやまって9.に進む
6. 用件を話す
7. 電話を切る旨を伝える
8. 相手からの確認があれば電話を切る
9. 電話を切る

もちろん電話の場合、このような手順が文書で決められているわけではないし、電話の相手によってはいきなり「あ、オレだけどさ、あのね……」と話しはじめることもあるのだが、基本的には上記のような手順があり、これがいわば電話の「プロトコル」に相当するものといえる。

ところで、このプロトコルにも実は「層」がある。電話をかける人にとっては、電話回線を接続するという層と、電話がつながってから用件を伝えるという層があり、さらに電話機のレベルでは、電話線が繋がっているかとか電話局との間で回線がつながっているかという層があるわけだが、電話をかける人としては、電話が故障していなければそれらの層については気にする必要はない。

上の「プロトコル」では、0.は電話機と電話局の間の接続に関する層、1.から3.までと9.が電話機の操作という層のプロトコル、4.から8.までが用件の伝達という層のプロトコルと言うことができるだろう。

ところでインターネットの通信でプロトコルというと、ふつうはHTTPやFTP、telnet、SMTPといった上位のアプリケーション層レベルでのプロトコルを指すことが多い。しかし、実際には電話の例のようにいくつかの層があり、おおざっぱにいえば一番低レベルのIPやARP、ICMPなど、ネットワーク層のプロトコル群の上にTCPやUDPといった接続に関するトランスポート層のプロトコル、そしてユーザーがインターネットを利用するうえで直接関係するアプリケーション層のプロトコルが乗るという形になる。利用者がふだん、IPやTCPなどを意識せずにHTTPやFTPを使えるのは、このようにプロトコル群が階層化されているからだ。

OSI参照モデルの層	各層の機能	実装例
7 アプリケーション層	アプリケーションごとの機能	HTTP、FTP、SMTP、等
6 プレゼンテーション層	データ形式の変換	HTML、MIME、等
5 セッション層	通信の確立、切断	TCP
4 トランスポート層	データの転送	TCP、UDP
3 ネットワーク層	ネットワーク的な接続	IP、ICMP、ARP
2 データリンク層	デジタルデータの受け渡し	Ethernet、PPP
1 物理層	物理的な接続	電話回線、ISDN、ADSL

表1　OSI参照モデルの層

て、それぞれの役割を分離している。TCP/IPの場合、OSIの7層モデルにきれいに当てはめることはできないのだが、基本的には7層モデルに対応させて考えることもできる。参考までに、OSI参照モデルでの各層の仕事の内容と、それに対応するインターネットのプロトコルを**表1**にまとめる。

■IPとルーティング

前に説明した通り、インターネットではパケットがルータを経由して相手に届けられるのだが、この部分をもう少し詳しく説明しよう。

前では単に「パケット」と呼んだものを厳密に言えば、IPプロトコルで決められたIPデータグラムということになる。ここで、IP（Internet Protocol）はプロトコルの名前で、データグラムとはひとまとまりのデータである。インターネットに送り出されるデータは、まずIPデータグラムの形にまとめられる。TCPやUDPのようなプロトコルのパケットは、このIPデータグラムのデータとして埋め込まれるわけだ。

IPデータグラムは**図4**、**表2**に示すようなヘッダを持っている。紙幅の関係でヘッダ部分のすべての情報について説明することはできないが、重要な部分は送信先IPアドレスと送信元IPアドレスで、いわば小包の宛先と送り主が書かれた伝票である。この情報を参照して、ルータはパケットを適切な方向に送り出すのだが、これをルーティングと呼び、インターネットを特徴づける技術だ。

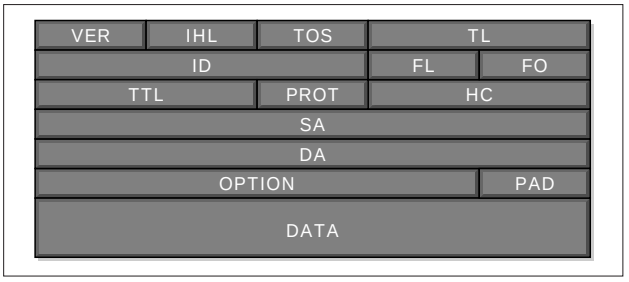


図4　IPデータグラム

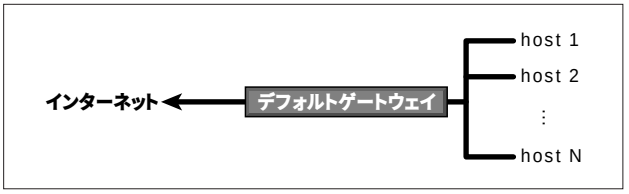


図5　インターネットに接続された末端ネットワークでのルーティング

ルーティングは、専用のルータだけで行われているのではない。インターネットに接続するコンピュータは、すべて何かの形でルーティングを行っており、たとえばLinuxの場合、netstat -r というコマンドでカーネルが管理しているルーティングテーブルを見ることができる。

ルーティングテーブルとは、どのネットワークに向けられたパケットが、どのインターフェイスに送り出されるかという情報を保持しているテーブルだ。小規模なネットワークなら、自分のネットワーク以外はデフォルトゲートウェイに送られ、ローカルなプライベートネットワークがある場合には、プライベートネットワークが接続されているインターフェイスにパケットが送り出される。また、直接イーサネットなどでつながっているホストにはゲートウェ

記号	意味
VER	バージョン。この図はIPv4のヘッダなので、常に4
IHL	ヘッダ長
TOS	サービスタイプ。このパケットの優先度、スループットなどを指示する
TL	パケット長
ID	パケット識別子。上位の層でパケットを再構成する場合などに利用
FL	分割に関するフラグ。配送の途中でパケットを分割できるかどうか、分割されたパケットである場合、最後のパケットかどうかを示す
FO	分割されたパケットのオフセット。ばらばらに到着したパケットを並べ替えるために使う
TTL	生存時間。ルータを通過するたびに減少し、0になるとパケットが破棄される
PROT	プロトコル。この値が1ならICMP、6ならTCP、17ならUDPのパケットであることを示す
HC	ヘッダのチェックサム
SA	送信元IPアドレス
DA	送信先IPアドレス
OPTION	セキュリティラベル、ソースルート、ルートレコード、タイムスタンプなどのオプション機能
PAD	パディング
DATA	データ部分

表2　IPデータグラムの内容

イを経由せずに直接接続するようになっている（**図5**）。

ルーティングテーブルの情報は、マシンをインターネットに接続するときに必要な設定項目をもとに設定される。具体的には、ゲートウェイのIPアドレス、自分のIPアドレス、そしてネットマスクの3種類の情報だ。

つまり、自分のマシンが直接つながっているLANについては、自分のIPアドレスとネットマスクからIPアドレスの範囲が計算できるので、この範囲のIPアドレスには直接接続することになる。また、それ以外のIPアドレスはすべてデフォルトゲートウェイ（通常は回線につながっているルータ）に送られる。

インターネットは、ユーザーが加入するプロバイダ、プロバイダ同士をつなぐバックボーン、そしてバックボーン同士の接続、という階層構造を持っているが、バックボーンに接続されているルータも、基本的な役割はユーザーのマシンの内部で行っていることと同様である。この階層の間でのルーティングを**図6**の模式図で説明しよう。

プロバイダAがユーザーAから送られたパケットを受け取ると、そのパケットの宛先を見て、これが自分の別の顧客宛なら、それをユーザーBに送る。そうでなければ、バックボーン1に送る。次にバックボーン1は、プロバイダAからのパケットの宛先を見て、自分のバックボーンにつながっているプロバイダ宛ならプロバイダB、プロバイダC……へと送る。そうでなければほかの適切なバックボーンに送り出す。

ここで、インターネットがうまく動くかどうかは、多数の宛先ネットワークを管理しなければならないプロバイダやバックボーンのルータが、きちんとルーティングテーブルを管理し、適切なネットワークにパケットを送り出すことができるかどうかにかかっている。手作業で無数のネットワークのルーティング情報を設定することは不可能であり、また、ネットワークの構成が変更された場合の対応が困難になってしまう。そこで、それぞれのルータは、隣のルータとの間でルーティング情報を交換し、自動的に最新のルーティングテーブルを維持するようになっている。

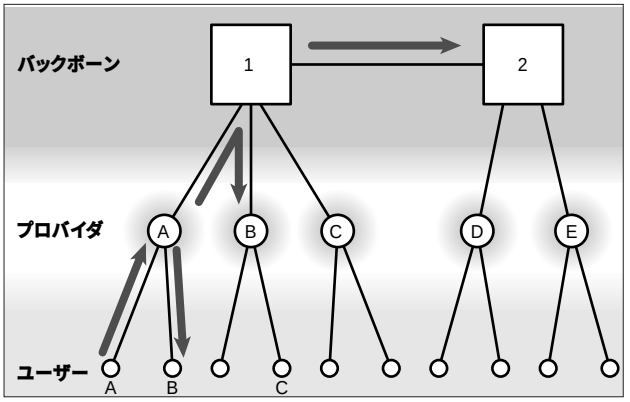


図6　インターネットでのルーティング

ちなみに、小規模なネットワークなら、このようなルーティング情報の交換はRIPと呼ばれるプロトコルで行われ、複数のバックボーンに接続するプロバイダやバックボーンのレベルでは、BGPなどのプロトコルでIPアドレスをまとめて数値化したAS番号と呼ばれる情報を使い、パケットの宛先を制御している。

■ICMP

インターネットの通信を可能にするために重要な役割を果たすのが、ICMP（Internet Control Message Protocol）と呼ばれる特殊なパケットだ。

たとえば、相手のホストが本当に動いているのかどうかは、インターネットでは保証されない。相手のホストが動作していることを確認するためのツールにPINGがあるが、これはICMPを使ったネットワークの管理ツールのひとつだ。また、途中のルーティングの動作に疑問があるという場合に、どのルータに異常があるのかを調べるtracerouteも、ICMPを使っている。

しかし、ICMPの主な使い方は、ネットワークの障害検出と修復で、どこかの機器に問題があることを見つけた機器から問題の通知や障害の回避に関する情報を伝えるためにICMPパケットが送られる。

ICMPは、IPデータグラムとして送られ、ICMPタイプとICMPコードを使って問題の原因やルーティングの変更を伝えることができるようになっている。ICMPのタイプの主なものには、ホストへの到達が不可能（Destination Unreachable）、ルータの生存時間超過（Time Exceeded）、ルーティング変更（Redirect）、応答確認（Echo）などがある。PINGはEcho、tracerouteはTime Exceededを使っているが、たとえばルータのルーティングテーブルが異常でパケットのループが発生した場合などは、Time Exceededを検出したルータがICMPにTime Exceededを入れて送信元ホストに送ることで、送信元ホストはネットワークに障害が発生していることを知ることができる。また、相手のネットワークに目的のホストが存在しない場合は、相手側のルータがホストの検出に失敗したことを知らせるためにDestination UnreachableをICMPで知らせる。

このように、つながらないかもしれない回線、動いていないかもしれないホストを前提とするTCP/IPネットワー

クで、ICMPが問題の発見に果たす役割は大きい。

■TCP、UDPとポート

送信元から送り出されたパケットは、さまざまなルータを通って相手のホストマシンに届けられる。もちろんルータは、実際には単にIPパケットの受け渡しだけではなく、もう少し複雑な処理を行っている。これがなかなか巧妙で面白い部分なのだが、それだけにやっかいな部分だ。これ以上のルーティングについての解説は別の機会に譲ることにして、ここでは無事、相手のホストにパケットが届いたものとして、パケットを受け取った側のコンピュータがそれをどのように処理するかを見ていくことにする。

ルータのレベルでは、すべての情報はIPデータグラムと呼ばれる情報単位にまとめられて転送されていたが、目的のホストに届いたIPデータグラムは、データ部分に収められたTCPやUDPの情報が取り出されて、TCPやUDPを処理する上位の層に送られる。

インターネットのサーバがクライアントとの間でやりとりする情報は、TCPとUDPのどちらかの形で送受信されるのだ。両者の違いは、UDPが単に情報を送出するだけのもので、チェックサムによりデータが壊れていないことを確認できる以外は、データの不着や一連のデータの到着順などについての管理を行わないのに対して、TCPは通信（セッション）の管理を行い、やりとりしたデータが壊れていないか、相手にきちんと伝わっているかを確認し、確実な通信ができる。

UDPは複雑な処理をせずにひたすら送り、受け手のマシンは単にそれを受け取るだけなので、シンプルで処理が高速だ。したがって、多少情報が欠落してもあまり大きな問題がなく、高速な処理を必要とするアプリケーション、たとえばオーディオやビデオのストリーミングなどに適している。このほかには、ネームサーバやネットワークタイムサーバなどがUDPを使っている。

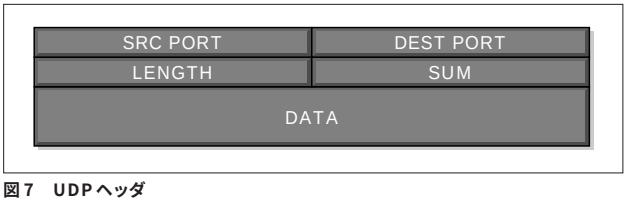


図7　UDPヘッダ

記号	意味
SRC PORT	送信元ポート
DEST PORT	送信先ポート
LENGTH	パケット長
SUM	チェックサム

表3　UDPヘッダの内容

それに対してTCPは、通信の開始と終了を管理し、バラバラに送られてくる可能性のあるパケットを一連の番号順に並べ替えて、不足しているパケットは再送信を要求し、ネットワークが混雑していればデータのやりとりをコントロールするなど、UDPに比べて多くの処理が必要になる。それでも通信の信頼性が必要とされるHTTPやtelnet、FTP、SMTPをはじめ、おなじみのアプリケーションの多くがTCPを使っている。

さて、ここからがいよいよクライアント／サーバ通信の説明らしくなってくるのだが、UDPやTCPのパケットを受け取ったホストコンピュータは、パケットのヘッダに書き込まれている情報を読み取り、適切なアプリケーションに情報を渡し、通信が開始されることになる。ここで必要になるのが「ポート」の情報だ。

TCP、UDPのどちらも、通信を行うためには必ずポートの指定が必要になる。通常、インターネットでは、たとえばポート21はFTP、23はtelnet、25はSMTP、そして80がHTTPと決まっているが、必ずしもプロトコルとポートの関係は絶対ではない。たとえば、telnetをポート80に割り当てて、HTTPをポート23に割り当てることも可能だ。1つのポートを複数のサービスに割り当てることはできないが、TCPとUDPではポート番号が重なってもか

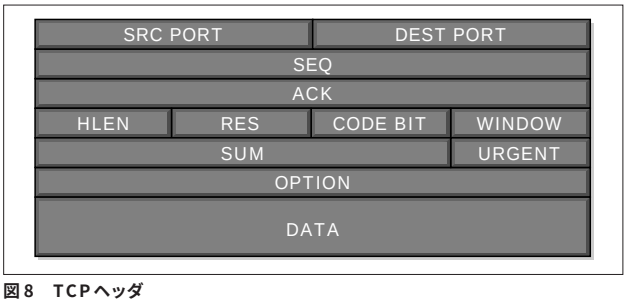


図8　TCPヘッダ

記号	意味
SRC PORT	送信元ポート
DEST PORT	送信先ポート
SEQ	シーケンス番号。データの位置を示す
ACK	応答確認番号。パケットの再送要求やフロー制御に利用
HLEN	パケット長
RES	未使用
CODE BIT	セッションの開始や切断に使う
WINDOW	転送効率を上げるためのウィンドウサイズを指定
SUM	チェックサム
URGENT	緊急ポインタ。緊急な処理が必要なデータへのポインタ
OPTION	オプション機能

表4　TCPヘッダの内容

まわない。つまり、TCPのポート80とUDPのポート80に、それぞれTCPを使うサーバとUDPを使うサーバという2種類のサーバを割り当てることができる（ただし、一般的にはこのような使い方は推奨されない）。

具体的には、接続のプロセスは次のようになる。

1. **まずサーバは起動した後、自分に割り当てられた特定のポートを常時監視する。たとえばHTTPサーバなら、TCPのポート80を監視し続ける。**

2. **このサーバにアクセスしようとするクライアントは目的のサーバのポートを指定してTCPまたはUDPで接続する。このとき、クライアントは返信先のポートをパケットに入れて送る。**

3. **ポートを監視していたサーバは、クライアントからの要求を受け取り、クライアントが指定したポートに返事を送る。**

4. **通信成立**

ここで重要な点は、クライアントがサーバに接続の要求を出すときに、サーバからの返事を受け取るポートを指定すること。通常、現在クライアントが使用していないポートを予約して送ることになる。

また、通信が始まっても、クライアントがサーバに情報を送り込むポートの番号は変わらない。これでは、別のクライアントが同じサーバ側のポートにアクセスしたときに、パケットが入り交じってしまうように思われるかもしれないが、サーバ側ではクライアントのIPアドレスとポート番号などを手がかりにして複数のクライアントからのパケットを区別することで、パケットの混乱を防いでいる。

なお、基本的にクライアント側からサーバが監視しているポートを知る方法はないため、標準的なアプリケーションがデフォルトで使用するポートは固定され、IANAが管理している。そのため、HTTPならTCPのポート80、DNSならUDPのポート53といったように、クライアント側から見ると「ポート＝サービス」になっている。標準的なポートとプロトコルの対応は、/etc/servicesに書き込まれているが、参考までに、ウェルノウンポートと呼ばれる有名なポートの一覧を**表5**に示す。

■アプリケーションのプロトコル

トランスポート層のレベルで通信が成立しても、それだけではサーバにどのようなコマンドを送ればいいのか、サーバから返ってきたデータをどのように解釈すればいいのかがわからず、意味のある通信ができない。

そこで、一番上位のアプリケーション層のプロトコルとして、WWWのためのHTTP、ファイル転送のためのFTP、電子メールのためのSMTPなど、サービスの種類に応じたプロトコルが必要になる。各プロトコルごとに、

TCP		UDP	
ポート	用途	ポート	用途
20	ftp-data	43	whois
21	ftp	53	DNS
22	ssh	67/68	bootp
23	telnet	69	tftp
25	SMTP	111	Sun RPC
53	DNS	123	ntp
79	finger	161	SNMP
80	HTTP	514	syslog
110	POP3		
111	Sun RPC		
119	NNTP		
143	IMAP4		
443	HTTPS		

注：0から1023までがいわゆるwell known port。一般にこのポートを利用するにはシステム特権が必要。1024から49151までは予約ポートで、IANAが管理する。49152から65535までは動的に割り当てられるポートとして使用。

表5　代表的なウェルノウンポート
http://www.iana.org/assignments/port-numbers

Column

ハッカーのアーミーナイフ ― netcat

本文で説明したように、telnetを使ってTCPポートに接続すれば、サーバをコントロールすることができるが、telnetでは任意のポートの監視ができない、UDPは使えないなどの制限がある。かといって、サーバのデバッグや障害の調査のたびにいちいちsocketを使ったプログラムを作るのも面倒くさい……というときに大活躍するのがnetcatという小さなツールだ。

netcatは、TCPとUDPのパケットに関する必要な操作、たとえば任意のTCP、UDPのポートにコマンドを送ったり、それを任意のポートで受け取ったり、あるいはsource routeと呼ばれる発信元アドレスの書き換えをしてパケットを送ったりと、さまざまなパケットの操作をすることができる。

サーバ管理者、ネットワーク管理者にとっては非常に便利なツールで、netcatを使ったシェルスクリプトでちょっとしたクライアント／サーバシステムを作ったり、デバッグツールを作ることもできてしまう。「ハッカーのスイス・アーミーナイフ」と呼ばれるゆえんだ。

Red Hatをはじめ、さまざまなディストリビューションに「nc」というパッケージ名で入っているので、使い方を覚えておくと、TCP/IPの勉強やデバッグなどに便利だ。

```
myhost:~$ telnet localhost 110          ← localhostのポート110(POP3)にtelnetでアクセス
Trying 127.0.0.1...
Connected to localhost.example.co.jp.
Escape character is '^]'.
+OK QPOP (version 3.0.2) at myhost.example.co.jp starting.<4579.994811785@myhost.example.co.jp>
                                        ← QPopperが応答
user yukihiro                           ← userコマンドを入力
+OK Password required for yukihiro.     ← Qpopperからのパスワード要求
pass secret                             ← "secret"というパスワードを入力
+OK yukihiro has 1 visible message (0 hidden) in 490 octets.
list                                    ← listコマンドを入力
+OK 1 visible messages (490 octets)
1 490                                   ← 490バイトのメールが一通
.
retr 1                                  ← 1番のメールを取得
+OK 490 octets
Return-Path: <yukihiro@myhost.example.co.jp>
Delivered-To: yukihiro@myhost.example.co.jp
Received: by myhost.example.co.jp (Postfix, from userid 500)
        id BF0A1771BA; Wed, 11 Jul 2001 09:34:29 +0900 (JST)
To: yukihiro@myhost.example.co.jp
Subject: test
Message-Id: <20010711003429.BF0A1771BA@myhost.example.co.jp>
Date: Wed, 11 Jul 2001 09:34:29 +0900 (JST)
From: yukihiro@myhost.example.co.jp (YASUDA Yukihiro)
X-UIDL: 098"!7p5!!gfC!!4mA!!
Status: U

get mails using telnet.

.
dele 1                                  ← 1番のメールを削除
+OK Message 1 has been deleted.
quit                                    ← 終了
+OK Pop server at myhost.example.co.jp signing off.
Connection closed by foreign host.
myhost:~$
```

画面1　POP3
telnetクライアントでPOP3を使い、POPサーバから電子メールを読み出すことができる

おなじみのApache（HTTP）、Wu-FTPD（FTP）、sendmail（SMTP）といったソフトウェアが用意され、それぞれのサービスを提供する。

このレベルでは、サービスごとにそれぞれのプロトコルが存在し、それぞれのプロトコルは独自の特徴を持っているため、一般化した説明は難しいが、大きく分けてストリーム型のプロトコルとデータグラム型のプロトコルに分類できる。

ストリーム型のプロトコルは、主にTCPを使うHTTPやSMTPなどでよく使われるタイプのプロトコルだ。このタイプのプロトコルは、TCPのストリームにアスキーキャラクタのコマンドを送ると、その反応がアスキーキャラクタまたはバイナリデータなどで送り返されてくる。このタイプのプロトコルでは、一般的なtelnetクライアントを使って直接コマンドを入力して、サーバの動作を確認することができる。

たとえば、コマンドラインから“telnet ホスト名 80”というコマンドでtelentクライアントを起動すると、指定したホストのHTTPサーバ（ポート80）にTCP接続する。ここでHTTPコマンドを入力すると、サーバはそのコマンドを受け取り、適切な情報を返す。具体的には、“HEAD / HTTP/1.0”といったコマンドをtelnetのウィンドウにタイプすると、/index.htmlのヘッダが返ってくる。

この種のストリーム型のプロトコルを使うサーバとしては、SMTP（電子メール）、POP3、IMAP、NNTP（ネットニュース）などがある（**画面1**）。FTPもプロトコルとしてはストリーム型なのだが、FTPの場合、制御用のコネクションのほかにデータ転送用のコネクションを張る

ため、telnetクライアントだけでFTPを実行することはできない。

もうひとつのデータグラム型プロトコルは、一定の書式にしたがってバイナリデータを埋め込んだパケットを使って通信を行うもので、主にUDPでの通信によく使われる。

データグラム型のプロトコルを使う代表的なサービスにはDNSがある。データグラム型のプロトコルは、コマンド指向のストリーム型のプロトコルのようにtelnetクライアントで対話型で情報をやりとりするというわけにはいかない。しかし、主にUDP で使われることからもわかる通り、必要な情報のパッケージをUDPに詰め込んで送れば、それに対してサーバは適切な情報をUDPに詰め込んで返してくる（**図9**）。考え方としては非常に単純なもので、クライアントもサーバもコマンドを組み立てたりパースする必要がないため、プログラムの構造をシンプルにすることができる。

■Sun RPC

TCP／UDPのポート番号でサーバに接続する一般的なTCP/IPの通信とは性格が異なるが、RPCと呼ばれるクライアント／サーバ通信の仕組みがある。

RPCは、Remote Procedure Call（遠隔手続き呼び出し）という名前が示すように、基本的にはサーバ側の機能をクライアントから利用することを目的とした仕組みで、トランスポート層としてはTCP/IPのUDPを利用している。主に分散処理などに使われるプロトコルで、ほかのマシンが提供する機能をクライアントのプログラムからサブルーチンのように呼び出して使うことができるというものだ。身近なところではNFSやNISもRPC上のサービスとして実現されている（**図10**）。

TCP/IP的には、ポート111からRPCのportmapperと呼ばれるデーモンに接続すると、portmapperは要求されているプログラムに処理を渡す。ちょうどTCP/IPが/etc/servicesに書き込まれているポートを見てサーバに処理を渡すように、RPCでは/etc/rpcに書き込まれているプログラム番号のテーブルから適切なサーバに処理を渡す。たとえば、portmapper自身はプログラム番号100000、NFSサーバは100003、マウントデーモンは100005など、RPC独自の情報を利用している。

このほかにも、最近はたとえばHTTPを使ってオブジェクトを呼び出すことができるCORBAやXML、SOAPなどの技術も使われている。しかし、これらはTCP/IPのネットワークを超えた汎用的な技術なので、ここではこれらの技術についての説明は割愛したい。

TCP/IPの概念や考え方について駆け足で概観してきた。「管理者の常識」という部分に焦点をあてて説明したため、細かい部分や専門的な知識が必要になる部分については説明を省略したが、次回は実際のサーバの動作について、もう少し具体的な部分についての説明をしていくことにしよう。

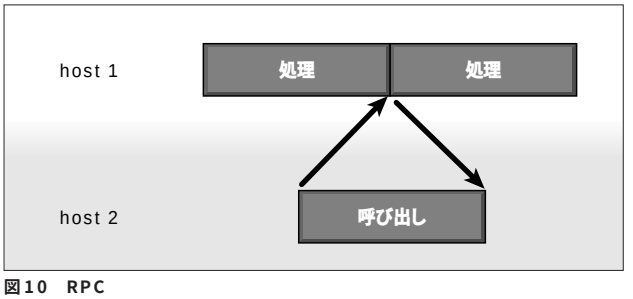


図10　RPC
RPCは、ほかのホストが提供する計算資源を利用することができる。NFSやNISなども、RPCを使って実現されている。

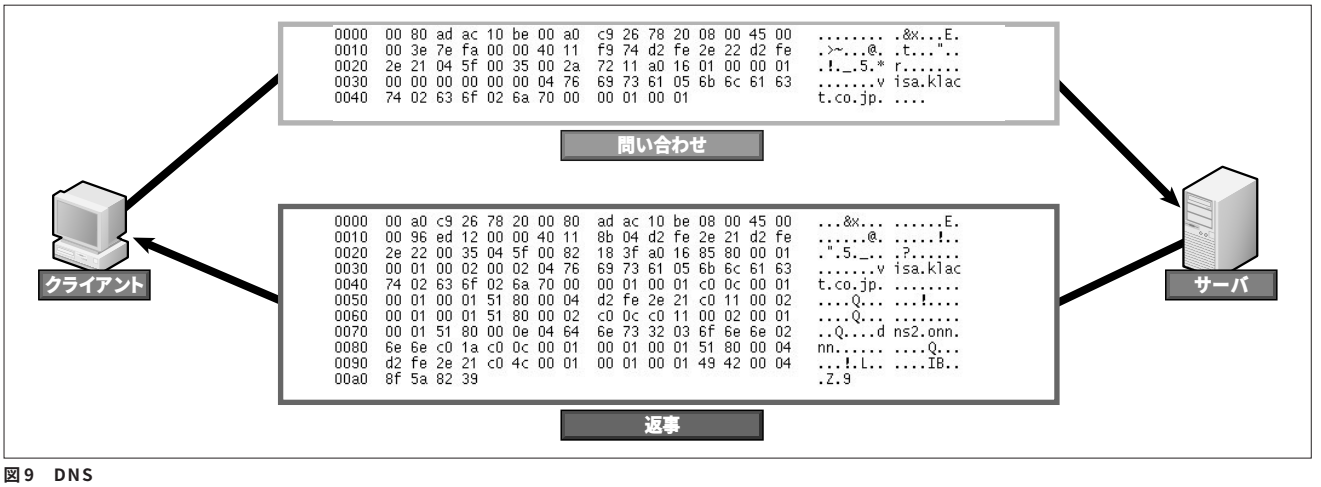


図9　DNS
DNSは問い合わせの情報をUDPパケットに入れて送ると、返事がやはりUDPのパケットに入れられて返される(パケットを表示しているソフトはethereal)。

Oracle8iで作るWebサイト入門

前回はPHPの基本をざっと見てみました。CやPerlの経験のある方なら、比較的簡単に理解できたと思います。しかし、PHPの関数群はとても強力で、ありとあらゆるものが用意されています。まずはWebアプリケーションには不可欠なフォーム処理の実例を見てみましょう。

第3回　PHPを使う（後編）

文：おもてじゅんいち／かざぐるま
Text : Junichi Omote/Kazaguruma

フォームを使う

Webでのフォームとは、テキストフィールドやプルダウンメニューなどが使われている、検索ページやアンケートのページなどで皆さんも一度は見たことのある画面です。HTMLの解説本には必ずフォームの説明もあるのですが、他のHTMLタグと違って、フォームだけはHTMLでページを作ればおしまい、というわけにはいかないのです。

フォームはそもそも、ブラウザ側から何らかのデータをサーバに送るための入力機構ですから、サーバ側でフォームのデータを受け取ってやるしくみが必要になります。これはHTMLだけではできないので、前回にも説明したCGIやサーバサイドスクリプトが使われるのです。

図1のように、フォームを使ったページはデータをサーバに送り返しますので、そのデータをPHPスクリプトが受け取って、処理をして、結果をHTMLとして返すまでが一連の動きになります。

■フォームの構成

HTMLを書いたことがあれば、フォームに関するタグを見たことがあると思います。フォームは<form>タグ、すなわちフォームそのものと、テキストフィールドやプルダウンメニューなどのフォームアイテムで構成されています（**図2**）。

フォームはひとつのページ内に複数作ることができますが、それぞれが<form>～</form>タグで囲まれていなければなりません。<form>には、name、action、method、という属性があります（**図3**）。

nameはフォームを識別するための名前を付けるためのものです。これは通常、JavaScriptなどのクライアント側のスクリプトで使われるもので、サーバサイドスクリプトには必要ありませんが、習慣として付けておくようにしたほうが良いと思います。

actionはそのフォームのデータを送る先のスクリプト（CGI）ファイル名を指定します。サブミット（送信）ボ

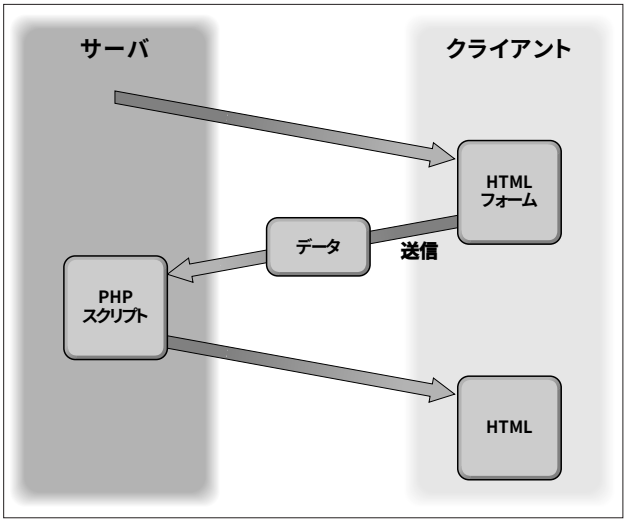


図1　フォームの動き

タンが押されるとクライアント側のブラウザはactionで指定されたスクリプト（CGI）ファイルを呼び出し、フォームアイテムの内容をデータとして送信します。

methodは、データの送り方を指定します。methodに指定できるのは、getとpostで、getのときはデータのリストをURLの末尾に、「？アイテム名1=値1＆アイテム名2=値2＆…」の形式で追加する方式です。検索サイトなどで、検索結果の画面のURLの後ろに？と＆の混じった記号の列が続いているのを見たことがあると思いますが、あれがmethod=getによるデータ送信です。

postはデータを標準入力としてサーバ側に送ります。URLには現われません。

methodを省略するとgetになりますが、getでは送信できるデータが128バイトまでに制限されます。

ではフォームアイテムをひとつずつ取り上げていきます。

■テキストフィールド

```
<input type=text name=（アイテム名）
       size=（入力欄の文字数）
       maxlength=（入力可能最大文字数）
       value=（デフォルト文字列）>
```

テキストフィールドは単一行のための文字入力フィールドです。size、maxlength、valueはオプションで、必要に応じて指定します。フォームデータを受け取ったPHPスクリプトでは、$（アイテム名）という変数に、このフィールドの内容が格納されています。

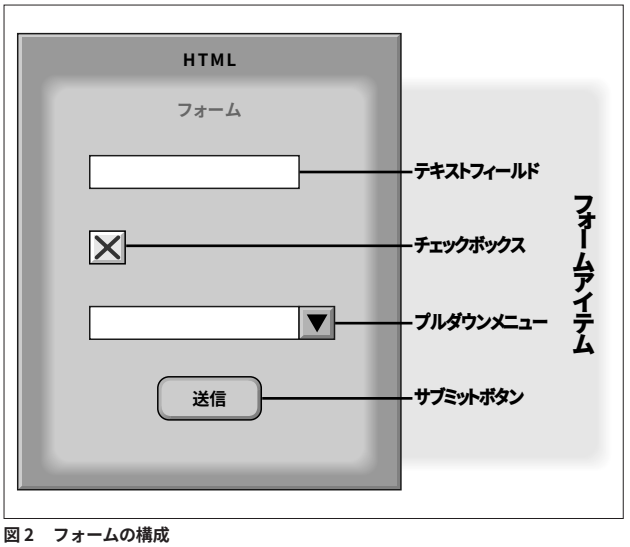


図2　フォームの構成

■パスワードフィールド

```
<input type=password name=（アイテム名）
       size=（入力欄の文字数）
       maxlength=（入力可能最大文字数）
       value=（デフォルト文字列）>
```

入力された文字が「＊」で表示されること以外はテキストフィールドと同じです。パスワード入力用に使用されます。

■テキストエリア

```
<textarea name=（アイテム名）
          rows=（行数）
          cols=（文字数）>
               ⋮
</textarea>
```

複数行の文字入力フィールドです。rows、colsはオプションで、フィールドの大きさを指定することができます。

他の<input から始まるアイテムとは違って、このタグだけが、</textarea>という閉じタグを必要とします。<textarea>と</textarea>ではさまれた部分に文字があると、デフォルト文字列として表示されます。

実際に入力された文字の量がフィールドの大きさを超えたときには自動的にスクロールバーが表示されます。

■チェックボックス

```
<input type=checkbox name=（アイテム名）
       （checked）>
```

チェックボックスです。checkedをタグ内に書いておけば、デフォルトでチェックされた状態になっています。

フォームデータを受け取ったPHPスクリプトでは、チェックされている場合は$（アイテム名）という変数の内容

```
<form  name="myform"  action="entry.php3"  method="post">
```

フォーム名　　データ送信先ファイル名　　POST方式でデータを送る

図3　<form>タグの属性

がTrue（1）に、チェックされていなければFalse（0）になります。

■ラジオボタン

```
<input type=radio name=（アイテム名）
       value=（識別用文字列）
       （checked）>
```

ラジオボタンは通常、複数のボタンを表示して、どれかひとつを選択するように使います。その場合は、

```
<input type=radio name=rd1 value=1>
<input type=radio name=rd1 value=2>
<input type=radio name=rd1 value=3>
```

というように、1グループのラジオボタンのアイテム名を同一にしておき、valueにそれぞれ違う値を指定しておきます。するとPHPスクリプト側では、$rd1 という変数に、チェックされたボタンのvalueの値が格納されます。この例では真中のボタンがチェックされていれば、$rd1の値は2になっています。

checkedをタグ内に書いておけば、デフォルトでチェックされた状態になります。

■プルダウンメニュー

```
<select name=（アイテム名）　（size=行数）>
  <option value=（識別用文字列）（selected）>（表示文字列）</option>
  <option value=（識別用文字列）>
        ⋮
</select>
```

プルダウンメニューを表示します。選択された候補のvalueが変数$（アイテム名）の値となります。

たとえば、

```
<select name="pref">
  <option value="tokyo">東京</option>
  <option value="osaka">大阪</option>
  <option value="nagoya">名古屋</option>
</select>
```

というプルダウンメニューで（**図4-A**）、「名古屋」を選んだ場合、PHPスクリプトでは変数$prefの値が"nagoya"になります。

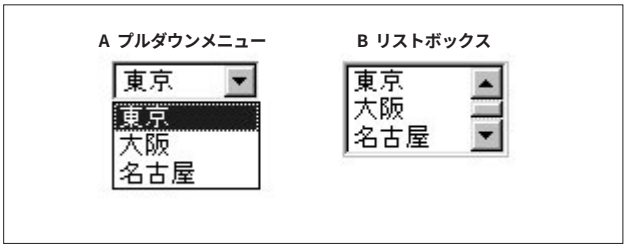


図4　プルダウンメニューとリストボックス

<select>タグ内でsizeを指定した場合は、指定した行数のリストボックスになります（**図4-B**）。

■リセットボタン

```
<input type=reset value=（ボタン名）>
```

リセットボタンを表示します。リセットボタンはすべてのフォームアイテムをデフォルトに戻します。アイテムによってはデフォルト値が設定されている場合もあるので、必ずしも「クリア」するわけではありません。

リセットボタンはアイテムの状態を元に戻すだけなので、サーバ側にデータは送られません。

■サブミットボタン

```
<input type=submit value=（ボタン名）>
```

送信ボタンを表示します。送信ボタンが押されると、ブラウザはフォームアイテムのデータをまとめてサーバ側に送信します。

■フォームデータの受け渡し

フォームデータを受け取って処理するスクリプトはアクションスクリプトとも呼ばれます。つまり、formタグ内のactionに指定されたスクリプトファイルのことです。

アクションスクリプトは、フォームのサブミットボタンが押されると、ブラウザによって自動的に呼び出されますので、スクリプト内では各フォームアイテムのデータを受け取って処理をするだけです。

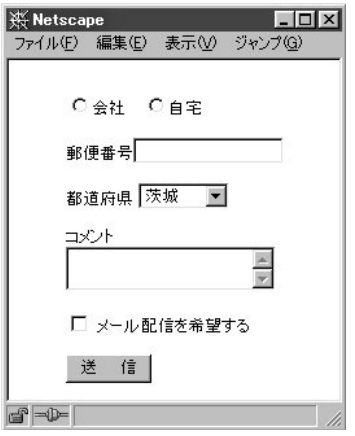
各フォームアイテムのところでも説明しましたが、PHPを使う場合、アクションスクリプト内では基本的に、

```
$（アイテム名）
```

という変数に、各アイテムのデータが格納されています。

```
<form action="test.php3">

<input type=text name="zipcode">                    $zipcode=(入力された文字)

<input type=radio name="address" value="com">会社<br>
<input type=radio name="address" value="home">自宅<br>   $address = "com" または
                                                          "home"

<select name="pref">
   <option value="tokyo">東京
   <option value="osaka">大阪                       $pref = "tokyo" または
   <option value="nagoya">名古屋                              "nagoya" または
                                                              "osaka"
</select>

</form>
```

図5　フォームアイテムとアクションスクリプト内の変数



画面1　登録フォームの例

フォームとアクションスクリプト内の変数の対応例を**図5**にまとめてみました。

PHPではこのように、フォームから送られてくるデータを、特に意識せずに通常の変数の値として利用できます。これまでのCやPerlで作ったCGIではこうはいかず、フォームとのやりとりが結構面倒でした。このあたり、PHPがフォーム処理などの純粋なサーバサイドスクリプトであることのメリットを感じます。

フォーム処理の実例

では実際にスクリプトを書きながら、フォーム処理の流れを見ていきましょう。

画面1のような登録フォームはよくあると思います。ここでは、このフォームに入力された情報をサーバで受け取って、**図6**のような登録情報確認画面を表示するものとします。

リスト1は、**画面1**のフォームを表示するためのHTMLソースです。このフォームから送られてきた各アイテムのデータを受け取って、**図6**のような画面をHTMLとしてブラウザに送り返すスクリプトを書けばよいことになります。

サーバサイドスクリプトの目的はHTMLファイルを生成することですから、このようなスクリプトを考える場合、まず完成形となるHTMLファイルを自分で作ってみることをお勧めします。普通は、スクリプトによって処理されない固定のHTMLタグなどが多く含まれるため、まずその部分を完成させます。そしてそのHTMLソースの中でスクリプトによって変化する部分を抜き出して、そこにスクリプトを埋め込んでいくほうが作業がスムーズに進むからです。

図6　登録確認画面

リスト2が、登録情報確認画面の完成形です。本来スクリプトによって変化する部分にはダミーの文字を入れてあります。**リスト2**の網掛け部分が、スクリプトによって変化する部分になりますから、この部分にスクリプトを埋め込んでいけばよいことになります。

■PHPスクリプトを書く

リスト3は、**リスト2**の完成形をもとにして、変化すべき部分にスクリプトを埋め込んだものです。順に見ていきましょう。

まず「会社／自宅」の部分です（①）。フォームではラジオボタンになってます。結果は$addrに格納されますが、$addrは「会社」を選べば"com"が、「自宅」を選べば"home"という値になっているはずです。確認画面の表示ではまた「会社」「自宅」と表示しなければなりませんから、"com"、"home"という形で渡された値を「会社」「自宅」に変換します。冒頭のスクリプト部分の、

```
if ($addr == "com")  $addr = "会社";
else                 $addr = "自宅";
```

▼リスト1　登録フォームのHTMLソース

```
<html>
<head></head>
<body>

<form action="register.php3">
<input type=radio name="addr" value="com">会社
<input type=radio name="addr" value="home">自宅
<br><br>
郵便番号<input type=text name="zipcode">
<br><br>
都道府県
<select name="pref">
   <option value="0">茨城
   <option value="1">群馬
   <option value="2">山梨
   <option value="3">埼玉
   <option value="4">東京
   <option value="5">千葉
   <option value="6">神奈川
</select>
<br><br>
コメント<br>
<textarea name="comment">
</textarea>
<br><br>
<input type=checkbox name="mailreq">
メール配信を希望する
<br><br>
<input type=submit value="送　　信">
</form>

</body>
</html>
```

がその処理で、"com"の場合に「会社」、それ以外の場合は「自宅」という値に変換しています。

$addrの値が「会社」または「自宅」になったので、①のところで、その値を表示しています。

```
<?php echo($addr) ?>
```

はよく使われる方法ですが、この部分に$addrの値を出力（表示）せよということです。これをHTMLタグの中に埋め込んでいけば、表示内容をスクリプトで変化させることができますね。

郵便番号についてはもっと簡単です（②）。$zipcodeは何の変換をする必要もなく、フォームから受け取ったデータをそのまま表示すればよいだけですから、このようにechoスクリプトを埋め込んでいます。コメントの部分も同じです。

▼リスト2　登録情報確認画面の完成形（HTML）

```
<html>
<head></head>
<body>
<center>

登録情報<br><br>
<table border=1>
<tr><td>会社／自宅</td><td>会社</td></tr>
<tr><td>郵便番号</td><td>151-8024</td></tr>
<tr><td>都道府県</td><td>東京都</td></tr>
<tr><td>コメント</td><td>○○○○○○○○
</td></tr>
<tr><td>メール配信</td><td>希望する</td></tr>

</table>

<form action="register2.php3">
<input type=submit value="登　録">
</form>
</center>
</body>
</html>
```

都道府県の部分はちょっと工夫しています（③）。リスト1のフォームのソースを見てください。プルダウンメニューで7つの県名を選択できるようになっています。プルダウンメニューの表示は県名ですが、フォームから送られてくるデータは県名でなく、0から6の数字になっています。文字データをそのまま送ると、データが大きくなってしまうことや、日本語の文字では文字化けが起きたりすることもあるため、このように数字に対応させてデータを送ることもよくあります。

この場合、スクリプト側では数字を受け取って、その数字に対応する県名を表示しなければなりません。会社／自宅のときのように2つくらいならif～elseを使ってもよいのですが、個数が多くなると煩雑になります。そこで、配列を使います。

スクリプトの冒頭で、あらかじめ県名を配列に格納しておきます。このとき、配列内の順序をフォームから送られてくる数字に対応させておけば、その数字が対応する県名のインデックス番号になってくれるのです。つまり、配列$plistに県名のリストを入れておいて、$prefにフォームから数字が送られてきたとき、

```
$plist[$pref]
```

としてやれば、うまく県名を取り出すことができるのです。この方法であれば、候補の県名が何件になってもこのまま

▼リスト3　登録フォームのアクションスクリプト（register.php3）

```
<?php
  if ($addr == "com")  $addr = "会社";
  else                 $addr = "自宅";

  $plist = array("茨城","群馬","山梨"
         ,"埼玉","東京","千葉","神奈川");
?>
<html>
<head></head>
<body>
<center>

登録情報<br><br>
<table border=1>
<tr><td>会社／自宅</td>
<td><?php echo($addr) ?></td></tr>          ①
<tr><td>郵便番号</td>
<td><?php echo($zipcode) ?></td></tr>       ②
<tr><td>都道府県</td>
<td><?php echo($plist[$pref]) ?></td></tr>  ③
<tr><td>コメント</td>
<td><?php echo($comment) ?></td></tr>
<?php
  echo("<tr><td>メール配信</td><td>");
  if ($mailreq)  echo("希望する");
  else           echo("希望しない");          ④
  echo("</td></tr>");
?>
</table>

<form action="register2.php3">
<input type=submit value="登　録">
</form>
</center>
</body>
</html>
```

のスクリプトが使えます。件数に制限がありません。そして③のところで表示しています。

最後にメール配信希望ですが（④）、ここではフォームから送られてくるデータがTrue／Falseになります。メール配信を希望するときTrue（真）、しないときFalse（偽）ですから、それぞれのときに「希望する」「希望しない」という文字を表示させています。他の部分と違い、ここではフォームからのデータを加工したりして利用するのではなく、場合に応じてスクリプトの処理を変えています。

また、④の部分では①②③のときのスクリプト埋め込み型でなく、スクリプトのブロックを作って、その中からHTMLタグも出力する方法をとってみました。

<?php と ?> の中では、HTMLタグはそのまま書くことができませんから、固定したタグ部分もechoを使って出力しています。肝心のスクリプト処理は、$mailreqの真偽に応じて、「希望する」か「希望しない」を表示します。

①②③のような埋め込み型の方法も、④のようなスクリプトブロックの方法でも、どちらでなければならないということはありませんので、とりあえず両方をマスターしておいて、あとは場合に応じて使い分けることをお勧めします。

以上、意識的にいくつかの違った方法でフォームを処理するスクリプトを書いてみました。それぞれ一長一短ですから、臨機応変に使いこなしてください。どの方法を使ってよいのか迷った場合は、できるだけスクリプトが簡単になって短くなる方法をとるのがコツです。

データベースへのアクセスなどは次回以降に学びますので、今回のスクリプトは単にフォームを受け付けるだけでしたが、複雑なWebアプリケーションでもスクリプトの役割はたいして変わりません。できれば、この単純なスクリプトでHTMLとスクリプトの関係、フォームとスクリプトのやりとりなどの原理をイメージできるようにしてください。

PHPのいろいろな関数

PHPはC言語とPerlの特徴をあわせ持つだけでなく、さまざまな機能を持つ関数が用意されています。ここまで用意されている言語はほかに見当たらないので、ひと通り紹介します。とはいってもすべて解説するのはとても無理なので、特徴的な関数群をあげてみます（巻末表）。興味がわいたら、詳しくは書籍等を参考にしてください（PHP4で新設された関数も含んでいます）。

■ネットワーク関数

低レベルのソケット関数だけでなく、各種インターネットサービスのための関数が揃っています。紙幅の関係でここでは取り上げていませんが、IMAP、POP関連の関数も用意されています。

■イメージ関数

CGIでは扱いにくかったイメージ（グラフィックス）を生成するための関数が用意されています。これらを使えばWebサーバでダイナミックにイメージを生成することができます。グラフを描くとか、地図に別のイメージを重ねるとか。特に携帯電話でのWebサービスでは、ダイナミックなイメージを活用することで、表示文字数の制限をカバーすることができそうです。

■正規表現関数

POSIX拡張正規表現とPerl互換の正規表現をサポートしています。

■マルチバイト関数

日本語処理には不可欠な、マルチバイト処理関数です。

■URLエンコード／デコード関数

base64やURLのエンコード／デコード関数です。

■XML関数

XMLのパーサとして、SAXパーサ、DOMパーサ（PHP4のみ）がサポートされています。

■データベース関数

本連載ではOracle8iを使いますので、Oracle関連の関数は次回以降に詳しく解説します。ただしPHPではOracleだけでなく、データベースまたはデータベースコネクションとして、

```
PostgreSQL
MySQL
Informix
Sybase
SQL Server
InterBase
ODBC
dba
LDAP
```

といった多くのデータベース関数が用意されています。

■その他の関数

まだまだここで紹介したくらいではPHPのすべての関数を説明したとはいえません。PDFやプリンタ制御、多様な数学関数など、きりがないほどの関数群が用意されています。ぜひ一度、書籍や関連サイトをのぞいてみてください。

C言語でCGIを作ったことのある方は、正規表現のライブラリがなかったり、XML処理やエンコード、日本語処理などで、苦労したことがあると思います。Perlは文字列処理や正規表現が得意ですが、ちょっと前のバージョンだと、ちょっとしたネットワーク機能やデータベースとのコネクションなどが難しかったものです。

PHPは入門としてもわかりやすいスクリプト言語ですが、そんな経験者の方々から見ても、PHPの関数群の威力はけっこうすごいものだといえるでしょう。ただ、多すぎるために、そんな関数があったなんて知らなかった、というのだけは避けたいものです。

プログラミングの解説書には、まったくの入門書をのぞいて、たいてい関数一覧といったリファレンスがついています。リファレンスだけで1冊の書籍もあります。みなさんの中でこのリファレンス部分を読み飛ばしている方はいませんか。辞書のようなリファレンス部分を「読む」というのは変に聞こえるかもしれませんが、まさに「読む」ことをおすすめします。

本来リファレンスというものは必要なときに調べるために使うものなので、もちろん調べるために使ってもらえばよいのですが、最初に必ず一度は通読してほしいのです。

言語仕様やライブラリ関数というものはおもしろいもので、そのようなリファレンスをながめていると、思わぬ便利な関数や、おもしろい関数が見つかることがあるのです。そしてその関数を知ったことで、新しいアプリケーションの発想が生まれることもあったりします。何か解決しなければならないプログラミング上の課題をかかえている場合でも、あるひとつの関数がちょうどばっちり解決してくれることなんかもよくあります。

つまりプログラミングそのものの技量よりも、ライブラリ関数を知っているか、知らないかがアプリケーションの開発能力を大きく左右するようになってきています。プログラミングの世界もまさに、「情報を制するものは……」といったところでしょうか。

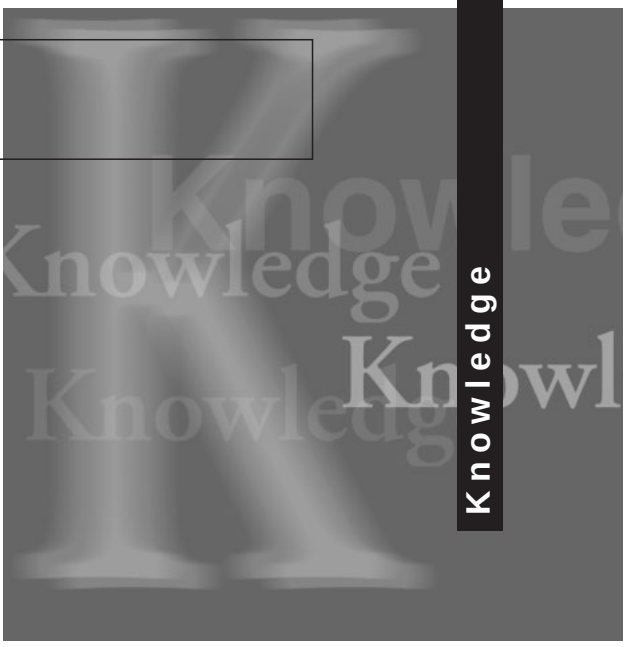
ネットワーク関数	
checkdnsrr	指定したインターネットホスト名もしくはIPアドレスに対応するDNSレコードを検索する
closelog	システムログへの接続を閉じる
debugger_off	PHPの内部デバッガを無効にする
debugger_on	PHPの内部デバッガを有効にする
define_syslog_variables	syslogに関係する全ての定数を初期化する
fsockopen	インターネットもしくはUNIXドメインのソケット接続をオープンする
gethostbyaddr	指定したIPアドレスに対応するインターネットホスト名を取得する
gethostbyname	インターネットホスト名に対応するIPアドレスを取得する
gethostbynamel	指定したインターネットホスト名に対応するIPアドレスのリストを取得 する。
getmxrr	指定したインターネットホスト名に対応するMXレコードを取得する
getprotobyname	プロトコル名のプロトコル番号を得る
getprotobynumber	プロトコル番号が指すプロトコル名を得る

getservbyname	インターネットサービスおよびプロトコルが関連するポート番号を得る
getservbyport	ポートおよびプロトコルに対応するインターネットサービスを得る
ip2long	(IPv4) インターネットプロトコルドット表記のアドレスを適当なアドレ スを有する文字列に変換する
long2ip	(IPv4) インターネットアドレスをインターネット標準ドット表記に変換 します。
openlog	システムログへの接続をオープンする
pfsockopen	持続的なInternet またはUnix ドメインソケット接続をオープンする
socket_get_status	既存のソケットリソースに関する情報を返す
socket_set_blocking	ソケットにおけるブロック非ブロックモードの設定
socket_set_timeout	ソケットのタイムアウト時間を設定する
syslog	システムログのメッセージを生成する
accept_connect	ソケットへの接続を許可する
bind	ソケットに名前をバインドする
connect	ソケット上の接続を初期化する
listen	ソケット上で接続待ち(listen)する
read	ソケットから読みこむ
socket	ソケットを作成する(通信時の終端)
strerror	ソケットエラーの内容を文字列として返す
write	ソケットに書き込む
header	HTTP ヘッダを送信する
headers_sent	ヘッダーが送信されている場合に TRUE を返す
setcookie	クッキーを送信する
ftp_connect	FTP 接続をオープンする
ftp_login	FTP 接続でログインする
ftp_pwd	現在のディレクトリ名を返す
ftp_cdup	親ディレクトリに移動する
ftp_chdir	FTP サーバー上でディレクトリを移動する
ftp_mkdir	ディレクトリを作成する
ftp_rmdir	ディレクトリを削除する
ftp_nlist	指定したディレクトリのファイルの一覧を返す
ftp_rawlist	指定したディレクトリの詳細なファイル一覧を返す
ftp_systype	リモート FTP サーバーのシステム型ID を返す
ftp_pasv	パッシブモードをオンまたはオフにする
ftp_get	FTP サーバーからファイルをダウンロードする
ftp_fget	FTP サーバーからファイルをダウンロードし、オープン中のファイルに保存する
ftp_put	FTP サーバーにファイルをアップロードする
ftp_fput	オープン中のファイルを FTP サーバーにアップロードする
ftp_size	指定したファイルのサイズを返す
ftp_mdtm	指定したファイルが最後に修正された時間を返す
ftp_rename	ftp サーバー上のファイルの名前を変更する
ftp_delete	ftp サーバー上のファイルを削除する
ftp_site	SITE コマンドをサーバーに送信する
ftp_quit	FTP 接続を閉じる
イメージ関数	
GetImageSize	JPEG、GIF、PNG、SWF 画像の大きさを取得する
ImageAlphaBlending	イメージを blending モードに設定する
ImageArc	部分楕円の描画
ImageFilledArc	部分楕円を描画し、塗りつぶす
ImageEllipse	楕円を描画する
ImageFilledEllipse	塗りつぶされた楕円を描画する
ImageChar	水平に文字を描画
ImageCharUp	垂直に文字を描画
ImageColorAllocate	画像で使用する色を作成する
ImageColorDeAllocate	イメージの色リソースを開放する
ImageColorAt	ピクセルの色のインデックスを取得
ImageColorClosest	指定した色に最も近い色のインデックスを取得する
ImageColorClosestAlpha	指定した色+アルファ値に最も近い色のID を取得
ImageColorExact	指定した色のインデックスを取得する
ImageColorExactAlpha	指定した色+アルファ値のID を取得
ImageColorResolve	指定した色または出来るだけ近い色のインデックスを得る
ImageColorResolveAlpha	指定した色+アルファ値または最も近い色のID を取得する
ImageGammaCorrect	GD イメージにガンマ補正を適用する
ImageColorSet	指定したパレットインデックスの色を設定する
ImageColorsForIndex	カラーインデックスからカラーの取得
ImageColorsTotal	画像パレットの色数の検出
ImageColorTransparent	透明色の定義
ImageCopy	画像の一部をコピーする
ImageCopyMerge	イメージの一部をコピー、マージする
ImageCopyMergeGray	グレースケールでイメージの一部をコピー、マージする
ImageCopyResized	画像の一部の複製とサイズ変更
ImageCopyResampled	再サンプリングを行いイメージの一部をコピー、伸縮する
ImageCreate	パレットを使用する新規画像の作成
ImageCreateTrueColor	TrueColor イメージを新規に作成する
ImageTrueColorToPalette	TrueColor イメージをパレットイメージに変換する
ImageCreateFromGif	ファイルまたはURL から新規画像を作成
ImageCreateFromJPEG	ファイル又はURL から新規JPEG 画像を作成する
ImageCreateFromPNG	ファイルまたはURL から新規PNG 画像を作成する
ImageCreateFromWBMP	ファイルまたはURL から新規イメージをWindows ビットマップ形式のイメー ジを作成する
ImageCreateFromString	文字列の中のイメージストリームから新規イメージを作成する
ImageDashedLine	ダッシュライン(破線)の描画
ImageDestroy	画像の消去
ImageFill	塗り潰し
ImageFilledPolygon	塗りつぶした多角形の描画
ImageFilledRectangle	塗りつぶした矩形の描画
ImageFillToBorder	特定色での塗りつぶし
ImageFontHeight	フォントの高さの取得
ImageFontWidth	フォントの幅の取得
ImageGif	ブラウザーまたはファイルへ画像を出力する
ImagePNG	PNG イメージをブラウザまたはファイルに出力する
ImageJPEG	画像をブラウザまたはファイルに出力する
ImageWBMP	ブラウザまたはファイルにイメージを出力する
ImageInterlace	インターレースを有効もしくは無効にする

ImageLine	直線の描画
ImageLoadFont	新規フォントのロード
ImagePolygon	多角形の描画
ImagePSBBox	PostScript Type1 フォントを用いてテキスト矩形のバウンディングボックス を指定する
ImagePSEncodeFont	フォントの文字エンコードベクトルを変更する
ImagePSFreeFont	PostScript Type フォント用メモリを解放する
ImagePSLoadFont	ファイルから PostScript Type フォントをロードする
ImagePsExtendFont	フォントを展開または圧縮する
ImagePsSlantFont	フォントを傾ける
ImagePSText	PostScript Type1 フォントを用いて画像の上に文字列を描く
ImageRectangle	矩形の描画
ImageSetPixel	点の生成
ImageSetBrush	線の描画用にブラシイメージを設定する
ImageSetTile	塗りつぶし用のイメージを設定する
ImageSetThickness	線描画用の線幅を設定する
ImageString	文字列を水平に描画する
ImageStringUp	文字列を垂直に描画する
ImageSX	画像の幅を取得
ImageSY	画像の高さを取得する
ImageTTFBBox	TypeType フォントを使用したテキストの bounding box の生成
ImageTTFText	TrueType フォントを使用したテキストの出力
ImageTypes	使用中の PHP の実行ファイルでサポートされているイメージの型を返す
read_exif_data	JPEG から EXIF ヘッダを読み込む
正規表現関数	
ereg	正規表現にマッチさせる
ereg_replace	正規表現による置換
eregi	大文字小文字を区別せずに正規表現によるマッチングを行う
eregi_replace	大文字小文字を区別せずに正規表現による置換を行う
split	正規表現により文字列を分割し、配列に格納する
spliti	大文字小文字を区別しない正規表現により文字列を分割し、配列に入れる
sql_regcase	大文字小文字を区別しないマッチングのための正規表現を作成する
Perl 互換の正規表現関数	
preg_match	正規表現検索を行う
preg_match_all	グローバル正規表現検索を行う
preg_replace	正規表現検索および置換を行う
preg_replace_callback	正規表現検索を行い、コールバック関数を使用して置換を行う
preg_split	正規表現で文字列を分割する
preg_quote	正規表現文字をクオートする
preg_grep	パターンにマッチする配列の要素を返す
マルチバイト関数	
mb_language	カレントの言語を設定/取得
mb_parse_str	GET/POST/COOKIE データをパースし、グローバル変数を設定する
mb_internal_encoding	内部文字エンコーディングの設定/取得
mb_http_input	HTTP 入力文字エンコーディングの検出
mb_http_output	HTTP 出力文字エンコーディングの設定/取得
mb_detect_order	文字エンコーディング検出順序の設定/取得
mb_substitute_character	置換文字の設定/入手
mb_output_handler	出力バッファ内で文字エンコーディングを変換するコールバック関数
mb_preferred_mime_name	MIME 文字設定を文字列で得る
mb_strlen	文字列の長さを得る
mb_strpos	文字列の中に指定した文字列が最初に現れる位置を見つける
mb_strrpos	文字列の中に指定した文字列が最後に現れる位置を見つける
mb_substr	文字列の一部を得る
mb_strcut	文字列の一部を得る
mb_strwidth	文字列の幅を返す
mb_strimwidth	指定した幅で文字列を丸める
mb_convert_encoding	文字エンコーディングを変換する
mb_detect_encoding	文字エンコーディングを検出する
mb_convert_kana	カナを全角かな、半角かな等に変換する
mb_encode_mimeheader	MIME ヘッダの文字列をエンコードする
mb_decode_mimeheader	MIME ヘッダフィールドの文字列をデコードする
mb_convert_variables	変数の文字コードを変換する
mb_encode_numericentity	文字を HTML 数値エンティティにエンコードする
mb_decode_numericentity	HTML 数値エンティティを文字にデコードする
mb_send_mail	エンコード変換を行ってメールを送信する
URL エンコード／デコード関数	
base64_decode	base64 方式によりエンコードされたデータのデコード
base64_encode	base64 方式によるデータのエンコード
parse_url	URL を解釈し、その構成要素を返します
rawurldecode	URL エンコードされた文字列をデコードする
rawurlencode	RFC1738 に基づき URL エンコードを行う
urldecode	エンコードされた URL 文字列のデコード
urlencode	URL 文字列のエンコード
XML 関数	
xml_parser_create	XML パーサを作成する
xml_set_object	オブジェクト内部で XML パーサを使用する
xml_set_element_handler	start および end 要素のハンドラを設定する
xml_set_character_data_handler	文字データハンドラを設定する
xml_set_processing_instruction_handler	処理命令 (PI) 用ハンドラを設定する
xml_set_default_handler	デフォルトのハンドラを設定する
xml_set_unparsed_entity_decl_handler	処理されないエンティティ宣言用ハンドラを設定する
xml_set_notation_decl_handler	表記法宣言ハンドラを設定する
xml_set_external_entity_ref_handler	外部リファレンスハンドラを設定する
xml_parse	XML ドキュメントの処理を開始する
xml_get_error_code	XML パーサのエラーコードを得る
xml_error_string	XML パーサのエラー文字列を得る
xml_get_current_line_number	XML パーサのカレントの行番号を得る
xml_get_current_column_number	XML パーサのカレントのカラム番号を得る
xml_get_current_byte_index	カレントの XML パーサのバイトインデックスを得る
xml_parse_into_struct	配列構造体に XML データを処理する
xml_parser_free	XML パーサを解放する
xml_parser_set_option	XML パーサのオプションを設定する
xml_parser_get_option	XML パーサからオプションを得る
utf8_decode	UTF-8 エンコードされた ISO-8859-1 文字列をシングルバイトの ISO-8859-1 に変換する
utf8_encode	ISO-8859-1 文字列を UTF-8 にエンコードする

RPMパッケージ開発講座

自分の使用している環境で正しく動作させるために、ソースプログラムにパッチを当てることがあります。パッチを適用するRPMパッケージの作成方法とそのほかの機能を紹介します。



最終回　RPMパッケージ作成のテクニック

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

みなさんこんにちは。今回でRPMパッケージ作成に関するお話も終わりとなります。でもその前に前回の話に関して、重大な補足をしておかないといけません。

前回、-ba（-bb）と--short-circuitは同時に指定できないというお話をしました。ところがこれがとある状況でできてしまうと、前号発売日よりあとに数件の報告がありました。状況を解析してみたところ、Kondara版RPMには、意図的にこの制限を外すパッチが適用されていたことが判明しました。証拠は各種ミラーから取ることのできる、Mary（Kondara 2.0）のSRPMSです。ソースパッケージ（rpm-3.0.6-16k.nosrc.rpm）を取得して各自の環境で展開してみましょう。するとSPECファイル（SPECS/rpm.spec）には、

```
Patch0: rpm-3.0.5-i586.patch
Patch1: rpm-3.0.4-shortcircuit-000213.patch ← これです
Patch2: rpm-handlecomment2.patch
```

という、いかにもな記述があります。このPatchに関してはこのあと説明しますが、ここにある名前のファイルがSOURCESディレクトリにあります。パッチの読める方はご覧ください。まさに回避パッチです。もちろんこのパッチはKondara独自のようで、開発者（この場合ダラズというべきか）が便利になるべく用意されたものです。しかし当のダラズの方より指摘も受けておりますが、最終的にサイトに公開するパッケージはちゃんと-baで作っているということです。

つまり、巨大なパッケージの作成の際にちょっとファイル指定（%files）を間違えた程度で止まってしまっては悲しいわけで、ちょこちょこ修正して、なんとかパッケージが生成できるようにしようという思惑です。ちなみにこのパッチ、rpm-4.0.2でも適用できたという噂ですが、これをお読みのみなさん、こういうこと（-ba --short-circuit）はふだんするものではありませんからご注意ください。

パッチを当てよう

さて、ようやく本題に入れます。今回もxalfを例にします。xalfは、X11 Application Launch Feedbackのことで、アプリケーションを起動するときのインジケータとして、マウスカーソルを砂時計などに変化させるプログラムです。xalfでスプラッシュスクリーンを出すには、GNOMEのコントロールセンターを使って設定します。xalfをインストールした後に、コントロールセンターの［Launch Feedback］内にある、［Display splashscreen when launching］を選んでおいてください（**画面1**）。これでメニューなどからアプリケーションを起動するときに表示されるようになります。

オリジナルのままのxalfを使うと、日本語の名前（メッセージ）は化けてしまいます（**画面2**）。今回はこの問題に

対して、パッチを当ててエセ国際化をさせてしまおうというわけです。

■パッチを作る準備

まずはパッチを作成しないといけません。パッチを作成するには、

・修正前のソース
・修正後のソース

を元に差分を作成します。変更するファイルが複数の場合にはディレクトリツリー同士で比較・差分を作成しますが、小さな修正ならツリーではなくファイル名で直接指定すればよいでしょう。

ここではその両方の場合の例を説明します。

まずは、オリジナルと修正後のソースを置くツリーを用意しましょう。xalfのソースRPM（xalf-0.12-1.src.rpm）は、Webサイト（http://www.lysator.liu.se/~astrand/projects/xalf/）よりダウンロードできます（付録CD-ROMにも収録してあります）。

とりあえずソースツリーを一度展開させます（**画面3**）。これでBUILDのディレクトリに展開されています、ディレクトリ名はxalf-0.12です。さて、パッチを当てるためには比較対象が必要なので、今作成されたディレクトリをオリジナルとしてディレクトリ名を変更して、さらにもう一度展開しましょう（**画面4**）。

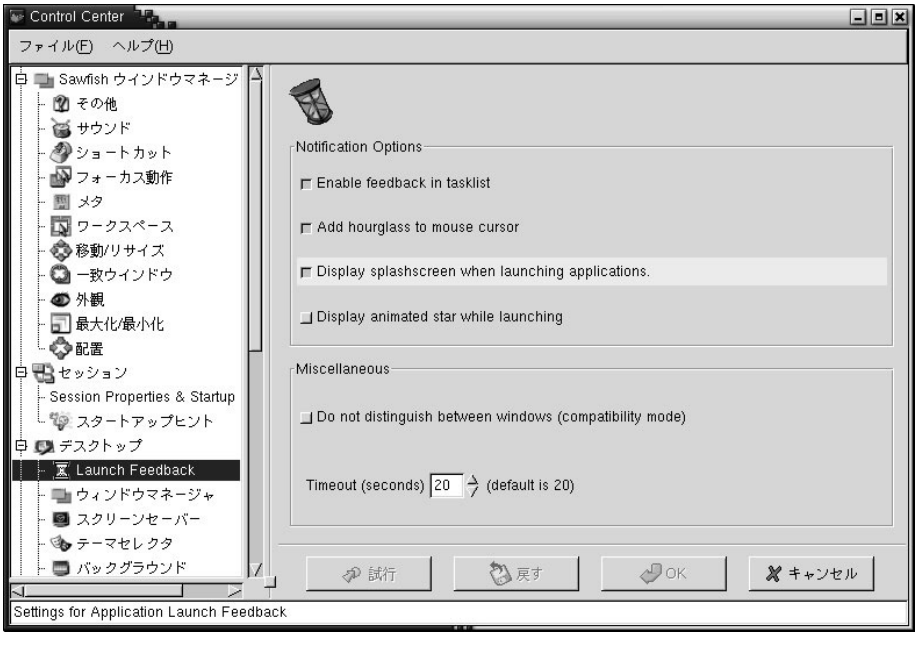
今の状態では、xalf-0.12とxalf-0.12.origは、どちらも同じですから、あとで展開したものに修正を加えてみましょう。

■パッチを作ろう（その1）

さて、このxalf、内部的にはORBit（CORBA Object Request Broker）やOAF（Object Activation Framework）といった機能を使っているかもしれませんが、この部分と今回のエセ国際化対応に関しては、たいした苦労はいりません。ただしGTK+のプログラムを一度でも（それが「Hello World」でもかまいません）書いたことがないとピンとこないのが悲しいところです。

src/xalf.cのmain（）の513～514行目に、

```
/* Initialize GTK etc. */
gtk_init (&argc, &argv);
```



画面1　GNOMEにxalfを組み込むGNOMEのコントロールセンターで、[Launch Feedback]を設定する。

画面2　オリジナルのxalfの実行画面「starting」に続くプログラム名が文字化けしている。

```
[densuke@yuzu densuke]$ rpm -Uvh ~/xalf-0.12-1.src.rpm
xalf                        ##################################################
[densuke@yuzu densuke]$ cd redhat/SPECS
[densuke@yuzu SPECS]$ rpm -bp xalf.spec
[densuke@yuzu SPECS]$ pushd ../BUILD; ls
xalf-0.12
```

画面3　xalfソースパッケージをインストール・展開

というGTK+の初期化があります、実はGTK+プログラミングにおいて国際化の初めの一歩とされているロケールの対応が入っていないだけなのです。そこで、次のように変更します。

```
/* Initialize GTK etc. */
gtk_set_locale();
gtk_init (&argc, &argv);
```

gtk_set_locale（）関数を呼ぶことでsetlocale（）してくれます。たったこれだけです。本当ならフォントの項目を探して日本語のフォント名を追加するとかしたほうがいいのですが、そういうことは宿題にしましょう。

ちょうどいいサイズなので（ORBitでの通信はともかく）GTK+でのプログラムの実例としてみるにはいいと思います。さて、今回はこれだけでいいですので（本職の方が見ればもっといろいろあるはずですが）、これでパッチを作成してみましょう。パッチの作成はdiffコマンドを使います（**画面5**）。

このファイル（patch.tmp）は**リスト1**のような内容です。1行しか追加していませんから、この程度の差分にしかなりません。なお、diffに渡したオプションスイッチ（-Naur）の説明を、**表1**にまとめておきましたのでご覧ください。

次に、今作ったパッチをSOURCESディレクトリに配置しなおしましょう（**画面6**）。そしてこのパッチをSPECファイルxalf.specに追加します。パッチファイルの指定はプリアンブル部分にPatch命令を追加することにより行います。複数ある場合は識別用の番号を付けてください（**リスト2**）。そして実際にパッチを当てるコマンドを、ソースを展開（%setup）したあとに指定します。

```
[densuke@yuzu BUILD]$ mv -v xalf-0.12 xalf-
0.12.orig
xalf-0.12 -> xalf-0.12.orig
[densuke@yuzu BUILD]$ popd
[densuke@yuzu SPECS]$ rpm -bp xalf.spec
  ⋮
(省略)
  ⋮
[densuke@yuzu SPECS]$ pushd ../BUILD; ls
xalf-0.12  xalf-0.12.orig
```

できています

画面4　xalfソースパッケージをインストール・展開

```
[densuke@yuzu xalf-0.12]$ cd ..
[densuke@yuzu BUILD]$ ls
xalf-0.12  xalf-0.12.orig
[densuke@yuzu BUILD]$ diff -Naur xalf-0.12.orig xalf-0.12 > patch.tmp
[densuke@yuzu BUILD]$ ls
patch.tmp  xalf-0.12  xalf-0.12.orig
```

BUILDに戻って

パッチファイルができています

画面5　パッチファイルの作成

▼リスト1　パッチファイルpatch.tmpの内容

```
diff -Naur xalf-0.12.orig/src/xalf.c xalf-0.12/src/xalf.c
--- xalf-0.12.orig/src/xalf.c   Sun Apr 22 21:57:08 2001
+++ xalf-0.12/src/xalf.c        Mon Jul 16 21:59:47 2001
@@ -511,6 +511,7 @@
        execvp (argv[optind], argv+optind);

     /* Initialize GTK etc. */
+    gtk_set_locale();
     gtk_init (&argc, &argv);
     XSetErrorHandler (xalf_error_handler);
     dpy = GDK_DISPLAY ();
```

オプション	説明
-N	新規にファイルが作成されていた場合は元のディレクトリに空のファイルがあったものとして扱う
-a	バイナリファイルであってもパッチを作成する
-u	diffのフォーマットを“unified output format”にする（こちらのほうが主流のようです）
-r	サブディレクトリに対しても検索する

表1　diffに渡すオプションスイッチ

```
[densuke@yuzu BUILD]$ mv -v patch.tmp ../SOURCES/xalf-nise-i18n.patch
patch.tmp -> ../SOURCES/xalf-nise-i18n.patch
```

画面6　パッチをSOURCESディレクトリにコピー

▼リスト2　xalf.specプレアンブルの抜粋

```
Summary: A utility to provide feedback when starting X11 applications.
Name: xalf
Version: %ver
Release: %{rel}_no_i18n ―― ここを変更
Copyright: GPL
Group: X11/Utilities
Source: xalf-%{ver}.tgz
Patch: xalf-nise-i18n.patch ― ここで追加
(Patch1: hokano.patch) ―― もし複数ある場合はこのように番号付きで
BuildRoot: /var/tmp/%{name}-root
```

```
[densuke@yuzu SPECS]$ rpm -bp xalf.spec
  ⋮
(省略)
  ⋮
+ /bin/chmod -Rf a+rX,g-w,o-w .
+ echo 'Patch #0 (xalf-nise-i18n.patch):'
Patch #0 (xalf-nise-i18n.patch):
+ patch -p1 -s
+ exit 0
```

画面7　rpmコマンドで%prep〜%patchまでを実行

```
%prep
%setup
%patch -p1  ← パッチの適用
```

画面8　パッチを当てたxalfの実行画面
「starting　端末...」と日本語メッセージが正しく表示されている。

パッチが複数ある場合は、「%patch1 -p1」のように数字を付けます。-p1はパッチファイルに含まれるディレクトリをいくつ無視するかという指示で、パッチを当てるときは、ソースのディレクトリ（xalf-0.12）上にいるため、パッチ（**リスト1**）中の最初のディレクトリは必要ありません。-p1で補正しているわけです。

この状態で、rpmコマンドに-bpオプションを付けて、%prep〜%patchまでを実行してみましょう（**画面7**）。

こうやってパッチを当てたxalfを再度インストールしてみましょう。今回はバージョン、リリースを同じ番号にしてしまったため、一度uninstallして入れなおすか、--forceを使って強制的にインストールしてみてください。今度は日本語でも大丈夫です（**画面8**）。

■パッチを作ろう（その2）

ところで今回はたかが1行パッチなので、わざわざディレクトリを分けるほどのものでもありません。そこでxalfのソースディレクトリで、**画面9**のようにします。

こうやって作ったパッチの場合、先ほどと違ってディレクトリを削る必要はありませんので、SPECファイル（xalf.spec）では、

```
%patch -p0
```

とすることでディレクトリを削る必要がないことを教えてあげましょう。

■パッチを当てる際の注意

既存のRPMパッケージがエラーでインストールできない場合などに、それを回避するための方法をパッチとして適用するときにも同様に行えます。ただし、C言語のプログラムのパッチを作成する場合には、C言語がある程度わかっていないと難しいです。

ソースにパッチを当てる際の注意はいくつかありますが、最近メジャーなconfigureタイプのものの場合、configure自体にパッチを当てるのはやめましょう。なぜ

ならconfigureファイルは、configure.inからautoconfにより自動生成されるため、SPECファイル中でautoconfされたらせっかくのパッチが無駄になるのです。そういう時はconfigure.inにパッチを当てるようにしましょう。え、GNU m4をご存じないですか？　すいませんが、少しだけお勉強なさってください。うまくいけばsendmail-cfの中身が少しだけ読みやすくなりますから（本当）。

そのほかの機能

さて、ここまでくればそのあたりにあるSPECがだいたい追えるようになります。結局やってることは、ふだんコンパイルする手順を自動化させてるにすぎませんから、一度でも自力でコンパイル（たとえそれがconfigure; make; sudo make installだとしても）の経験があれば、動きを追うことでやってることはやっぱりほとんど同じであろういうことも理解していただけると思います。ここでRPMのそのほかの機能をいくつか紹介しておきましょう。

■サブパッケージの生成

1つのSPECファイルから複数のパッケージを生成することができます。この時はその数だけプレアンブルが必要になります。ここでは、筆者も参加している通信用語の基礎知識（**コラム**参照）のドキュメントをパッケージ化させてみましょう。この中では、大きく4つのカテゴリに分割されて配布されています。

- **通信用語（ネットワーク上の用語やスラング）**
- **技術用語（ソフトやハード用語が主）**
- **萌え用語（人によって萌える用語が主）**
- **鉄道用語（電車関係）**

これらに対し、それぞれパッケージを作っていては面倒なので一度に作らせます。この場合、サブパッケージを定義してあげればいいので、**リスト3**のような形で%package命令により定義します。こうすると本来のパッケージ（wdic）以外にwdic-techパッケージが作成できます。

なおこの時、%descriptionもSummary（ja）同様に言語に合わせたメッセージにするための命令が使えます。

```
%description tech
This is description for English
%description tech -l ja
日本語の時の解説
```

という具合になります。

```
[densuke@yuzu xalf-0.12]$ cd src
[densuke@yuzu src]$ cp xalf.c xalf.c.orig
[densuke@yuzu src]$ vi xalf.c          ← エディタで同様に修正を加える
[densuke@yuzu src]$ cd ..
[densuke@yuzu xalf-0.12]$ diff -Naur src/xalf.c.orig src/xalf.c > ../patch.tmp
```

画面9　ディレクトリツリーを使わないパッチの作成

Column

通信用語の基礎知識について

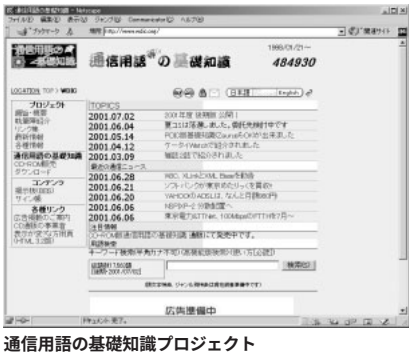
草の根BBSが繁栄していた時代においてかなり内輪向きの用語というものが氾濫しました。おかげで途中から参加してもよくわからないという問題もありました。

「通信用語の基礎知識」は、そのため多少特殊と思われる用語に対する各種の解説を付けた文書として、当時はメンテナンスされておりましたが、今ではネットワーク上で流れるさまざまな用語を扱う辞書的存在となっております。

現在は、おかのまりも氏を中心とした「通信用語の基礎知識プロジェクト」（http://www.wdic.org/）にて40人程のメンバーで、メンテナンスしつつ用語の追加等を行い、半期に一度配布されております。

本来の対象はWindows向けのテキストですが、コンバートツールや検索システムの登場により、さまざまな人に使えるようになっております。EPWING化したり、namazuに対応させたりも行われていますので、Linuxユーザーでも利用可能です。

百聞は一見にしかず、とりあえずWebページをご覧ください。



通信用語の基礎知識プロジェクト
http://www.wdic.org/

ファイル名も同様に**リスト4**のように定義できます。実際の定義例は、付録CD-ROMに収録したwdic.specを参照してみてください。

用語ファイル（wdic＊.lzh）をredhat/SOURCEディレクトリにコピーし、このSPECファイルを用いてパッケージを作成すると、

- **wdic-2001_final-1.noarch.rpm**
- **wdic-tech-2001_final-1.noarch.rpm**
- **wdic-moe-2001_final-1.noarch.rpm**
- **wdic-wdic-2001_final-1.noarch.rpm**
- **wdic-rail-2001_final-1.noarch.rpm**

という5つのパッケージができます。この場合、wdic本体のパッケージには特になにも含まれず、サブパッケージとして、wdic-tech/wdic-moe/wdic-wdic/wdic-railの4つが生成されます。

このパッケージは練習用のため、あくまで単なるテキストで（テキストファイルが入るだけ）読むぐらいしかできませんが、少し手を加えることで、namazu対応をさせたり、用語の分割等も可能になります。通信用語の基礎知識プロジェクト（http://www.wdic.org/）を参照いただくと、より実用的に調整を施した拙作のパッケージを取得できます。実際にどのようにハックすればいいのかという参考にお使いください（用語集としてもご利用くださいね）。

▼リスト3　サブパッケージの定義の例

```
Summary: wdic -- Wired/Wireless Dictionary.
Summary(ja): 通信用語の基礎知識(2001年後期版)
Name: wdic
  ⋮
(省略)
  ⋮
Source0: %{wdic_down}/wdic012a.lzh
Source1: %{wdic_down}/wdic012b.lzh
Source2: %{wdic_down}/wdic012c.lzh
Source3: %{wdic_down}/wdic012d.lzh
BuildRoot: /var/tmp/%{name}-root
  ⋮
(省略)
  ⋮
--- ここからサブパッケージ定義
%package tech
Summary: wdic-tech
Summary(ja): 通信用語の基礎知識 「技術用語」編
Group: Documentation

%description tech ──── サブパッケージに対するdescription命令
wdic-tech
```

■-taによるtar.gz形式からのパッケージ生成

ちょっと悲しいことに最近使われているtar.bz2形式には対応していない（もしかすると本当に最新のRPMでは対応しているかもしれません）ようですが、-taというスイッチを使うと、アーカイブ中に.specで終わるファイルがあればそれを使ってパッケージを作るという機能が使えます。例としてCourier-IMAP（http://www.inter7.com/courierimap/）を見てみましょう（**画面10**）。

ソース（この場合はcourier-imap-1.3.8.2.20010716.tar.gz）中にSPECファイルがあればそれを利用するという動きをしてくれます。最近の正式配布のソースコードでは、SPECファイルを中に入れている場合がありますので、

▼リスト4　サブパッケージのファイル名定義の例

```
%files tech ──── サブパッケージtechが受け持つファイルリスト
%defattr(-, root, root)
%dir /usr/share/doc/wdic/tech
/usr/share/doc/wdic/tech/*
```

```
[densuke@yuzu densuke]$ rpm -ta --clean --target=i586 \
                    courier-imap-1.3.8.2.20010716.tar.gz  --nodeps
作成中ターゲットプラットフォーム: i586
Building for target i586
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.87962
+ umask 022
+ cd /home/densuke/redhat/BUILD
+ cd /home/densuke/redhat/BUILD
+ rm -rf courier-imap-1.3.8.2
+ /bin/gzip -dc /home/densuke/courier-imap-1.3.8.2.tar.gz
+ tar -xf -
  ⋮
(省略)
  ⋮
```

画面10　SPECファイルが含まれているソースからRPMを作成
Courier-IMAPではrpmコマンドの最新版を要求しているので--nodepsによって無視させています（本当はだめ）。

もしSPECファイルが存在していたらお試しください。

なお、Courier-IMAPパッケージを生成させると、3つのパッケージができます（1つのメイン、2つのサブパッケージ）。それぞれ、

- **courier-imap　本体（通常はこれでよい）**
- **courier-imap-ldap　LDAPを用いた認証用ドライバ**
- **courier-imap-mysql　MySQLを用いた認証用ドライバ**

となっています。筆者自身は本体しか入れておりません。実際の使い方に関しては、Courier-IMAPのホームページを参考にしてみてください。

■トリガーで事前、事後処理

パッケージを入れる時、外した時などの処理は、%preや%postun等で処理させることができますが、他の依存するパッケージがあった時に挙動に応じた処理をしてもいいでしょう。

```
%triggerin -- fileutils < 3.0
echo "your fileutils is very old, please update."
```

という形で書くことで、fileutilsのバージョン3.0以前のものがインストールされた時に発動するスクリプトを埋め込めるわけです。詳しい書式はrpmコマンドのマニュアルに含まれるtriggerファイルを参照してみだください。タイミングはやはりいくつかあるわけで、

- **triggerin: 条件を満たすインストールが行われた時**
- **triggerun: 条件を満たすパッケージがアンインストールされる時**
- **triggerpostun: 条件を満たすパッケージがアンインストールされた後**

となっています。

■%configで設定ファイルを保護しよう

パッケージをアップデートした時に、そのままやると、設定ファイルまで上書きされて泣く泣くやりなおしという悲しい目にあうことも少なくないと思います。

そんな時に役立つのが%config指定です。このスイッチは%files内で対象とするファイルに付けるだけでOKです。先ほどのCourier-IMAPのSPECを読むと、このような記述があります。

```
%files -f authdaemon.files
%defattr(-, bin, bin)
%config /etc/pam.d/imap
%config /etc/pam.d/pop3
```

この場合、/etc/pam.d/imapと/etc/pam.d/pop3は設定ファイルとして認識され、変更されていた場合には削除しないでおきます。実際には、.rpmsaveという拡張子付きでファイルをリネームして保存されますので、必要に応じて戻したり、修正を（比較しながら）追加したりということが可能になります。

■こんなものはもう古い、そんな時のObsoletes

昔使っていたアプリケーションがすでに別の名前のものになってしまったということって過去に何度かありました。最近だとsawfishがそうですね（sawmillという名前から変更）。この場合、sawfishをアップグレードインストールした時に両者が残ってしまって気持ち悪いので、sawfishはsawmillの置き換えになりますということを明示するのがObsoletes指定です。プリアンブルで指定します（**リスト5**）。

ただ、このようにすると、他の古いパッケージがsawmillをRequireしていて依存性の破壊ということになりえます。そのため、同時にProvideを用いてsawmillの機能も提供していることを示しておきます。これで依存性の破壊は起きなくなります。

■パッケージを作る時の助けになるBuildPrereq

パッケージを作る人達だって、なにげなくやっていると、開発に必要なパッケージを忘れたまま-bbするという事態におちいるものです。そのため、開発に必要なパッケージを埋め込むという機能が用意されています。

さきほどのsawfishのSPECを参照してください。

```
BuildPrereq: rep-gtk-libglade control-center-devel
gmp-devel >= 3.1.1 texinfo
```

という行がそのためのものです。少なくともBuildPrereqにあるものは必要なので、事前に依存性がチェックされます。たいていは開発用パッケージ（〜-devel）を要求させたり、テキストの変換ツール（texinfoやplatexなど）を

要求するのに用います。

■アーキテクチャを明示するBuildArch

普通にコンパイルしたものは、ほとんどが機種依存のコードですが、ドキュメントの場合は必ずしもアーキテクチャに依存する必要はありませんよね。

こういう時は作成時のアーキテクチャを指定することができます（普通はドキュメントのために用います）。先ほどの「通信用語の基礎知識」のSPECファイルを参照してみましょう。その中で、プレアンブルにてこのような指定が入っています。

```
Source3: %{wdic_down}/wdc012d.lzh
BuildRoot: /var/tmp/%{name}-root
BuildArch: noarch    ← noarchなので「アーキテクチャ非依存」
```

こうして作成したパッケージは、たとえアーキテクチャ依存コードが入っていてもnoarchになりますので、十分注意してご使用ください（昔そういうミスを私はやりました）。

■%setupや%buildって本当に必要なの？

いらない場合もあるかもしれません、たとえばテキストファイルを置くだけならどうなりますか？　まさにJFのドキュメントアーカイブをパッケージにしたいという場合がうってつけです。実はどちらもなくても機能します。先に例として挙げたwdicパッケージ群はまさに%installしかありません。%prepもなくても動作可能ですが、過去に作成した痕跡が残っているとさすがに失敗するので、ここでrm -frしておくのが普通です。

最後に

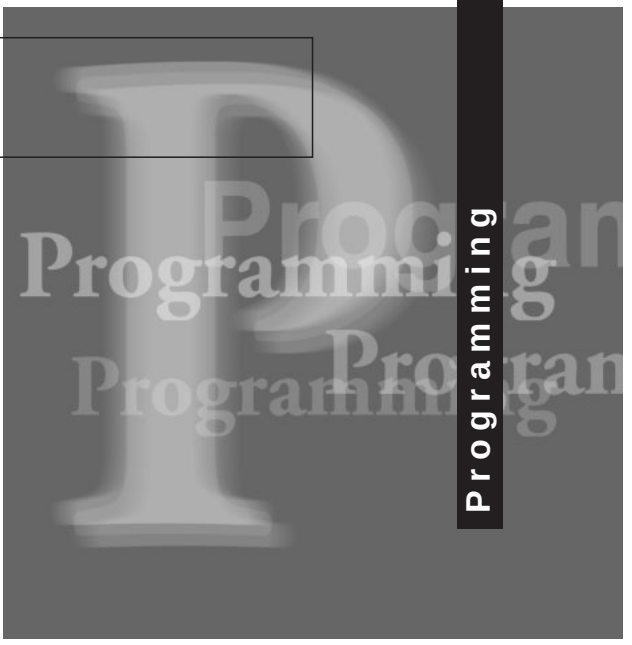
さて、ここまでででRPMを作るために必要になることをかいつまんでお話してまいりましたが、いかがでしたか？もちろんこれはあくまでとっかかりにすぎません。ほかにやることはいろいろありますが、サンプルは星の数ほどあります。なにせすべてのsrc.rpmにはSPECという名前のソースファイルが添付されていますから。各ファイルを読むことでより理解が深まっていくと思います。いっぱい見て、自分でいろいろ書いて進んでいきましょう。

この記事に関してのツッコミを個人的にされたい場合、お気軽に私までメールをお送りください。そして、この記事の補足が必要な場合も、私の個人的なWebページ「ふが日記」（http://usi.dyndns.org/~densuke/diary/）で補足を出していきたいと思っております。特に今回のwdicパッケージは本来の姿ではありません。本来の姿のパッケージもWeb上に用意する予定です。

▼リスト5　Obsoletes指定の例（sawfish）

```
Summary: a highly configurable and extensible X11 window manager
Name: sawfish
Version: 0.38
Release: 14k
Requires: librep >= 0.13, rep-gtk >= 0.14
License: GPL
Group: User Interface/Desktops
Source0: ftp://sawmill.sourceforge.net/pub/sawmill/%{name}-%{version}.tar.gz
#Source1: sawfish-20000929-ja.po
Patch0: sawfish-0.38-kondara-20010604.patch
#Patch1: sawfish-0.29-configure.patch
#Patch2: sawfish-desktop-temp.patch
#Patch3: sawfish-0.28-msgko.patch
Patch4: sawfish-gtkpath.patch
Patch5: sawfish-customize.patch
URL: http://sawmill.sourceforge.net/
Buildroot: /var/tmp/%{name}-%{version}-root
BuildPrereq: rep-gtk-libglade control-center-devel gmp-devel >= 3.1.1 texinfo
BuildRequires: rep-gtk >= 0.14, audiofile-devel >= 0.2.1
Obsoletes: sawmill   ← ここでsawmillのかわりであることを明示
Provides: sawmill
NoSource: 0
Requires: xinitrc >= 2.4.6-1k20, audiofile >= 0.2.1
```

プログラミング工房



サーバ管理システム第4回目の今月は、共有ファイルのマウント方法についてとりあげる。現状のSambaの実装では、Windowsドメインログオンができない場合があるが、Sambaのソースコードを変更することにより、ログオンスクリプトでのマウントを可能にする。

第21回　サーバ管理プログラム（4）

文：藤沢敏喜
Text : Toshiki Fujisawa

狭い範囲でしか便利でないブラウジング

Windowsでは、デスクトップの［マイコンピュータ］アイコンをクリックすると、ネットワークに接続されたPCが見える。Sambaを使ったサーバもこの中に表示され、アイコンをクリックすると、そのサーバで利用できる共有名が現れる。ユーザーはこの共有名をクリックすれば、それに対応したディレクトリにアクセスできることになる。

このように、自動的にアクセスするコンピュータがアイコンの形で現れるのは、見ためにはたいへん便利に思える。しかし、コンピュータの数が数百台になると、アイコンを探すのが困難になるばかりでなく、ネットワーク上にはブロードキャストパケットが吹き荒れることになる。また、マイクロソフトネットワークはブロードキャストにかなり依存しているため、ルータを超えるためにはいろいろと難しい面が多く、ネットワーク管理者を悩ませる。

マイクロソフトネットワークで使用するSMBプロトコルはRFC1001とRFC1002で規定されているが、ブラウジングやドメインに関する部分は闇の中にあるようで、その動作を知ることは困難である。トラブル対策のためには、膨大な金額のサポート料金を支払わなければならないらしい。マイクロソフトネットワークは、その開発当初に想定した十数台のマシンだけで使用するには便利だが、ルータがたくさん設置されているような大規模組織で使うには難しい。

これに対して、インターネットではサーバを識別するためにマシン名をDNSに登録し、そのホスト名を用いてIPアドレスの解決が行われる。この方法では、ホスト名を知らないとアクセスできないという欠点があるが、インターネットという巨大なネットワークでも破綻することなく動作し、特に不自由がないのはご存じのとおりだ。ある程度大きなネットワークでは、ブラウジングではなく、指定した名前からIPアドレスを解決することが必要なのである。

WINSによる名前解決

マイクロソフトネットワークにも、ブロードキャストに頼らないで名前解決を行うWINSというしくみがある。ただし、WINSは狭い範囲でしか使えないDNSのようなものであり、DNSが稼働している環境では非常に迷惑なシステムであると感じている。WINSのように、オープンでない独自プロトコルを使うことにより、自社のOSを有利にするというメリットがあることはわかるが、それ以外のメリットは筆者にはあまり見いだせない。しかし、インターネットが全盛になると、不便であることを認識してか、Windows NT以降ではDNSを使った名前解決ができるようになっている。

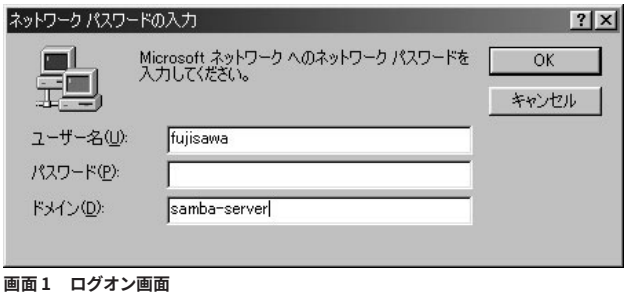
したがって、Windows NTやWindows 2000クライアントだけからなるネットワークでは、WINSのことは考え

ずに済ませることも可能だが、まだまだWindows 95は現役でありWINSと格闘せざるをえない。特に後述のドメインログオン機能を使用する場合は、WINSでの名前解決が必要になる。

共有ファイルのアクセス方法

Sambaを使い、Windowsのクライアントをターゲットとするファイル共有では、ユーザーにどのようにディレクトリを見せるかが問題となる。Windowsでは、UNC（**コラム**参照）で直接サーバを指定する方法と、共有資源にドライブレターをマップして共有ディレクトリがドライブとして見えるようにする方法がある。UNCを使えば、UNIXのrcpコマンドやNFSのように「host名:/ディレクトリ」という指定ができる。UNCを使ってアプリケーションから直接ファイルを読み書きもできるし、サーバが数百に増えても対応可能とういうスケーラビリティもある。ところが古いソフトウェアではUNCアクセスはほとんど不可能で、最近のフリーソフトでもUNCによるアクセスができないものがある。

一方、ドライブレターをマップする方式は、AからZまでの26ドライブしか使えないという制限がある。そのため、多くのサーバをマウントする必要のある規模が大きなネットワークでの使用には難がある。しかしながら、一般的なユーザーにとっては、UNCによるアクセスよりも、ドライブ単位でのアクセスのほうになじみがあるようだ。



画面1　ログオン画面

以上のように、ドライブマウント方式は制限があるので、UNCアクセスにしたいのはやまやまだが、古いソフトウェアやユーザーの教育のことを考えると、ドライブレター方式を採用せざるをえないのが現状ではないかと思う。

さて、サーバ上のディレクトリをドライブとしてマウントするには大きく分けて次のような方法がある。

- **DOSプロンプトで、net.exeコマンドを使用してマウント**
- **［ネットワークコンピュータ］アイコンを右クリックして、ドライブを選択し、マウントするUNCを入力**
- **ドメインログオンを行い、ログオンスクリプトで自動的にマウント**

最初のDOSプロンプトを使う方法は、一般のユーザーにはちょっと敷居が高い。2番目のGUIを使う方法は簡単だが、再接続時にハブの電源が抜けていたりすると、UNCを再入力しなければならず、管理者への質問電話増加が懸念される。最後は、ログオン時に自動的にマウントしてしまう方法である。ユーザーがPCを立ち上げたときに表示されるログオンウィンドウ（**画面1**）で、ユーザー名とパスワード、ログオンするサーバ名を入力すると、そのユーザーに対応したログオンスクリプトを実行できる。このログオンスクリプトの中で、net.exeコマンドを実行することにより、そのユーザーに応じたサーバの特定のディレクトリをマップすることが可能になる。また、ドメインログ

Column

Universal Naming Convention（UNC）

アスキーデジタル用語辞典（http://yougo.ascii24.com/）では、UNCは「Universal Naming Convention」の略で、Windowsネットワークにおいて、ネットワーク環境にある資源を指し示す場合に利用する表記法であると解説されている。

たとえば、ホスト名が「samba-serv」というSambaサーバの、「share01」という共有名の「\design\trial_prod」というディレクトリにあるdummy01.jpgというファイルの場合、UNCを使うと、

```
¥¥samba-serv¥design¥trial_prod¥dummy01.jpg
```

というように表記される。つまり、従来のドライブレターとコロンの代わりに、2つのバックスラッシュ（\\）で、ホスト名を記述する。

Windowsでは、DOSプロンプトのコマンドラインでも、このUNCを使用することができるようになっていて、

```
C>net use n: ¥¥samba-serv¥design
```

とすることにより、Nドライブにディレクトリをマウントすることができる。

なお、共有名のあとに「%」を使ってユーザー名を指定したり、パスワードなどを指定することもできる。またWindows 2000系では、「/USER:」などのオプションもある。詳しくは「net /?」を実行したり、リソースキットなどの書籍を参照するとよい。

オンスクリプトの中では、次のようなコマンドを実行することにより、クライアントPCの時計を自動的に合わせることもできる。

```
net time ¥¥サーバ名 /set /yes
```

クライアントPCの時計がずれていると、メールサーバやファイルサーバ管理に支障をきたすが、PCの時計は1日に秒単位で狂うことは珍しくない。時計を正確に合わせることは、サーバ管理上重要である。

時計の設定以外にも、管理者からのメッセージを表示したり、特定のプログラムを実行してファイルを置き換えるなど、ログオンスクリプトは便利である。

Sambaサーバにおけるドメインログオン設定

SambaにはWINSサーバ機能があり、WINSの名前解決を行うことができるし、Sambaを使ってドメインログオンを行うこともできる。ところで、マイクロソフトネットワーク用語の「ドメイン」は、URLやメールアドレスに使われる「ドメイン」とは、意味も概念もまったく違う言葉である。インターネットで古くから使用されている言葉を、独自の意味で使われると本当に紛らわしくて困る。

さて、マイクロソフト用語でいうドメイン（以下ではNTドメインと呼ぶ）とは、ドメインコントローラと呼ばれるマシンを核とした複数のNTサーバを集めて、論理的なまとまりにしたものである。今回作成しているシステムでは、単にドメインログオンスクリプトを実行させたいのが目的なので、本来のNTドメインが想定しているような使い方をするわけではない。そのため、今回のシステムで使用する範囲では、サーバに付けられたホスト名とは別の（紛らわしい）名前のことだと理解しておけばよいだろう。

6月号で説明したように、今回は札幌から福岡まで6台のサーバを使うことを仮定している。そこで、NTドメインを**表1**に示すような名称にする。たとえば、東京本社サーバ「tok.fujisawa.gr.jp」では、**リスト1**に示すような

設置場所	ホスト名	NTドメイン名
札幌支店	sap.fujisawa.gr.jp	sap-1
仙台支店	sen.fujisawa.gr.jp	sen-1
東京本社	tok.fujisawa.gr.jp	tok-1
大阪支店	osk.fujisawa.gr.jp	osk-1
名古屋支店	nag.fujisawa.gr.jp	nag-1
福岡支店	fuk.fujisawa.gr.jp	fuk-1

表1　ホスト名とWindows NTドメイン

smb.confファイルを用意すると、ドメインログオンが可能になる。**リスト1**の2行目で、

```
workgroup = TOK-1
```

と指定しているが、これがNTドメイン名の指定である。なお、workgroup = TOKとして、ホスト名とNTドメイン名を同一の文字列にすることはできないので注意してほしい。この行で設定した名称が、Windowsクライアントを起動したときに表示されるログインウィンドウ（**画面1**）で、ユーザー名やパスワードとともに入力するNTドメイン名となる（画面にはドメインとしか書いてないため、ユーザーはメールアドレスのドメインを入力したりして混

▼リスト1　smb.conf

```
 1 [global]
 2 workgroup = TOK-1
 3 wins support = yes
 4 domain master = no
 5 local master = no
 6 preferred master = yes
 7 os level = 0
 8 domain logons = yes
 9 security = user
10 encrypt passwords = yes
11 browseable = yes
12 username map = /usr/local/samba/lib/smb.map
13 preserve case = yes
14 short preserve case = yes
15 case sensitive = no
16 create mask = 0770
17 directory mask = 0770
18 force directory mode = 0770
19 dos filetime resolution = yes
20 map archive = yes
21 map hidden  = yes
22 map system  = yes
23 client code page = 932
24 coding system = hex
25 preserve case = yes
26 short preserve case = yes
27 oplocks = no
28 socket options = TCP_NODELAY
29 logon script = %u.bat
30 ;------------------------------
31 [netlogon]
32 comment = Net Log on
33 path = /smb/logon
34 writable = no
35 share modes = no
36 browseable = no
37 ;------------------------------
38 [N]
39 comment = N: drive on FreeBSD
40 path = /smb/home/%u
41 writable = yes
42 browseable = no
43 ;------------------------------
```

乱する)。

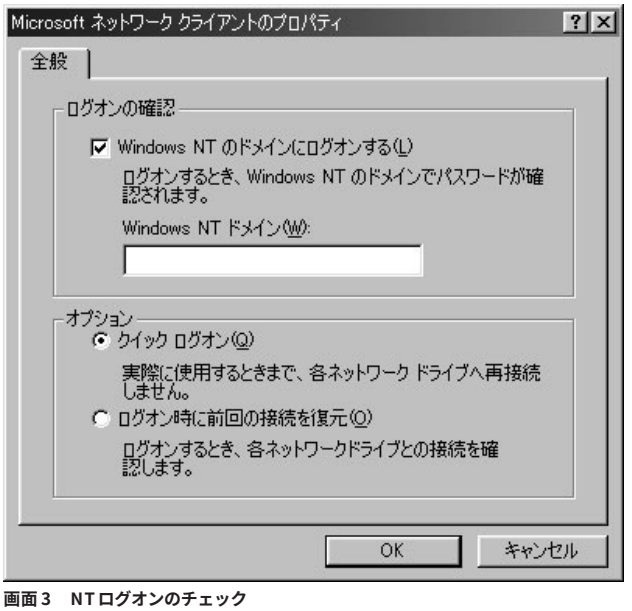
また、**リスト1**の3行目で、

```
wins support = yes
```

としているが、上記指定によりこのSambaサーバがWINSサーバとして機能することになる。そして8行目と29行目で、

```
domain logons = yes
logon script = %u.bat
```

画面2　ネットワークの設定



画面3　NTログオンのチェック

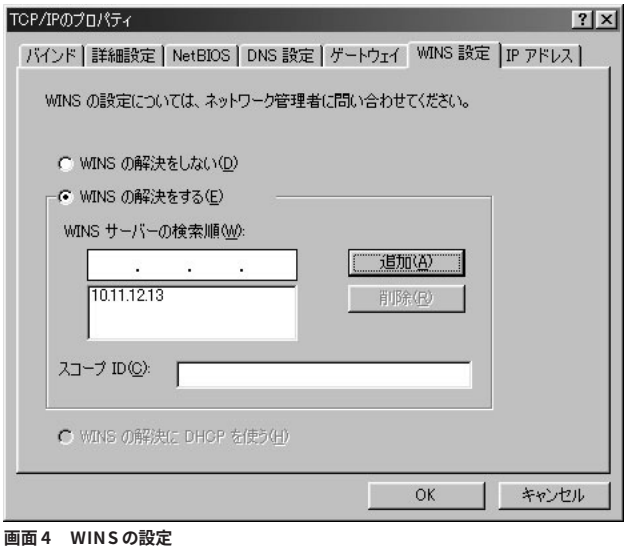
とすることにより、ドメインログオンを可能にし、ログオンスクリプトを実行できる。%uはログオンしたユーザー名の文字列に展開されるので、fujisawaとしてログオンした場合は、fujisawa.batが実行されることになる。このfujisawa.batを置くディレクトリやそのアクセスに関しては、31行目からの［netlogon］セクションで指定される。なお、fujisawa.batの内容は、改行コードをCR＋LFとした次のようなものである。

```
net time  ¥¥tok   /set /yes
net use n: ¥¥tok¥n
```

リスト1のsmb.confでは、38行目からの［N］という共有名が定義されているため、fujisawaというユーザーがログオンした場合には、tokサーバの共有名n（UNCでは\\tok\n）はNドライブにマウントされ、/smb/home/fujisawaというディレクトリが参照可能になる。

Windowsクライアントのドメインログオン設定

ドメインログオンを行うためには、Windowsのデスクトップにある［ネットワーク］アイコンを右クリックし、［プロパティ］を選択する。そして、優先的にログオンするネットワークを［Microsoftネットワーククライアント］に変更し、現在のネットワークコンポーネント内にある［Microsoftネットワーク］をダブルクリックする（**画面2**）。すると、**画面3**のようにMicrosoftネットワーククライアントのプロパティが表示されるので、［Windows NTのドメインにログオンする］をチェックする。最後に、



画面4　WINSの設定

TCP/IPのプロパティを開き、［WINSサーバの設定をする］をチェックし、WINSサーバのIPアドレスを入力する（**画面4**）。

設定が終わってリブートすれば、**画面1**のようなログオンウィンドウが現れる。ここでユーザー名とパスワード、そしてログオンするサーバに対して与えられるNTドメイン名を入力すれば、ログインスクリプトが実行される。

共用クライアントマシン

上記のようにドメインログオン機能を使うことで、共用クライアントマシンの設置が容易になる。工場の組み立てラインで8時間おきに交代勤務する場合など、自分専用のデスクを持てない社員もPCの利用が可能になる。

また、出張者にとってもこのシステムは便利である。たとえば、札幌支店から東京本社に出張してきた田中さんは、本社に設置してある共用クライアントマシンのログインウィンドウで、ユーザー名をtanakaとして入力し、いつものようにNTドメインを「sap-1」として入力することで、Nドライブにふだん自分が使用している札幌支店サーバのディレクトリをマウントすることができる。

しかしながら、このようなシステムをSambaで運用するには壁がある。現状のSambaのソースコードでは、自分自身にドメインログオンするためのWINS情報を提供することは可能だが、ルータを超えた自分以外のサーバにドメインログオンするためのWINS情報を提供することができないのである。たとえば、tok.fujisawa.gr.jpというサーバのIPアドレスが10.111.0.100で、クライアントマシンのWINS設定が、10.111.0.100となっていると、tok-1にはログオンできても、札幌支店のNTドメインsap-1にはドメインログオンすることができない。

これを解決するためには、Sambaサーバでは名前解決をせずに、WINSサーバはNTにまかせてしまうという方法がある。しかし、WINSのためだけにNTを立てるのは不経済であり、マシンの台数が増えれば故障率も指数関数的に増加する。LinuxマシンにVMwareをインストールし、その中でNTを動作させることも考えられるが、システムを複雑にすれば信頼性が落ち、遠隔地でのリモートメンテナンスも難しくなる。なによりも、Windows NTでは可能だが、Linuxでは不可能だというのは非常に不愉快である。

WINSによるNTドメイン名の名前解決

マイクロソフトのサポートページで「#DOM」をキーワードとして検索すると、WINSの代わりにlmhostsを使った名前解決の方法が見つかる。c:\windows\lmhostsに、

```
199.199.199.1  controller1   #PRE  #DOM: ドメイン名
199.199.199.1  "domainname,,,,,\0x1b"  #PRE
```

というような記述をすることにより、NTドメイン名などの登録ができることが示されている（筆者が試した環境ではうまく動作しなかった）。

ここで、上記の0x1bはWINSでの名前解決で使用される種類の番号（0xは16進数を意味する）である。マイク

Column

Sambaソースコードの解析

日本語版Microsoft Windowsには、

```
C:\>net use n: \\サーバ名\共有名
C:\>n:
N:\>mkdir  123456789
N:\>cd  123456789
N:\123456789>mkdir  漢字文字
N:\123456789>cd  漢字文字
N:\123456789\漢字文字5678>
```

というようなバグがあり、Windows Meになっても直っていない。こうしたWindowsのバグ回避や、先月号の容量制限そして今月号のドメインログオンなど、自分の環境で都合がいいようにSambaを運用しようとすると、ソースコードを変更しなければならないこともある。

ソースコードを変更するためには、それを理解する必要があるが、Sambaくらい規模が大きくなると、ただ漠然とソースコードを眺めていてはまったく理解できないものである。そのため、挙動を理解するにはデバッグのログレベルを上げたり、あちこちにprintfを挿入してさまざまな変数を眺めたりする必要がある。

しかし、このようなコードを追加するとそのための実行時間が問題を引き起こすことがある。実際、Sambaでもログレベルを上げると不具合を起こすことがあった。

このような場合は、コンソールやファイルに直接出力せずに、いったんメモリ中にデバッグメッセージを保存する。そして、タイミングを見はからって、メモリに保存したメッセージをファイルなどに出力するとよい。Sambaのようなサーバプロセスの場合、メモリからファイルへログを出力するタイミングとしては、signal（ステップアップC言語を参照）を使うのが便利である。

ロソフトのサイトではこれらの意味を見つけることができなかったが、いろいろと検索してみると、

0x00	サーバ名	ワークステーションサービス
0x03	サーバ名	メッセンジャーサービス
0x20	サーバ名	ファイルサーバ
0x1b	ドメイン名	ドメインマスタブラウザ
0x1c	ドメイン名	ドメインコントローラ

というような意味をもっているらしいことがわかった。なお、この番号は、Windowsに標準搭載されているnbtstat.exeコマンドにより確認することができる。このコマンドを用いて調べると、SambaをWINSサーバとしていて、正常にドメインログオンできない場合は、**画面5**のような結果が得られ、後述のパッチを当て正常にドメインログオンできる場合は、**画面6**のような結果が得られる。つまり、オリジナルのSambaでは0x1cが解決できないのである。

```
C>nbtstat -a sap

Lana # 0:
Node IpAddress: [10.111.222.33] Scope Id: []

           NetBIOS Remote Machine Name Table

       Name               Type         Status

-----------------------------------------------
    SAP            <00>  UNIQUE      Registered
    SAP            <03>  UNIQUE      Registered
    SAP            <20>  UNIQUE      Registered
    SAP-1          <00>  GROUP       Registered
    SAP-1          <1E>  GROUP       Registered

    MAC Address = 00-00-00-00-00-00
```

画面5　ドメインログオン不可能な場合

```
C>nbtstat -a tok-1

Lana # 0:
Node IpAddress: [10.111.222.33] Scope Id: []

           NetBIOS Remote Machine Name Table

       Name               Type         Status

-----------------------------------------------
    TOK            <03>  UNIQUE      Registered
    TOK-1          <00>  GROUP       Registered
    TOK-1          <1C>  GROUP       Registered
    TOK-1          <1E>  GROUP       Registered

    MAC Address = 00-00-00-00-00-00
```

画面6　ドメインログオン可能な場合

/usr/local/samba/lib/lmhostsでの名前解決

上記のように、SambaでWINSを提供するnmbdコマンドを、0x1cを解決できるようにソースコードを変更すれば、NTサーバを使わずにLinuxマシンだけでサービスができるようになるはずだ。もともとSambaには、/usr/local/samba/lib/lmhostsで、**リスト2**のような記述をすることにより、0x1bなどの情報を提供できるような機能がある。なお、リスト2はtok.fujisawa.gr.jpサーバの例なので、自分自身のTOK#00とTOK#20を定義する必要がある。一方、自分自身のNTドメイン名であるTOK-1#0x1bやTOK-1#0x1bを定義するとうまく動作しないので注意してほしい。一方、札幌支店のサーバのlmhostsでは、東京支店の情報を、

```
10.111.0.100     TOK-1#00
10.111.0.100     TOK-1#20
10.111.0.100     TOK-1#1C
10.111.0.100     TOK-1#1B
```

と定義し、自分自身の情報は、

```
10.222.0.100     SAP#00
10.222.0.100     SAP#20
```

▼リスト2　/usr/local/samba/lib/lmhosts

```
#-----------------------------------------------
#
#      Domain = "TOK-1"
#
10.111.0.100       TOK#00
10.111.0.100       TOK#20
#-----------------------------------------------
#
#      Domain = "SAP-1"
#
10.222.0.100      SAP-1#00
10.222.0.100      SAP-1#20
10.222.0.100      SAP-1#1C
10.222.0.100      SAP-1#1B
#-----------------------------------------------
#
#      Domain = "FUK-1"
#
10.333.0.100      FUK-1#00
10.333.0.100      FUK-1#20
10.333.0.100      FUK-1#1C
10.333.0.100      FUK-1#1B
#-----------------------------------------------
```

の2つだけを定義することになる。この管理はちょっと面倒なので、サーバの数が多い場合は、各サーバのlmhostsを自動生成するPerlスクリプトを書き、rsyncで自動配送するようにすると便利である。

Sambaソースコードの変更

上記のように、0x1cを解決できるようにlmhostsを記述しても、現状のSamba-2.0.10のソースコードでは解決できない。0x1cを解決するためには**リスト3**のパッチを適用する必要がある。なお、この部分のソースコードは、Samba-2.0.5aからほとんど変更が行われていないので、2.0.10以前のバージョンにも同様に適用できると思う。

ただし、このパッチはsmb.confで「wins support=yes」となっていないとコアダンプしてしまうという不完全なパッチであり、場合によっては不具合を引き起こす可能性があるので十分注意してほしい（筆者は自分でソースを変更したことを忘れて、1時間ほど悩んだ）。

ちなみに、この修正をSamba-2.0.5aに適用し、FreeBSDマシン十数台で2年ほど前より動作させているが、特に問題は発生していない。なお、Samba-2.0.10は2週間前に公開されたばかりのリリースである。そのため、このパッチをあてた簡単なテストはしたものの、実際に運用したことはない。ドメインログオンに関しての安定運用を目指すなら、実績のあるSamba-2.0.5a＋FreeBSDがお勧めである。

おわりに

マイクロソフトネットワーク（SMB）は、その一部がRFCとして公開されているものの、プロトコルの多くの部分が深いベールで覆われている。今回取り上げたWINSに関するドキュメントも一般に公開されているものは非常に少ない。

したがって、トラブルがあるとネットワークをモニタするtcpdumpコマンドを使って解析を行うという、虚しく膨大に時間がかかる作業が必要になるうえ、ブラックボックスの部分にバグがあるとお手上げになることが多い。このような非公開プロトコルを使用するOSがなくなり、すべてのクライアントOSがLinuxのようにオープンなプロトコルを利用する時代が一日も早く到来することを願いたい。

▼リスト3　ドメインログオンを可能にするパッチ

```
diff -rc samba-
2.0.10.org/source/nmbd/nmbd_incomingrequests.c
samba-2.0.10/source/nmbd/nmbd_incomingrequests.c
*** samba-
2.0.10.org/source/nmbd/nmbd_incomingrequests.c
Fri Jan  8 08:01:59 1999
--- samba-
2.0.10/source/nmbd/nmbd_incomingrequests.c    Mon
Jul  9 01:05:05 2001
***************
*** 469,478 ****
--- 469,488 ----
       came from a subnet we don't know about then
search all the broadcast subnets
       for a match (as we don't know what
interface the request came in on). */

+   #if 1 /* By Toshiki Fujisawa for "wins support
= yes" */
+   namerec =
find_name_on_subnet(wins_server_subnet, question,
FIND_ANY_NAME);
+   if( ! namerec ){
+       if(subrec == remote_broadcast_subnet)
+         namerec =
find_name_for_remote_broadcast_subnet(question,FIN
D_ANY_NAME);
+       else
+         namerec = find_name_on_subnet(subrec,
question, FIND_ANY_NAME);
+   }
+   #else /* By Toshiki Fujisawa for "wins support
= yes" */
    if(subrec == remote_broadcast_subnet)
      namerec =
find_name_for_remote_broadcast_subnet( question,
FIND_ANY_NAME);
    else
      namerec = find_name_on_subnet(subrec,
question, FIND_ANY_NAME);
+   #endif /* By Toshiki Fujisawa for "wins
support = yes" */


    /* Check if it is a name that expired */
***************
*** 498,503 ****
--- 508,516 ----
      if( !bcast
       || ( bcast
         && ( (name_type == 0x1b)
+          #if 1   /* By Toshiki Fujisawa for
"wins support = yes" */
+          || (namerec->data.source ==
LMHOSTS_NAME)
+          #endif  /* By Toshiki Fujisawa for
"wins support = yes" */
           || (namerec->data.source == SELF_NAME)
           || (namerec->data.source ==
PERMANENT_NAME)
           || ( (namerec->data.source ==
WINS_PROXY_NAME)
```

特別講座

ステップアップC言語

signalシステムコール

■サーバプロセスへの通信

Sambaのようにサーバとして実行されるプログラムに特定の指令を送りたい場合、signalが使われることがある。たとえば/etc/inetd.confを書き換えた場合、

```
# kill -HUP inetdのプロセス番号
```

を実行することにより、inetdプログラムに新しい/etc/inetd.confを読み込ませることができる。

killコマンドは、プロセスを殺すだけのコマンドのように思われることが多いようであるが、実際にはプロセスにさまざまなシグナルを送ることができる。

なお、プロセスを終了（TERM）させるシグナルは、多くのOSでは15番が割り当てられ、killコマンドで引数を指定しない場合のデフォルト番号は、15番（SIGTERM）である。

■signalシステムコール

killコマンドから送られたシグナルを受け取るためには、signalシステムコールを使用する。たとえば、

```
#include <stdio.h>
#include <signal.h>

void
usr1_handler(int no)
{
    printf("get signal no=%d\n", no );
}

int
main(void)
{
    signal(SIGUSR1, usr1_handler);

    for(;;){
        /* */
    }
}
```

というコードをsigtest.cとして保存して、

```
$ gcc -o sigtest sigtest.c
```

としてコンパイルを行い、X Window System上のktermなどから、

```
$ ./sigtest
```

どで実行する。そして別のktermウィンドウで、ps axを実行して、sigtestのプロセス番号を確認する。続いて、

```
$ kill -USR1 sigtestのプロセス番号
```

を実行すると、sigtestを実行している画面に、

```
get signal no=10（FreeBSDでは30）
```

という表示が行われるはずである。

■handler関数の引数

上記のようにkillコマンドでシグナルが送られると、signalシステムコールで設定した関数が呼ばれる。なお、このように割り込みが起きたときに呼ばれる関数をハンドラと呼ぶことがある。

signalシステムコールで設定したハンドラ関数はint型の変数を1つ持つが、この変数には発生したシグナル番号が渡される。たとえば、

```
#include <stdio.h>
#include <signal.h>

void
sig_handler(int no)
{
    switch( no ){
    case SIGTERM:
        printf("Can't TERMINATE\n");
        return;
    case SIGUSR1:
        printf("get SIGUSR1\n");
        return;
    case SIGINT:
        printf("Can't stop\n");
        return;
    }
}

int
main(void)
{
    signal(SIGTERM, sig_handler);
    signal(SIGINT,  sig_handler);
    signal(SIGUSR1, sig_handler);

    for(;;){
        /* */
    }
}
```

というプログラムでは、発生したシグナルに応じたメッセージが表示される。

上記のプログラムでは、sigtestを実行したウィンドウで、Ctrl＋Cキーを打鍵しても「Can't stop」と表示されるだけで、プログラムは終了しない。また、killコマンドで終了させようとしても、終了させることもできない。エディタ（編集プログラム）など、不用意に終了しては困るアプリケーションでは、このようにsignalを使って終了しないようにしているのである。

なお、9番のシグナルをsignalシステムコールでトラップすることはできず、9番のシグナルを送ると、I/Oを待っているプロセス以外は確実に消滅させることができる。

```
$ kill -9 プロセス番号
```

Perlスクリプト入門
すぐにできる実践Perl

構造化プログラミングとかオブジェクト指向という言葉がよくとりあげられますが、Perlスクリプトでも例外ではありません。思いつけばすぐ書けるというのもPerlのメリットですが、Perlでも書き方しだいで構造化スクリプトが書けるのです。

第7回　関数とサブルーチン

文：おもてじゅんいち／かざぐるま
Text : Junich Omote/Kazaguruma

これまでに出てきたprintやsplitなどは関数と呼ばれます。数学で習った「関数」とはちょっとイメージが違いますが、「コマンド」とか「命令」という感じで捉えてもらえばよいと思います。printやsplitなどは組み込み関数とも呼ばれ、Perlの機能として最初から用意されているものです。

同じ処理をするスクリプトをその都度何度も書くのは面倒ですし、なによりスクリプトがどんどん長くなってしまい見にくくなります。また、一度書いたスクリプトの部分的な機能を、別のスクリプトで利用したいときもあります。そんなときは自分で書いたスクリプトを再利用できるように、サブルーチンとして独立させることができます。「ルーチン」とはひとかたまりの処理を行うスクリプト（プログラム）です。

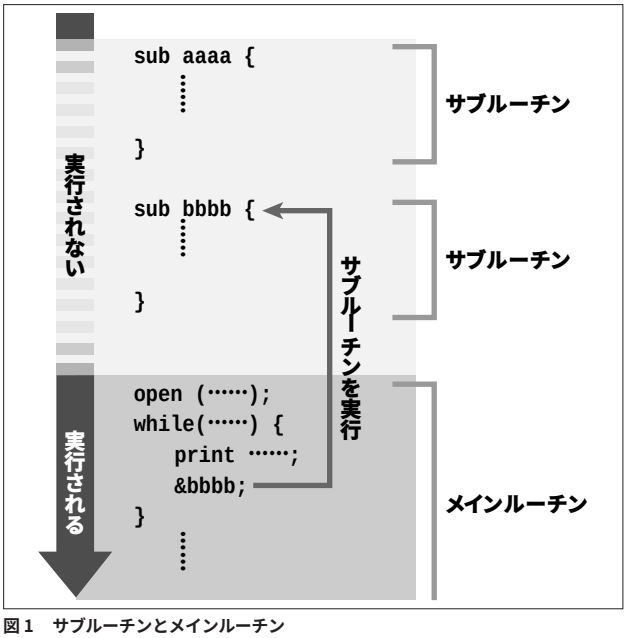


図1　サブルーチンとメインルーチン

サブルーチンに対してメインルーチンというものもあります。メインルーチンはこれまで皆さんが書いてきたPerlスクリプトそのものです。つまり、メインルーチンとはそのスクリプトが実行されるとき、1行目から順に読み込まれて処理される部分です。

一方、サブルーチンは同じスクリプト上に書かれていても、そのままでは実行されずにスキップされます（**図1**）。サブルーチンが実行されるのは、メインルーチン（またはサブルーチン）から呼び出されたときに限ります。

サブルーチンの定義

サブルーチンは、

```
sub （サブルーチン名） {
  ……
  ……
}
```

という書式で定義します。サブルーチン内には今までと同

様にスクリプトを書くことができます。サブルーチン名の付け方の制約は変数名と同じです。変数名として使用されているものと同じ名前を使ってもかまいません。それぞれ別物として区別されます。

サブルーチンはスクリプトのどこに書いてもかまいません。呼び出されるまでは無視されます。普通はスクリプトの最初か、最後にまとめて置きます。最初か最後かはその人のスタイルによるものなので、特に決まりはありません。ほかの言語の習慣がない人は最後にまとめるようにすればよいでしょう。

サブルーチンを呼び出す

サブルーチンはどのように使うのでしょうか。定義されたサブルーチンを呼び出して実行したいときは、

```
&(サブルーチン名);
```

と書きます。変数名は前に$を付けますが、サブルーチンの呼び出しは前に&を付けることを覚えておいてください。Perlバージョン5では、&を付けなくてもサブルーチンを呼び出すことができるようになりましたが、まぎらわしいので&を付けることをお勧めします。

サブルーチンの定義と呼び出し方法がわかったところで、**図2**を見てください。**図2**はサブルーチンを最初にまとめた例です。このスクリプトが実行されるとき、前半のサブルーチン部分は読み飛ばされます。サブルーチンが全部終わった後、open()の行からメインルーチンが実行されるのです。そして、そのメインルーチン内に、

```
&bbbb;
```

というサブルーチンの呼び出しがあるので、ここでやっとサブルーチンが実行されます（&aaaa;でも同様です）。

このとき、メインルーチン内で&bbbb;を何度も書いてあればどうでしょう。そのたびにサブルーチンbbbbが呼び出され、実行されることになります。つまり、サブルーチンとして一度スクリプトを書いておくだけで、あとはこのサブルーチンを呼び出す度に同じ処理が実行されるので、同じ処理のスクリプトを何度も書く必要がなくなるのです。

サブルーチンが呼び出されると、呼び出した側のメインルーチンは停止して、サブルーチンの処理が実行されます。サブルーチンの処理が終わると、メインルーチンのサブルーチンを呼び出した次の行から処理が再開されます。

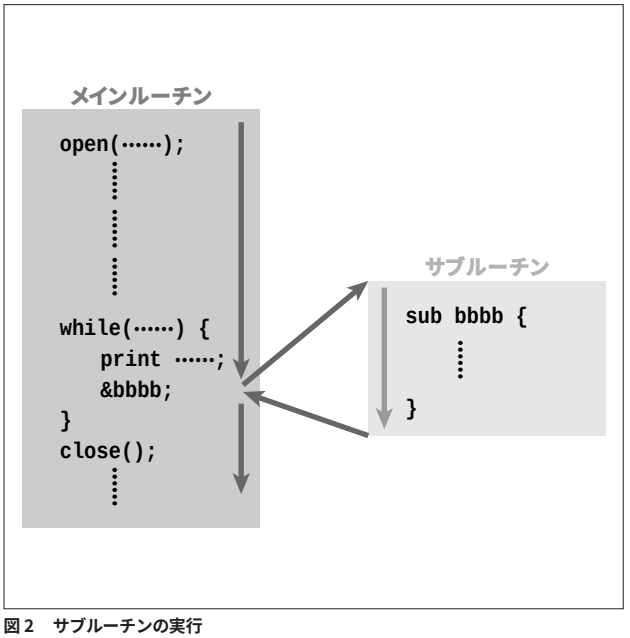


図2　サブルーチンの実行

サブルーチンの実際

では実際にサブルーチンを作ってみます。**リスト1**は「こんにちは」と出力する部分をサブルーチンにしてメインルーチンから呼び出し、実行しています。

短すぎてわかりにくいかもしれませんが、&greeting;の1行がメインルーチンになります。

これでもサブルーチンには違いないのですが、単に同じものを出力するだけではあまり利用価値はありません。

サブルーチンを実行する度に何らかの値を与えることによって、その場に応じた処理をさせたい場合は、組み込み関数と同じようにサブルーチンに引数を渡すことができます。

引数を渡してサブルーチンを呼び出す場合は、

```
&greeting("アスキー");
```

としたり、変数を使って、

▼リスト1　サブルーチンの例

```
&greeting;

sub greeting {
   print "こんにちは。¥n";
}
```

```
&greeting($name);
```

と書きます。複数の引数を並べることもできます。

```
&greeting($name,$age);
&greeting("アスキー","Linux","Perl");
```

引数を渡して呼び出すのはこのように簡単ですが、サブルーチン側で引数を受け取るのは少しやっかいです。

サブルーチンに渡された引数は、自動的に@_という配列に格納されます。上記の3個の引数の例だと、"アスキー"、"Linux"、"Perl"の順に配列@_に格納されますから、配列@_の各要素を取り出せば、サブルーチン内で引数を受け取ることができます。

ところで、配列@_の各要素はどのように取り出せるのでしょうか。記号だけでできている変数名なので、少しとまどうかもしれません。そこで、ちょっと復習です。

@listという配列があったとき、この配列の各要素を参照するためには、

```
$list[0]
$list[1]
$list[2]
  :
```

というように、$list[……]というスカラー変数にアクセスすることを思い出してください。つまり、配列変数を表す@を$に変えて、後ろに[ ]で括ったインデックス番号を付けるのでした。これを@_について行うと、配列@_の各要素は、

```
$_[0]
$_[1]
$_[2]
  :
```

という変数になるのです。ややこしい話ですが、この$_[0]、$_[1]……は、あくまでも@_の各要素であって、デフォルト変数$_とはまったく別のものです。混同しないようにしてください。

では、**リスト1**の例に引数の受け渡しを付け加えてみます。**リスト2**は名前を引数として渡しながら、サブルーチンを3回呼び出しています。実行結果は、

```
田中さん、こんにちは。
山本さん、こんにちは。
鈴木さん、こんにちは。
```

となります。

これで引数の受け渡しができるのですが、サブルーチン内での引数の扱いがすべて$_[ ]では、とても見やすいスクリプトとはいえなくなってしまいます。やはり、変数名はできるだけ意味を表すようにしたいものです。

引数が@_という配列に一括で格納されていることを利用すれば、とりあえずそれらを意味のある名前の変数に移し変えることができます。

たとえば、名前、住所、郵便番号の3つを引数として渡す場合、呼び出し側は、

```
&greeting("アスキー","東京都","151-8024");
```

となりますが、これをサブルーチン側では、

```
sub greeting {
  ($name,$addr,$tel) = @_;
    ……
    ……
}
```

として受け取ります。すると、配列@_に格納された引数はそれぞれ、$name、$addr、$telという変数にコピーされます。引数がいくつであっても、それを意味のある名前の変数にコピーして使えばわかりやすいスクリプトになるわけです。

グローバル変数とローカル変数

引数を受け取るために、$name、$addr、$telという変数を使いましたが、もしこれらの変数名がメインルーチンでも使われていたらちょっとまずいことになります。

▼リスト2　引数を渡す

```
&greeting("田中");
&greeting("山本");
&greeting("鈴木");

sub greeting {
   print "$_[0]さん、こんにちは。¥n";
}
```

リスト3を見てください。メインルーチンで使っている変数名と同じものをサブルーチン内で使いました。すると、サブルーチン呼び出し前の$numの値と呼び出し後の値とが変わっています。

つまりここでの$numは、メインルーチン、サブルーチンの区別なく、どこででも同一の変数にアクセスすることになります。このような変数をグローバル変数と呼びます。Perlの変数はデフォルトではすべてグローバル変数になります。

先ほどの例で、引数を意味のある名前の変数にコピーしましたが、その場合も、

```
sub greeting {
  ($name,$addr,$tel) = @_;
    ……
    ……
}
```

としたときは$name、$addr、$telはグローバル変数になります。もし同名の変数がメインルーチンにあると、メインルーチンで使っていた変数に引数がコピーされることになり、予期せぬ結果になります。

引数をコピーして使う場合、$name、$addr、$telなどの変数は、あくまでもサブルーチン内でのみ有効であればよく、たまたま同じ名前の変数をメインルーチンで使っていてもそれとは別のものとして扱えるほうが便利です。

このような、サブルーチン内でのみ有効な変数をローカル変数と呼びます。

ローカル変数はサブルーチン内で宣言されて使えるようになり、サブルーチンが終了すると自動的に破棄されます。同名の変数がメインルーチンにあっても、それとは別のものとして扱われます。

ローカル変数の宣言はmyを使って、

```
my($name,$addr,$tel);
```

▼リスト3　$numはグローバル変数

```
$num = 100;
print "$num ¥n";    # 結果：100
&aaaa;
print "$num ¥n";    # 結果：200

sub aaaa {
   $num = 200;
}
```

のように行います。ただし、myが使えるのはPerlバージョン5からで、Perlバージョン4の場合はlocalを使って、

```
local($name,$addr,$tel);
```

と書きます。myとlocalとはほとんど同じ演算子なのですが、サブルーチンからサブルーチンを呼び出したときに結果が異なります。

my演算子を使うと（**図3**）、宣言された変数はそのサブルーチン内でのみローカルになり、そこから呼び出されたサブルーチンには渡されません。呼び出されたサブルーチンで同名の変数を使っても、それはグローバル変数を使っていることになります。つまり、ローカル変数は宣言されたサブルーチン内でだけローカル扱いになります。

一方、local演算子の場合は（**図4**）、あるサブルーチンで宣言した変数はそのサブルーチンが終了するまでずっとローカル変数の値を保持するため、そこから別のサブルーチンが呼び出されても、呼び出されたサブルーチン（**図のサブルーチン2**）では宣言したサブルーチン（**図のサブルーチン1**）と同じローカル変数になります。

リスト4で実際に試してみましょう。サブルーチンaaa1のローカル変数をmyで宣言したときの出力結果は、

```
100
200
100
```

になりました。1行目はメインルーチンからの出力ですから、グローバル変数の値100、2行目はサブルーチンaaa1内で宣言されたローカル変数の値200です。3行目はサブルーチンaaa1から呼び出されたサブルーチンaaa2内の

▼リスト4　myとlocal

```
$num = 100;
print "$num ¥n";
&aaaa;

sub aaa1 {
   my($num);  ——— local($num);
   $num = 200;
   print "$num ¥n";
   &aaa2;
}
sub aaa2 {
   print "$num ¥n";
}
```

$numの値で、100と表示されています。つまり、サブルーチンaaa2ではグローバル変数にアクセスしていることがわかります。

次に、myをlocalに変えて実行してみましょう。結果は、

```
100
200
200
```

となりました。3行目の結果だけが違っています。これは、サブルーチンaaa2内でも、サブルーチンaaa1と同じローカル変数にアクセスしているためです。

どちらも一長一短であると思うのですが、やはりローカル変数というからには、サブルーチンという1つのブロック内でのみ有効であるというほうがよりシンプルでわかりやすいと思いますので、Perlバージョン5をお使いの方はできるだけmy演算子を使ってください。

ローカル変数と構造化

スクリプトやプログラムを構造化するということは、まずサブルーチンを独立した部品として組み立てていくことといえます。サブルーチンがメインルーチンや、ほかのサブルーチンから独立して動くように設計されていれば、そのサブルーチンを別のスクリプトで再利用することが可能になりますし、スクリプトをデバッグ（不良箇所を修正する）する場合にも、サブルーチン単位で確認することができて効率的です。

サブルーチンを独立して動くようにするためには、ローカル変数が重要な意味を持ちます。つまり、そのサブルーチンがグローバル変数を使っていると、そのグローバル変数の状況によって、メインルーチンやほかのサブルーチンの影響を受けることになり、独立しないことになるからです。ですから、サブルーチンを作るときはできるだけローカル変数のみを使うようにして、グローバル変数にアクセスしないようにしてください。

外部（メインルーチンやほかのサブルーチン）とサブルーチンとのデータのやりとりは、必ず引数を使って行い、受け取った引数はローカル変数に格納するようにします。

my(local)演算子でローカル変数を宣言しながら、引数を受け取るには、

```
my($name,$addr,$tel) = @_;
```

のように、宣言と引数のコピーを同時に行います。その後、サブルーチン内でのみ使うローカル変数を宣言します。

```
my($val,$num);
```

ですから、引数を必要とするサブルーチンの基本形は、

```
sub aaaa {
  my($name,$addr,$tel) = @_;
  my($val,$num);
```

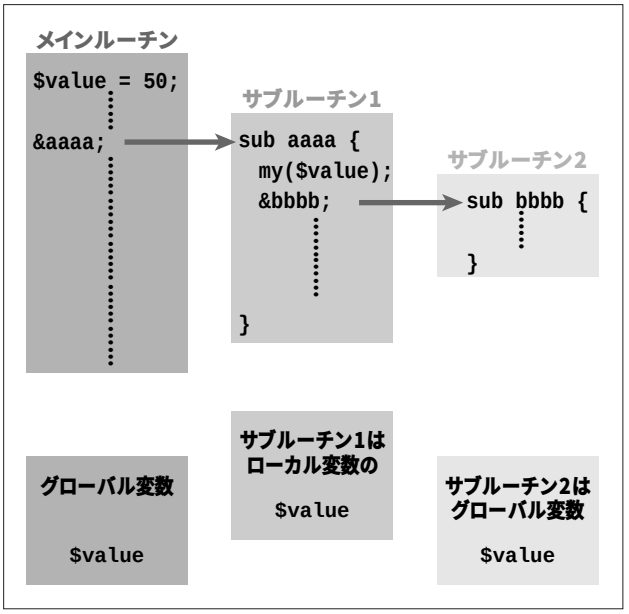


図3　my宣言したローカル変数

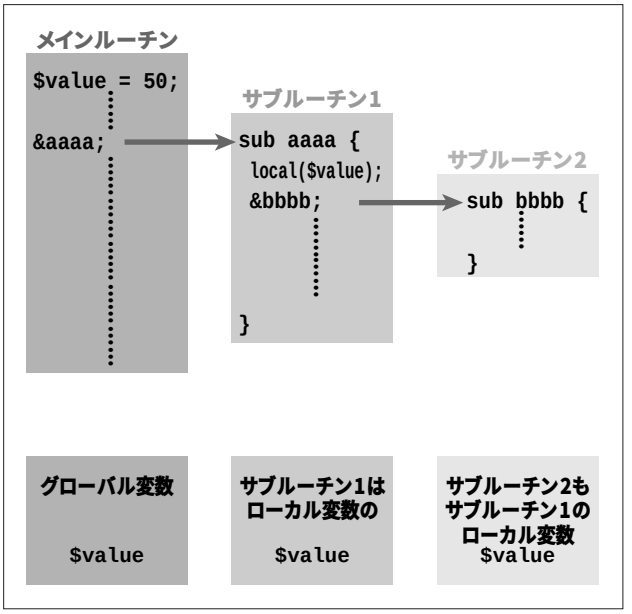


図4　local宣言したローカル変数

```
  ……
  ……
}
```

のようになると覚えておきましょう。

サブルーチンの戻り値

組み込み関数のように、サブルーチンでも、

```
$ret = &aaaa(……);
```

というように、実行結果の値を呼び出し側に返すことができます。サブルーチン内に、

```
sub aaaa {
  ……
  ……
  return($result);
}
```

と書くと、$resultの値が返されます。つまり、

```
$ret = &double(100);

sub double{
  my($num) = @_;
  $result = $num * 2;
  return($result);
}
```

であれば、メインルーチンの$retには200が返ります。

return()は、サブルーチンの途中でサブルーチンから抜けるためにも使われるもので、

```
$ret = &double(100);

sub double{
  my($num) = @_;
  if ($num == 0)  { return(0); }
  $result = $num * 2;
  return($result);
}
```

ということもできます。また、サブルーチンの最後に限ってはreturnを使わなくても、「最後に評価された式の値を返す」という決まりがあるので、

```
$ret = &double(100);

sub double{
  my($num) = @_;
  if ($num == 0)  { return(0); }
  $result = $num * 2;
}
```

とreturnを書かなくても、最後の式の結果がサブルーチンの戻り値として返されます。ただし、サブルーチンの途中でサブルーチンを抜けるにはreturnが必要です。

配列を引数にする場合の注意

引数は、すべてまとめて配列としてサブルーチンに渡されます。そのため、配列を引数としたいときに注意しなければならないことがあります。たとえば、

```
&aaaa(@list,$num);
```

というように、配列@listとスカラー変数$numを引数として渡す場合、サブルーチン側で、

```
sub aaaa {
  my(@list,$num) = @_;
  ……
  ……
}
```

としてしまうと、$numの値がうまく渡せません。なぜなら、リストと配列のところで学んだように、左辺にある配列変数（サブルーチン内のmyの部分）に、リスト値（@_の中身）を代入する場合は、左辺の配列変数に右辺の残りのリストデータすべてが格納されてしまうからです。つまり、ここではメインルーチン側の@listの中身だけでなく、$numの値までもがmy(@list,$num)の@listに格納されてしまって、my(@list,$num)の$numには何も格納されないことになってしまいます。

引数を受け渡しする場合には、配列を最後に置くように

注意してください。先の例は、

```
&aaaa($num,@list);

sub aaaa {
  my($num,@list) = @_;
  ……
  ……
}
```

とすれば、思い通りの結果になります。

再帰呼び出し

サブルーチンは通常、メインルーチンやほかのサブルーチンから呼び出されます。しかし、あるサブルーチンが自分自身を呼び出すこともできます。このような呼び出し方を再帰呼び出しといいます。

たとえば、家系図を思い浮かべてください。家系図はどの部分をとっても「枝分かれ」という処理でできています。そして、枝分かれでできた「子」にあたる部分が、次の世代には「親」となって、またそこから枝分かれします。しかも、最初から全体の枝分かれ回数が決まっているわけではありません。それぞれの枝で、もう枝分かれしないという状態になったときに、枝分かれがストップするだけです。

つまり全体を総じて、何回処理するとか、どこまで処理するとかは決められず、部分部分の処理が終わっていくのを待つしかないような処理に再帰呼び出しが使われます。

身近な例題として、ディレクトリのツリー表示を考えます。ディレクトリの構造は、ルートから見たときに、いったい何レベルのディレクトリ構造になっているかは決められませんし、それぞれに枝分かれしていったディレクトリごとにその深さが違っています。

ここでは、/home/httpdディレクトリ以下の全ファイルを、ディレクトリ構造に沿って表示して行くスクリプトを作ります。Perlの文法としては、これまで学んだことだけで十分ですから、皆さんもまずは自分で考えてみてください。結果は**画面**のようになればOKです（表示内容は各自のディレクトリ内容によります）。

▼リスト5　ディレクトリツリー①

```
opendir (DIRH,"/home/httpd");
@dir = readdir(DIRH);
closedir(DIRH);

foreach $d (@dir) {
   if ( $d eq "." || $d eq ".."  ) {
   next;
   }
  print "$d¥n";
}
```

ディレクトリツリー

まずは、1つのディレクトリの内容を表示するスクリプトです（**リスト5**）。これが基本ですね。ディレクトリをオープンして、その内容を一気に@dirに読み込み、「.」と「..」以外を表示しています。

さて、ある1つのディレクトリの内容を表示する処理は、ディレクトリツリーを作る場合に何度も実行する必要がありますから、これをサブルーチン化します。

リスト6を見てください。せっかくサブルーチンにするのですから、どのディレクトリを表示するかという、ディレクトリの指定は引数で行います。つまり、メインルーチンから、

```
&dirlist("/home/httpd");
```

という具合に呼び出せるようにします。この引数は、$pathという名前で受け取ります。サブルーチン内で使う、$dと@dirはローカル変数として宣言しておきます。

後の処理はメインルーチンのときと同じです。

さてこれで、ディレクトリの内容表示処理をサブルーチ

```
cgi-bin
        entry.cgi
        list.cgi
html
        entry.html
        form.html
        index.html
        guide
                f-content.html
                s-content.html
                start.html
        image
                title.gif
                button.gif
icons
        README
        alert.black.gif
        alert.red.gif
           :
```

画面　ディレクトリツリー表示結果

ン化することができました。試しに実行してみると、リスト5と同じ結果になります。メインルーチンの処理をサブルーチン化しただけですから当然ですね。

次に、ディレクトリのレベルが1つ下に移ったときに、表示上でもインデントしてレベルを確認できるようにします（**画面**参照）。

リスト7では、レベル設定用の引数を用意しています。サブルーチンを呼び出したときに指定したレベルの数値分だけ、ファイル名を表示するときにタブ文字を加えてインデントしています。

```
print "¥t" x $level . "$d¥n";
```

▼リスト6　ディレクトリツリー②

```
&dirlist("/home/httpd");

##### Sub Routine #####
sub dirlist {
   my($path) = @_;
   my($d,@dir);

   opendir (DIRH,$path);
   @dir = readdir(DIRH);
   closedir(DIRH);

   foreach $d (@dir) {
      if ( $d eq "." || $d eq ".."  ) {
         next;
      }
      print "$d¥n";
   }
}
```

▼リスト7　ディレクトリツリー③

```
&dirlist("/home/httpd",1);

##### Sub Routine #####
sub dirlist {
   my($path,$level) = @_;
   my($d,@dir);

   opendir (DIRH,$path);
   @dir = readdir(DIRH);
   closedir(DIRH);

   foreach $d (@dir) {
      if ( $d eq "." || $d eq ".."  ) {
         next;
      }
      print "¥t" x $level . "$d¥n";
   }
}
```

の部分をよく見てください。$levelの分だけタブ文字が追加されます。よくわからない方は、文字列処理のところを復習してください。

さてこれで準備は整いました。気をつけるべき点は、サブルーチンがローカル変数だけで動くようになっているかということと、サブルーチン内のループが無限ループになる可能性がないかです。この2つが満たされていれば、再帰呼び出しといってもそれほど難しいものではありません。

では、**リスト7**のディレクトリ内容表示スクリプトをディレクトリツリー表示にするには、どうすればよいのでしょうか。ようは、あるディレクトリの内容を表示するときに、その内容の中にディレクトリがあった場合、その内容を表示するということを次々と繰り返していけばよいのです。このあたりはじっくりと頭の中で考えてみてください。

ディレクトリの内容の中にディレクトリがあった場合、その内容を表示するということは、「スクリプト上では配列@dirの各要素を1つずつ見ていって、それがディレクトリであったなら、今度はそのディレクトリを引数としてこのサブルーチン（自分自身）を呼び出して実行し、レベルを1つ増やす」という処理になります。

スクリプトの修正は、**リスト8**の網掛け部分を追加するだけです。簡単ですね。実は、ここでdirlistサブルーチン、すなわち自分自身を呼び出しているのです。呼び出したサブルーチンが終了すれば、またここに戻ってきますから、残りのディレクトリ内容を表示します。

▼リスト8　ディレクトリツリー④

```
&dirlist("/home/httpd",0);

##### Sub Routine #####
sub dirlist {
   my($path,$level) = @_;
   my($d,@dir);

   opendir(DIRH,$path);
   @dir = readdir(DIRH);
   closedir(DIRH);

   foreach $d (@dir) {
      if ( $d eq "." || $d eq ".."  ) {
         next;
      }
      print "¥t" x $level . "$d¥n";

      if ( -d "$path/$d" ) {
         &dirlist("$path/$d",$level+1);
      }
   }
}
```

[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、コマンドラインの処理を二度繰り返すコマンドevalの使い方や、シェル変数の内容を別の変数の名前として使う「間接展開」などの応用例について説明したあと、初期値・終了値・増分値を指定して処理を繰り返すシェル関数を作成する。

第11回　evalで二度おいしいコマンドライン

文：大池浩一
Text:Koichi Oike

あなたがシェルのプロンプト（「$」や「#」など）に対してキー入力したり、シェルスクリプトの一部として記述したコマンドラインには、行頭に指定したコマンドが実行されるのに先立って、シェルによるさまざまな処理が段階的に加えられる。

具体的には、パイプやリダイレクトなどを使った入出力の切り替え、エイリアス（別名）の展開、「~」をホームディレクトリに置換するといったチルダ展開、「$変数名」をシェル変数の内容に置換する変数展開、両端が「`」で囲まれた内容をその実行結果で置き換えるコマンド置換、「*」や「?」などのワイルドカードを含んだワードを実際のファイル名の並びに置換するパス名展開などだ。こうした処理により、コマンドが実行される段階でのコマンドラインは、あなたが入力（あるいはシェルスクリプトに記述）した文字列とはかなり異なった内容になる。

このような処理は、1つのコマンドラインに対して一度だけ行われるのが普通だ。コマンドラインを「'」や「"」で囲むクォーティングにより、上記の処理の一部をスキップすることも可能だが、処理が一度だけ行われる点に違いはない。

ところが、bashを含むBourne系シェルには、こうしたコマンドライン処理を二度繰り返す組み込みコマンドevalが用意されている。今回は、このevalを利用して、実行時にコマンドラインの内容が動的に変化するようなシェルスクリプトを作成してみることにしよう。

evalを使って内容が動的変化するスクリプトを作成

正確に言うと、evalは「引数で指定した内容をコマンドラインとみなしてして実行する」組み込みコマンドだ。変数展開などのコマンドライン処理は、eval自身を実行する際にまず1回、引数の内容をコマンドラインとみなして実行する際にもう1回行われる。結果的に、コマンドライン処理が二度行われることになるわけだ。

この2回のコマンドライン処理を活用すると、「シェル変数に格納された内容を変数名とみなして、その内容を参照する」とか、「パイプなどを含むコマンドラインを状況に応じて変化させる」といった、通常のシェルスクリプトでは実現が難しいと思える処理を、たった1行の記述で実現できる。

以下では、

- evalの記述のしかた
- シェル変数の内容を変数名として使う「間接展開」
- シェル変数の内容でコマンドラインを変化させる方法
- シェル関数の配列を擬似的に実現する方法

などについて説明したのち、C言語などのforループに似た、指定された処理を一定回数繰り返すシェル関数「loop」をevalを利用して構築する。

●evalの引数にコマンドを指定する

evalの書式は、それを使って実現可能な処理の複雑さとは対照的に、以下のように単純なものだ。

```
eval 実行したいコマンドライン
```

たとえば、

```
$ eval ls -l
```

とすると、最初にeval自身が実行され、その結果として引数の「ls -l」が実行されて、カレントディレクトリの全ファイルの詳細情報が一覧表示される。このとき得られる結果は、evalを使わないで、

```
$ ls -l
```

を直接実行した場合とまったく同じだ。このように、変数展開などの処理を含まないコマンドラインに対しては、evalを使う必然性はない。

次に、変数展開を含んだ例をお見せしよう。シェル変数fooに「bar」が格納されている状態で、

```
$ eval ls -l $foo
```

とすると、evalを実行する際のコマンドライン処理で$fooが変数展開されて「bar」に置換される。その後、引数の「ls -l bar」が実行されて、ファイルbarに関する詳細情報が表示される。こちらの例も、得られる結果は直接実行した場合と同じだが、lsのコマンドライン処理より前の時点で$fooが変数展開されている点に注意されたい。

実際のところ、evalを利用する必要があるのは、

(a)変数展開などの同じ処理を二度繰り返す
(b)変数展開のあとにチルダ展開やパイプの処理が必要

など、通常の処理の流れでは対応できない場合だけだ。以下では、それぞれの代表的な例を挙げながら、evalを使った対処方法について説明しよう。

この節での要点をまとめると、

・evalの引数に実行したいコマンドラインを記述する。

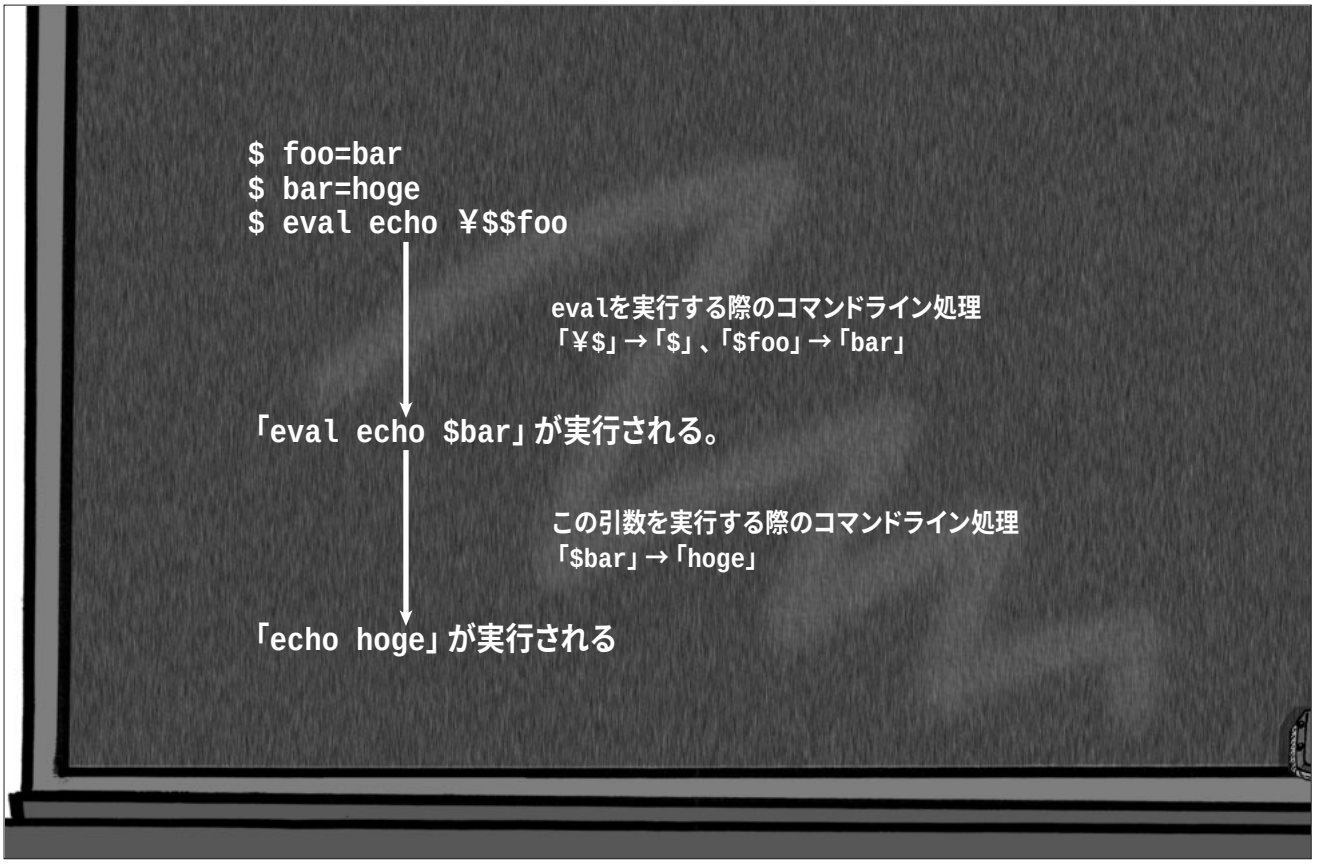


図　シェル変数barの内容をシェル変数fooのみを使って取り出す

・evalを利用すると便利なのは、通常のコマンドライン処理の流れでは対応できない場合だけ。

ということになる。

●シェル変数の内容を変数名として使うには

まずは、シェル変数の内容を変数名とみなして、その内容を取り出す方法を考えてみよう。こうしたテクニックを変数の「間接展開」と呼ぶ。bashのバージョン2以降では、「${!変数名}」という書式で間接展開が可能だが、以下ではこの書式を使わずに間接展開を実現する。

たとえば、シェル変数fooに「bar」、シェル変数barに「hoge」がそれぞれ格納されているとする。このとき、シェル変数barの内容「hoge」を、シェル変数fooだけを使った表現で取り出すにはどうすればいいだろうか。

シェル変数の内容を取り出す変数展開には、ご存じのように「$変数名」という書式を利用する。たとえば、シェル変数fooの内容を出力するには、「$foo」を使って、

```
$ echo $foo
```

とすればいい。$fooが「bar」に展開された後、echoによって端末画面に表示される。

ここで問題になるのは、変数展開後の文字列「bar」を、どのようにしてシェル変数名としてシェルに扱ってもらうかということだ。

単に、$記号を重ねて、

```
$ echo $$foo
```

とすると、「397foo」のように何桁かの数字（端末画面ごとに異なる）と文字列「foo」がそのまま表示される。実は、「$」はシェルのプロセスIDを表す組み込み変数（「$$」で参照のみ可能）なので、「$$」と書くとシェルのプロセスIDに置換されてしまうのだ。

そこで、最初の「$」がコマンドライン処理で解釈されないようにエスケープして（「¥」を前に付けて）、

```
$ echo ¥$$foo
```

とすると、今度は「$bar」と表示されてしまう。つまり、$barの変数展開は行われない。

実際のところ、1回だけのコマンドラインの処理では、ここで取り上げている間接展開は実現できない。最初の変数が展開された時点で変数展開処理は終了し、次の段階の処理に移るからだ。

この問題を解決するには、evalを使ってコマンドラインの処理を二度繰り返せばいい。それぞれの処理に含まれる変数展開処理を利用するのだ。具体的には、

```
$ eval echo ¥$$foo
```

とすればいい。

まず、eval自身が実行される際のコマンドライン処理では、「¥」でエスケープした「$」は処理されずに残り（「¥」自体は消える）、その後ろの「$foo」が変数展開で「bar」に置換される。この段階でevalの引数は「echo $bar」に変化する。続いて、この引数が実行される際のコマンドライン処理で、「$bar」が変数展開により「hoge」に置換される。その結果、「echo hoge」が実行されて、端末画面に「hoge」が表示される（**図**）。

次に、先月号の本連載で取り上げたスクリプトselargについて補足しよう。これは、コマンドラインで指定した選択肢を一覧表示して、キー入力した数字に対応する選択肢を表示するスクリプトだ（リストは先月号の211ページを参照）。

たとえば、

```
$ selarg hoge fuga hogehoge
```

とすると、

```
1) hoge
2) fuga
3) hogehoge
選択は? (1-3)
```

という形式で選択肢の一覧が番号付きで表示され、いずれかの番号（たとえば2）を入力すると、「選択したのはfugaです。」などと表示される。

このスクリプトの全体的な解説は先月号のこの記事を参照していただきたいが、今回は、evalコマンドを使った14行目、

```
eval "ans=¥$$reply"
```

に着目する。キー入力した番号に対応する選択肢をシェル変数ansに格納する部分だ。

シェル変数replyには、実行時にキー入力した番号が格納されている（ここでは仮に「2」としよう）。まず、eval自身のコマンドライン処理により$replyが変数展開され、evalの引数は「ans=$2」に変わる。続いて、この引数が実行される際のコマンドライン処理によって、「$2」が2番目の引数「fuga」に置換され、最終的に「ans=fuga」が実行されてシェル変数ansに「fuga」が格納される。

こうした処理をevalを使わずに実現するのは意外に面倒だ。たとえば、case制御構造を使うとしても、

```
case "$reply" in
1) ans=$1;;
2) ans=$2;;
3) ans=$3;;
    ⋮
(中略)
    ⋮
esac
```

となり、引数の数が増えるにつれてスクリプトが長くなってしまう。これに対し、evalを利用すればたった1行の記述でOKだし、引数の数にも影響されない。

以上、まとめると、

- **evalを使うと、シェル変数の内容を変数名として、その内容を取り出すことができる（間接展開）。**
- **具体的には、evalの引数のコマンドラインに「￥$$変数名」を記述すればいい。**

ということになる。

●コマンドライン処理の順番を考慮しよう

冒頭でも簡単に説明したが、bashを含むBourne系シェルのコマンドラインでは、コマンドを実行する際に以下のような処理が順番に行われている。

(1)トークンに分解

空白文字（スペース、タブ、改行）や記号（<、>、|、&、;、カッコ）を区切りとして、コマンドラインをトークンに分割する。なお、先頭のワードがifなどのキーワードの場合には複合コマンドとして特別な処理が行われる。

(2)エイリアス処理

エイリアス（別名）の展開。最初のワードがエイリアスの場合は定義内容と置換して(1)に戻る。

(3)ブレース展開

「{」「}」（ブレース）を使った展開。たとえば、「hoge.{cpp,h}」が「hoge.cpp hoge.h」になる。

(4)チルダ展開

「~」（チルダ）や「~ユーザー名」をユーザーのホームディレクトリに置換する。

(5)変数展開（パラメータ展開）

$で始まる式に対する展開。たとえば「$変数名」は対応するシェル変数の内容に置換される。

(6)コマンド置換

両端が「`」で囲まれた部分をサブシェルで実行し、実行結果で置換する。bashでは「$(…)」も利用可能。

(7)数値置換

「$((…))」の内容を数値演算式とみなして、評価結果で置換する（bashのみ利用可能）。

(8)ワードの抽出

環境変数IFSの内容（初期値は空白文字）を区切り文字としてコマンドラインをワードに分割する。

(9)パス名展開

「＊」や「？」などのワイルドカードを含む文字列に対して、マッチするファイル名の並びに展開する。

(10)コマンドの検索

先頭のワードを関数、組み込みコマンド、外部コマンドの順で検索する。

(11)コマンドの実行

リダイレクトやパイプによる入出力の切り替えを設定後にコマンドを実行する。

なお、両端が「'」で囲まれた部分は(2)から(9)の処理がすべてスキップされ、「"」で囲まれた部分は(2)から(4)と(8)(9)の処理がスキップされる。

こうしたコマンドライン処理が行われる順番に注意しないと、複数の展開処理を組み合わせようとして意図しない結果を招くことになる。

たとえば、あるユーザーdaichiのホームディレクトリが「/home/daichi」で、使用中のシェルの変数fooの内容が「~/bar」だとしよう。このシェルで、

```
$ echo $foo
```

としても、「~/bar」がそのまま出力される。(5)の段階で展開したシェル変数の内容に「~」が含まれていても、チルダ展開はその前の(4)で終わっているので、ホームディレクトリには展開されないのだ。

ところが、evalを利用して、

```
$ eval echo $foo
```

とすると、「/home/daichi/bar」とチルダ展開が行われた状態の出力が得られる。

まず、eval自身のコマンドライン処理で、「$foo」が変数展開により「~/bar」に置換され、evalの引数は「echo ~/bar」に変わる。続いて、この引数が実行される際のコマンドライン処理で、「~」がチルダ展開により「/home/daichi」に置換され、最終的には「echo /home/daichi/bar」が実行されるのだ。

このほか、シェル変数の内容にリダイレクト「>」「<」やパイプ「|」が含まれている場合にも、似たような問題が発生する。たとえば、実行したいコマンドラインの内容を格納したシェル変数cmdを、

```
$ $cmd
```

として実行することを考えてみよう。

cmdの内容が「ls -l」のようなものなら、何も問題はない。ところが、「ls -l > hoge」や「ls | wc -l」などのようにリダイレクトやパイプを含んでいると、「そのようなファイルやディレクトリはありません」というエラーメッセージが表示される。リダイレクトやパイプの処理は、コマンドライン処理の最初の段階(1)で行われるため、変数展開した内容に含まれる「>」や「|」は、単なるファイルとして扱われてしまうのだ。

解決策はもちろん、evalを利用して、

```
$ eval $cmd
```

とすることだ。eval自身のコマンドライン処理の段階で$cmdが実際の内容に置換されたあと、引数のコマンドライン処理の段階で、変数の内容に含まれるリダイレクトやパイプが正しく扱われる。

また、コマンドラインの全体ではなく、その一部だけを変化させたいという場合もある。たとえば、引数なしの場合は「ls -t」を実行するが、引数が指定されたら「ls -t | head -$1」を実行するスクリプトnewfileを考えよう。lsの-tオプションは、ファイルを更新時刻の新しい順に並び替えるオプションなので、

```
$ newfile
```

とした場合にはカレントディレクトリの全ファイルが新しい順に表示され、

```
$ newfile 5
```

とすると、最新の5つのファイルだけが表示される。

if制御構文を使ったバージョンのnewfileは以下のように書ける。

```
#!/bin/sh
if [ -n "$1" ]; then
  ls -t | head -$1
else
  ls -t
fi
```

このスクリプトでは、条件式「-n "$1"」によって分岐処理を行う。最初の引数が指定された場合は「ls -lt | head -$1」、引数が指定されなかった場合は「ls -lt」がそれぞれ実行される。

これに対し、evalを利用すれば、もっと短い（実質は1行の）スクリプトで同じ処理を行える。

```
#!/bin/sh
eval ls -t ${1:+"| head -¥$1"}
```

まず、「${変数名：＋置換文字列}」という表現は、変数の内容が1文字以上あれば置換文字列、空文字列の場合は空文字列に置換されることを押さえておこう。

スクリプト実行時に引数を指定した場合、eval自身のコマンドライン処理により、evalの引数は「ls -lt | head -$1」に変わり、引数のコマンドライン処理では、$1が実際の引数に変数展開され、lsとheadが実行される。

なお、「$1」の前に「¥」を置いてエスケープしているのは、引数のコマンドライン処理まで$1の変数展開を遅らせるためだ。もし、$1の内容に「>」や「&」などシェルにとって特殊な意味を持つ記号が含まれていたとしても、

二度目のコマンドライン処理まで変数展開を遅らせることで、思わぬ事態の発生を抑えることができる。

一方、引数を指定しなかった場合は、eval自身のコマンドライン処理により、evalの引数は「ls -lt」に変わる。引数のコマンドライン処理では展開処理は行われず、結局lsだけが実行される。

この節での要点をまとめると、

- **コマンドラインの処理は、さまざまな展開・置換処理が順番に一度だけ行われる。**
- **コマンドライン処理の順番を直接変えることはできないので、evalを利用して二度処理を繰り返し、異なる順番での処理を実現する。**

ということになる。

●evalを使ってシェル変数の擬似配列を実現する

Bourne系シェルには、foo[0]、foo[1]、foo[2]、…といった配列形式、すなわち複数の要素を数値（インデックス）で制御する仕組みが用意されていないため、多数の情報を効率よく管理することが難しい。なお、bashのバージョン2以降には、シェル変数の配列機能が用意されている。これについては次回取り上げることにしよう。

配列の代わりとして広く用いられているのは、空白文字や「:」（コロン）などを区切り文字として、複数の情報を1つのシェル変数にまとめて格納する方法だ。たとえば、コマンドの実行結果をコマンド置換を使ってシェル変数に格納したり、環境変数PATHに複数のディレクトリを格納する場合などに使われている。この方法の欠点は、情報の一部に区切り文字が含まれている場合、その情報を正しく扱えないか、扱うのがとても面倒なことだ。

evalを利用すれば、配列によく似た仕組みを実現して、複数のシェル変数に情報を分散して格納できる。これを「擬似配列」と呼ぶことにしよう。アイデアはごく単純で、foo[0]のかわりに「foo_0」、foo[1]の代わりに「foo_1」といった具合に、インデックスの値を変数名自体に含んだシェル変数を用意し、それぞれの要素の内容を格納するというものだ。インデックスの値を含むシェル変数名はevalを利用した間接展開で生成する。

擬似配列を使ったスクリプトの例として、標準入力から読み込んだ行を、line_0、line_1、line_2、…という擬似配列に記憶して、標準出力に逆順に出力するスクリプトmytacを**リスト1**に示す。

たとえば、

```
$ mytac < hoge
```

とすると、ファイルhogeの内容が、最終行から先頭行まで逆順に表示される。

スクリプトの内容を見ていこう。2行目では、インデックスの値を格納するシェル変数idxに、初期値として「0」を格納している。

3行目では、環境変数IFSに空文字列を設定している。これは、標準入力をreadで読み込んでシェル変数に設定する際に、ワード分割を行わないようにするためだ。IFSが初期値（空白文字）のままだと、ワード分割によって行頭のスペースやタブが失われてしまう。

4～6行目のwhile制御構造では、標準入力の内容をreadで1行ずつ読み込み、擬似配列に格納する。whileの条件部「eval read line_$idx」では、eval自身のコマンドライン処理において、「$idx」が現在のインデックスの値（たとえば「0」）に置換され、evalの引数は「read line_0」に変わる。続いて、この引数が実行されて、シェル変数line_0に標準入力の内容が1行格納される。

5行目では、インデックスの値を1ずつ増やす処理をコマンド置換とexprを組み合わせて行っている。bashのみ使うなら、「idx=$((idx＋1))」としてもいい。

こうして、標準入力の内容は1行ずつ擬似配列line_0、line_1、line_2、…に格納される。標準入力をファイルにリダイレクトした場合はそのファイルの終わり、キーボードのままならCtrl-Dキーの入力でreadが異常終了し、while制御構造による繰り返しから抜ける。

後半の7～10行目では、while制御構造を利用して、擬似配列に格納した内容を、インデックスの値が0になるまで逆順に出力する。8行目でインデックスの値を1ずつ減らしながら、9行目のevalで擬似配列の変数名を生成し、

▼リスト1　標準入力を逆順に出力するスクリプトmytac

```
 1: #!/bin/sh
 2: idx=0
 3: IFS=''
 4: while eval read line_$idx; do
 5:   idx=`expr $idx + 1`
 6: done
 7: while [ $idx -gt 0 ]; do
 8:   idx=`expr $idx - 1`
 9:   eval echo \$line_$idx
10: done
```

その内容をechoで出力している。たとえば、インデックスの値が0の場合、eval自身のコマンドライン処理で引数が「echo $line_0」に変わり、この引数が実行されてline_0の内容が出力されるのだ。

以上、まとめると、

- **evalによるシェル変数の間接展開を利用して、配列に似た仕組み（擬似配列）を実現できる。**
- **インデックスの値を保持するシェル変数を使って、インデックスの値を変数名に含むシェル変数を作成する。**

ということになる。

今月のスクリプト

後半は、与えられたテーマを実現するスクリプトの作成手順を説明する。今月は、

- **指定した回数だけ処理を繰り返すシェル関数「loop」**

を作成する。

これは、C言語などのforループによく似た繰り返し処理を行うものだ。evalを利用して、ループの制御に使われるシェル変数を引数で指定し、繰り返す回数（または初期値・終了値・増分値）を指定できるようにする。

●指定した回数処理を繰り返すには

シェルスクリプトに用意されているfor制御構文は、複数のファイルに同じ処理を加えるといった目的には便利だが、C言語などのforループのように一定回数処理を繰り返す目的には向いていない。

手始めに作成するプロトタイプ版のシェル関数loopでは、最初の引数に繰り返しを制御するループ変数名、2番目の引数に繰り返す回数、3番目の引数に繰り返す内容（コマンドライン）を指定する。たとえば、

```
$ loop i 100 'echo i=$i'
```

とすると、ループ変数iの値が1から100まで更新されつつ「echo i=$i」が実行され、「i=1」「i=2」…「i=100」の計100行が出力される。

繰り返す内容は、3番めの引数でまとめてloopに渡すため、両端を「 ’」で囲む（クォーティングする）必要がある。「 ’」で囲んだ内部には、「$変数名」やコマンド置換、「｜」（パイプ）などを記述できる。また、複数のコマンドを「；」で区切って並べることも可能だ。

リスト2はプロトタイプの関数loopだ。内容を詳しく説明していこう。まず、3行目では、evalを利用した間接展開により、最初の引数（$1で参照）で指定された名前のシェル変数に「1」を設定している。たとえば、最初の引数に「i」を指定した場合、eval自身のコマンド処理の段階でevalの引数が「i=1」に変わり、この引数が実行されることでシェル変数iに1が設定されるわけだ。

4～8行目のwhile制御構造では、2番目の引数（$2で参照）で指定した回数だけ、do～done間の処理を繰り返す。まず、4行目のwhileの条件部で、ループ変数の値が2番目の引数で指定した値以下かどうかを判断し、条件が成立（真）の場合は処理を続行する。ここでは、ループ変数を間接展開するためにevalが必要だ。

5行目では、3番目の引数（$3で参照）で指定した内容をコマンドラインとして実行する。引数に含まれる変数の展開処理やパイプなどを正常に扱うために、evalを利用して2回コマンドライン処理を繰り返す必要がある。

6、7行目はループ変数の値を1ずつ増やす処理だ。いったんループ変数の値をシェル変数loop_iにコピーし、コマンド置換の内部ではloop_iを利用してループ変数の値を1増やしている。わざわざこんな面倒なことをする理由は、コマンド置換で指定されたコマンドが、関数loopを処理中のシェルとは別のサブシェルで実行されるためだ。ほかの部分のように間接展開でループ変数の値を参照すると、サブシェルには該当する名前の変数が存在しないため、exprがエラーになってしまうのだ。

●初期値や終了値を指定できるように改良する

リスト2のプロトタイプを叩き台として、ループ変数の「初期値」と「終了値」を指定して繰り返し処理を行えるように改良しよう。

▼リスト2　シェル関数loop（プロトタイプ）

```
1: function loop
2: {
3:   eval $1=1
4:   while eval [ ¥$$1 -le $2 ]; do
5:     eval $3
6:     eval loop_i=¥$$1
7:     eval $1=`expr $loop_i + 1`
8:   done
9: }
```

たとえば、

```
$ loop i 10 20 'echo i=$i'
```

とするとループ変数iが10から20まで、「i=10」「i=11」…「i=20」の計11行が表示されるようにする。

さらに、ループ変数の増分値として1以外の値も指定できるようにしよう。

たとえば、

```
$ loop i 0 10 5 'echo i=$i'
```

とすると、ループ変数iが5ずつ増加して、「i=0」「i=5」「i=10」の計3行が表示されるようにする。

まとめると、回数だけ、初期値と終了値、初期値・終了値・増分値という3通りの指定方法に対応する必要がある。この3つは、コマンドライン引数の数（$#で参照）で区別すればよいだろう。

▼リスト3　シェル関数loop（完成版）

```
 1: function loop
 2: {
 3: # local loop_start loop_end loop_step loop_op
 4:   case $# in
 5:   3) loop_start=1
 6:      loop_end=$2
 7:      loop_step=1 ;;
 8:   4) loop_start=$2
 9:      loop_end=$3
10:      loop_step=1 ;;
11:   5) loop_start=$2
12:      loop_end=$3
13:      loop_step=$4 ;;
14:   *) echo -e "usage: loop VarName [Start] End
   [Step] Command" > /dev/stderr
15:      return 1
16:   esac
17:   loop_op=-le
18:   if [ $loop_start -gt $loop_end ]; then
19:     loop_op=-ge
20:     if [ $loop_step -gt 0 ]; then
21:       loop_step=`expr -$loop_step`
22:     fi
23:   fi
24:   eval $1=$loop_start
25:   while eval [ ¥$$1 $loop_op $loop_end ]; do
26:     eval eval ¥$$#
27:     eval loop_i=¥$$1
28:     eval $1=`expr $loop_i + $loop_step`
29:   done
30: }
```

このほか、初期値より終了値のほうが小さい場合の処理や、引数を指定しなかった場合のエラー処理などを追加した完成版のシェル関数loopが**リスト3**だ。

それでは内容を見ていこう。3行目は「#」で始まるコメントだが、bashのみで利用する場合は「#」を削除すれば、いくつかのシェル変数がローカル変数になる。

4～16行目のcase制御構造では、引数の数（$#で参照）に基づいて、初期値・終了値・増分値をそれぞれloop_start、loop_end、loop_stepに格納している。回数だけ指定する場合は初期値と増分値をいずれも1、初期値と終了値だけ指定する場合は増分値を1と考えればいい。また、引数の数が3～5個以外なら、使い方（Usage）を端末画面に表示してシェル関数を終了する。

17～23行目では、繰り返し処理の条件判断に使われる演算子の決定と、増分値の正負チェックを行う。プロトタイプでは演算子は「-le」（より小さいか等しい）で固定だったが、初期値と終了値を自由に指定できる完成版では、初期値が終了値より小さい場合には「-le」、逆に初期値が終了値よりも大きい場合には「-ge」（より大きいか等しい）と使い分ける必要がある。また、初期値が終了値よりも大きい場合、増分値はマイナスでなければならない。

24行目では、evalを利用して、最初の引数（$1で参照）で指定された名前のシェル変数に、初期値（$loop_startで参照）を設定する。

以上の準備が完了したら、いよいよ25～29行目のwhile制御構造で繰り返し処理を行う。まず、25行目のwhileの条件部で、繰り返しを続行するかどうかの判断を行う。初期値と終了値の大小関係により、条件判断に使用する演算子（$loop_opで参照）が異なることに注意されたい。もちろん、この演算子の変数展開やループ変数を間接展開するためにevalが必要だ。

do～done間の処理では、最後の引数で指定した内容をコマンドラインとして実行する26行目に注目してほしい。よく見ると、evalの引数としてevalが書かれた二段重ねの構成になっている。

先頭のevalのコマンドライン処理は、最後の引数の内容を取り出すのに使われる。最後の引数の番号は、引数の数（$#で参照）と等しいので、最後の引数の内容はevalを利用した間接展開「¥$$#」で取り出せるのだ。

しかし、単に引数の内容を取り出しただけでは、そこに含まれる変数の展開処理やパイプなどの処理が行われない。そこで、もう一度evalを実行することで、変数展開などの処理を行っているわけだ。

Linux日記

第24回　メール配送（11）

組織の規模や方針によって、メールサーバの運用形態もいろいろなものが考えられます。今回は、どのような運用形態があるのかを見てみましょう。

illustration ; Aki

文：榊 正憲
Text : Masanori Sakaki

毎月書いている原稿のファイル名を見たら、どうやら今回は24回目のようである（もしかすると数え間違えがあるかもしれないが）。「Linux日記」というタイトルで、よくもまぁ2年もLinuxのことをろくに書かずに続けてきたものだ（今度インストールするのもBSDの予定だ）。もっとも、2年のうちの1年半は、DNSとsendmailの話だったから、Linuxデペンドな話というのはほとんどないのだが。

この原稿を書いているのは7月上旬、関東地方は梅雨の合間で、真夏並みの暑い日が続いている。昨年の夏は、10年選手のエアコンがくたばり、新しいものに交換したのだが、今シーズンは、意外にエアコンを使っていない。かつては、5月の下旬ごろから9月いっぱいくらいまでは24時間エアコンを入れっぱなしだったのであるが、今年は、1日数時間程度しか使っていない。以前に行った部屋の模様替えにより、窓を開けて風を通せるようになったからだ。筆者の住んでいる場所は海沿い、すなわち海洋性気候というおかげもあり、都心ほど気温が上がらない。そのため、窓による通風効果が非常に大きいのだ。もっとも、湿度はかなり高くなるが（ある夜など、湿度計の読みで99%まで上がっていた。この時はさすがにエアコンのお世話になったが）。加えて、サーバ構成の変更や、トラブルで動かなくなったマシンなど、常時稼働させているマシン数が減ったという影響もある。おかげで、晴れた日の午後でも、自然通風冷却で室温は28度程度で済んでいる。こういう時は、東京を離れてよかったとしみじみ思う。

前のエアコンは、最後には、周辺温度より気持ち低温の空気を吹き出しているかなぁという程度にも関らず、室外機は熱風をぶぉんぶぉん吐き出していたから、さぞかし無駄な電気を喰っていたのだろう。今年は去年に比べて、電気代が大幅に低下している。マシン台数も減っているので正確にはわからないが、おそらくエアコンだけで月に1万円くらい節約できているようだ。去年買ったエアコンは、今年1年で元が取れそうである。

さて、前回までは、sendmailを動かすための各種設定について解説した

Column

東京を離れるということ

筆者は生まれてからずっと東京暮らしだったのだが、結婚退職を機に神奈川県に移った（秋葉原までバスと電車で2時間といったところだ）。住環境は都内よりもいいし、都心ほど暑くなくてよいのだが、もちろんデメリットもある。ひとつは秋葉原が遠くなったということだ。実家、小学校から大学（予備校も含む）、会社のどれも、秋葉原まで乗り換えなしで20分ないし45分程度で行けた）。ここ最近くやしいのは、新しい通信サービスがなかなか提供されないことである。実家のある地域は、試験運用時からFTTHが引き込めたようだが、今暮らしているところは、いつになったらサービスが始まるのかもわからない。

（もちろん、あれだけの解説で実際に設定できるわけではないが）。今回は、ネットワーク上でsendmailをどのように運用するかという話をする。この連載の最初のころに説明した話を繰り返す部分もあるが、かれこれ1年経つので、復習だと思ってほしい。

■単純なメールシステムの構成

まず、一番基本的な構成を考えてみよう。インターネットに常時接続されたサイトに、sendmailが動作するコンピュータを1台用意する。この組織は独自のドメイン名takobeya.comを持ち、このドメイン名について自サイトでDNSネームサーバを運用しているとしよう。メールサーバmail.takobeya.comとネームサーバns.takobeya.comはグローバルIPアドレスを持ち、インターネット側から自由にアクセスできる。この組織のユーザーは、mail.takobeya.comにログインし、メールサーバ上で動作するMUAを使って、user@takobeya.comという形式のメールアドレスを使ってメール送受を行う。

メールの配信では、DNSのMXレコードが重要な役割を果たす。送信側MTAは、宛て先メールアドレスのドメイン部についてMXレコードを検索し、送信先MTAを判定するからだ。したがって、外部からメールを受信するMTAを運用する場合は、そのMTAが扱う受信メールアドレスのドメイン部について、MXレコードを登録しなければならない。

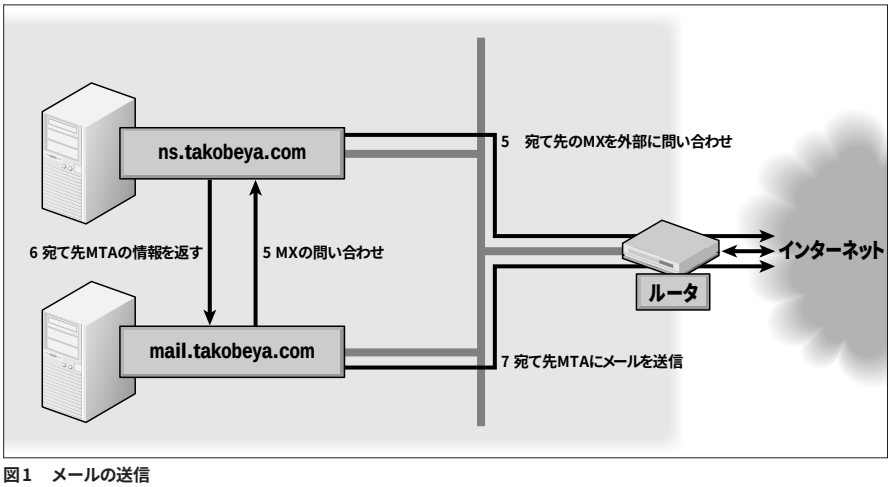


図1　メールの送信

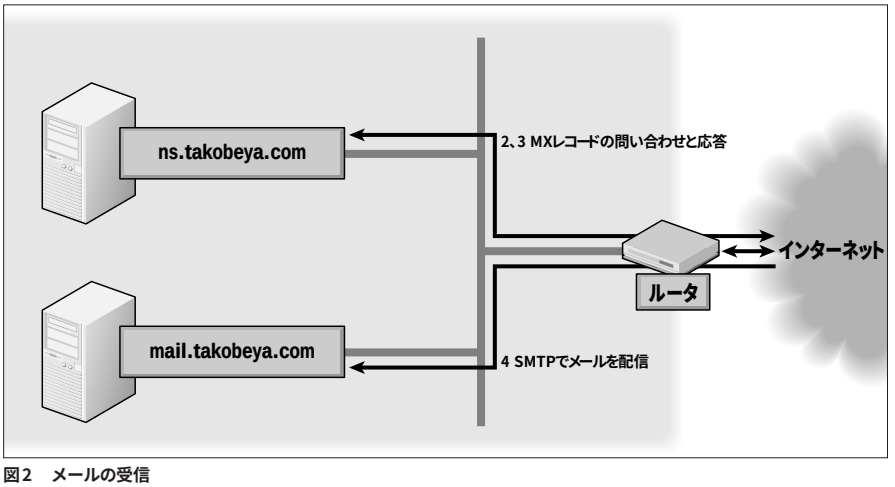


図2　メールの受信

まず、このサイトから外部にメールを送信する場合の処理を考えてみよう（**図1**）。宛て先はsomeone@ascii.co.jpとする。

1. **ユーザーが、メールサーバ上で動作するMUAを使って、宛て先としてsomeone@ascii.co.jpを指定したメールを作成する。**
2. **ユーザーがメール送信処理を行うと、MUAからsendmailが起動され、作成したメールが渡される。**
3. **sendmailは、MUAから指定された宛て先アドレス、あるいはメールヘッダ中に記述された宛て先アドレスを認識する。**
4. **指定された宛て先アドレスについて配信エージェントを選択する。この場合は、外部宛なので、sendmailに内蔵されたSMTP配信エージェント（あるいはESMTP配信エージェント）を選択する。**
5. **sendmailは、宛て先アドレスのドメイン部について、DNSにMXレコードを問い合わせる。**
6. **DNSは、ascii.co.jpについて、mail-svr.ascii.co.jpというメールエクスチェンジャ情報を返す。さらに、このホストのIPアドレスを求める。**
7. **sendmailは、送信するメールへのヘッダの付加などを行ったのち、目的のMTAに対してSMTP接続を行い、メールを送信する。**
8. **ログの記録などの処理を行い、MUAから起動されたsendmailは終了する。**

どこかのサイトのMTA（mail-svr.ascii.co.jp）がこのサイト宛にメールを送った場合は、次のような処理が行われる（**図2**）。

1. 発信側MTAが、MUAか別のMTAからmasa@takobeya.com宛のメールを受け取る。
2. 発信側MTAは、takobeya.comのMXレコードを検索する。
3. ns.takobeya.comはMXレコードの検索に対して、mail.takobeya.comを返す。
4. mail-svr.ascii.co.jpは、mail.takobeya.com上で動作しているデーモンモードのsendmailにSMTP接続を行い、メールを送信する。
5. mail.takobeya.comは受信したメールの宛て先を調べ、配信エージェントを選択する。この場合はローカル配信エージェントが選択される。
6. mail.takobeya.comのsendmailは、受信したメールをローカル配信エージェントに渡す。
7. ローカル配信エージェントは、受信したメールをユーザーのメールボックスにスプールする。
8. ログの記録などの処理を行う。

SMTPを使ってメール配信を行う場合について、以上がもっとも基本的なsendmailの動作である。送信側MTAが宛て先アドレスのドメイン名についてDNSにMXレコードを問い合わせ、返されたメールエクスチェンジャに対して送信を行うという点が重要だ。

■複数のMTAホスト

組織の規模が大きかったり、特殊な構成でメールシステムを運用する場合、複数のMTAホストを用意する場合がある。これにはいろいろなパターンがある。

- **部署のサブドメインまで含むメールアドレスを使用し、組織の部署ごとにMTAが存在する場合**
- **組織の全ユーザーのメールアドレスを統一的に扱うハブサーバと、部署ごとのMTAが存在する場合**
- **負荷分散、信頼性の向上のために、複数のサーバを運用する場合**

1つの組織で複数のMTAを運用する場合の運用パターンをいくつか紹介しよう。

■サブドメイン単位のMTA

現在、会社などの組織では、「ユーザー名@組織のドメイン名」というメールアドレスが一般的だが、組織によっては、「ユーザー名@サブドメイン.組織のドメイン名」というメールアドレスを使うこともある。たとえば大学などで、ネットワーク管理が学部や研究室の単位で行われている場合、このような形になることが多い。全学のレベルでは、学部単位のサブドメインやグローバルIPアドレスの割り当てなどを行うだけで、それ以外の管理は、DNSもメールもすべて下位組織まかせという場合、このような形になる（**図3**）。

たとえば、代々木大学の工学部計算機科学研究室でMTAを動かす場合、nobu@cs-lab.tech.u-yoyogi.ac.jpといったメールアドレスを使うことになる。そしてこのメールアドレスで外部からメールを受信するために、代々木大学のDNSには、cs-lab.tech.u-yoyogi.ac.jpというドメイン名に対して、たとえばms.cs-lab.tech.u-yoyogi.ac.jpといったMXレコードを登録することになる（**リスト1**）。

このような運用形態は、組織が複数のMTAを持つといっても、最初に説明した基本的な構成とほとんど違いはない。MTAを配置するドメインレベルが、組織単位のレベルではなく、組織のサブドメインになっているだけだからだ（このような組織では、DNSの管理も学部などの単位で委任されているのが普通だ）。

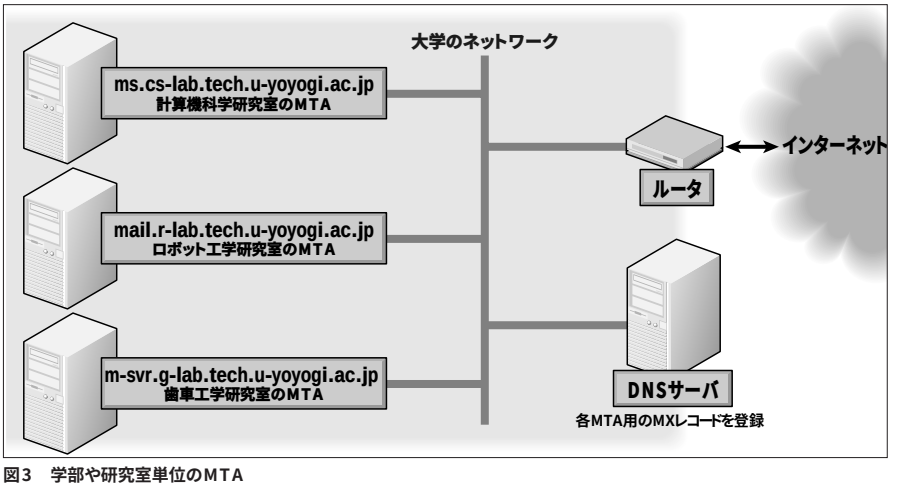


図3　学部や研究室単位のMTA

▼リスト1　tech.u-yoyogi.ac.jpゾーンのDNSデータベース

```
;               計算機科学研究室
cs-lab          IN MX       10 ms.cs-lab.tech.u-yoyogi.ac.jp.
ms.cs-lab       IN A        192.168.14.3
;               ロボット工学研究室
r-lab           IN MX       10 mail.r-lab.tech.u-yoyogi.ac.jp.
mail.r-lab      IN A        192.168.17.18
```

大企業など、管理が一元化されている組織であっても、ユーザー数が非常に多い場合などは、事業部などの単位でこのような形で運用することもある。歴史的経緯（事業部単位でネットワークが発展してきたなど）により、ユーザー名の重複がある場合なども、このような形で運用を続ける場合もある。しかしこのような形は、メールアドレスが長くなり、また人事異動などでメールアドレスが変わってしまうことになる。

■ハブサーバの導入

会社などの組織では、メールアドレスに事業部などのサブドメイン名を含めず、「ユーザー@会社名.co.jp」といった形にすることが多い。しかしユーザー数が非常に多い大企業などでは、実際には事業部や事業所といった単位で、社内で複数のMTAを運用していることが多い。MTAを複数使いながら、すべてのMTAで同じ形式のメールアドレスを使いたい場合は、ハブサーバを運用することになる（**図4**）。

ハブサーバは、外部とのメールのやり取りの窓口として機能する。takobeya.comは、user@takobeya.comという形式のメールアドレスを使用し、MTAとしてハブサーバhub.takobeya.com、部署サーバms0.tech.takobeya.com、ms1.sales.takobeya.comなどを運用している。まず、メールの受信処理を考えてみよう。

外部からこの組織に送られてきたメールは、すべてハブサーバhub.takobeya.comが受信するようにする。そのために、組織のドメイン名takobeya.comに対するMXレコードがhub.takobeya.comを示すよう設定する。ハブサーバは、受信したメールについて、宛て先アドレスを評価し、適切な部署単位のMTAに中継しなければならない。宛て先アドレスのドメイン名はすべてtakobeya.comになっているはずなので、メールの中継は、ユーザー名に基づいて行うことになる。これには、エイリアス定義による転送機能が使える（ファイルクラスやデータベースアクセス機能を使って、sendmail.cf中で評価することもできるだろう）。たとえば、masa@takobeya.com宛のメールをms0.tech.takobeya.comに中継するのであれば、aliasesファイルに**リスト2**のように記述する。

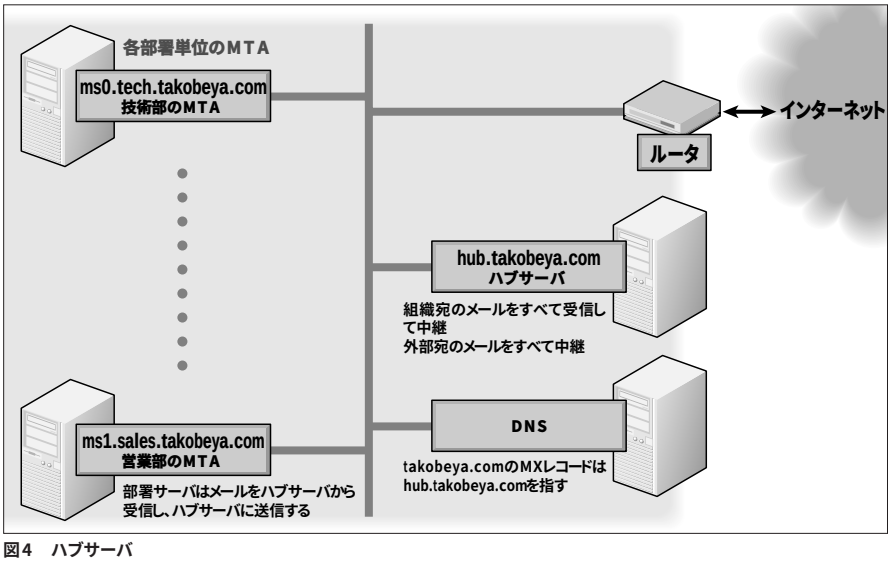


図4　ハブサーバ

▼リスト2　ハブサーバを利用するときのaliasesファイル記述例

```
masa@takobeya.com:          masa@ms0.tech.takobeya.com
```

これにより、masa@takobeya.comという宛て先アドレスを持つメールは、ハブサーバによってmasa@ms0.tech.takobeya.comという宛て先に転送される。部署サーバms0.tech.takobeya.comは、ハブサーバから中継されたメールを各ユーザーごとにスプールする。これで、外部から送られたメールを目的の部署サーバに配信することができるようになった。

次に、メール送信について考えてみよう。組織内のユーザーが作成したメールは、まずそのユーザーを管轄する部署サーバで処理される。部署サーバは、すべてのメールをハブサーバに中継する。これを受け取ったハブサーバは、宛て先アドレスを評価し、インターネット上の適切なMTAや、社内の部署サーバに中継する。同一の部署サーバが管轄するユーザー間のメール配信であれば、ハブサーバにわざわざ中継する必要はないが、このようにするには、宛て先アドレスがローカルユーザー宛かどうかを部署サーバのレベルで判定しなければならない。

メール送信に際しては、From：アドレスについても考える必要がある。ms0.tech.takobeya.comから送られたメールであっても、From：行は、user@ms0.tech.takobeya.comやuser@tech.takobeya.comではなく、user@takobeya.comでなければならない。これは、部署サーバのsendmail.cf中で指定できる。

ハブサーバを使う場合の運用形態はこれだけではない。たとえば、部署サーバは、外部にメールを送る際に、ハ

ブサーバを経由せず、直接目的のMTAに接続することもできるだろうし、組織内に送るメールであれば、別の部署サーバに直接中継することも可能だ。このようにすることで、ハブサーバは受信窓口の機能だけ果たせばよくなるので、ハブサーバの負荷を大幅に低減できる。しかし、sendmail.cf中に記述するルールは、場合分けが増えるので、部署サーバのsendmail.cfや設定ファイルなどは多少複雑になる。どのような構成で運用するかは、組織や管理者次第である。

■サーバの負荷分散

ハブサーバを導入することで、複数のMTAが存在する場合でも、組織のレベルで統一された形でメールシステムを運用できるようになる。ところで、サーバを複数運用する目的のひとつは、多くのユーザーにサービスを提供すること、つまり、1台のサーバでは処理をまかないきれない際の、負荷分散のための対処である。前述したハブサーバの運用は、残念ながら、負荷分散の役には立たない。部署のサーバの負荷は少ないが、ハブサーバには、組織内、組織外のすべてのメール中継処理の負荷がかかることになる。多くのユーザーがいる場合、ハブサーバがボトルネックになる可能性があるわけだ。

ハブサーバを使う場合の負荷分散について考えてみよう。まず、外部からメールを受信するハブサーバと、送信に使うハブサーバを分けることができるだろう。外部に対する窓口となるMTAは、DNSのMXレコードで指定される。一方、送信時や組織内のメール配信に使われるハブサーバは、MXとは関係なく指定できる（たとえばsendmail.cf中にハードコードすることができる）。これにより、組織に送られてくるメールを扱うMTA、組織が送り出すメールを扱うMTAが別ホストとなり、各コンピュータあたりの処理量は大幅に低減する。

DNSのMXレコードは、同じドメイン名について、複数のメールエクスチェンジャを指定できるということを思い出そう。MXにより、同じプライオリティで複数のメールエクスチェンジャを指定した場合、組織宛のメールは、指定された複数のMTAに分散して配信されるようになる。各MTAは受信メールをスプールせず、ただちに部署サーバに中継するだけなので、同じ宛て先のメールが異なるMTAに送られても問題はない。最終的には、目的のMTAに届き、スプールされる。これにより、さらに負荷を分散することができる（**図5**）。また、このような構成は、バックアップにもなる。ハブサーバのどれかがダウンしても、メールはほかのハブサーバに配信されるからだ。

外部への送信や社内への配信に使われるMTAも、複数のホストに分散することができる。たとえば、部署サーバごとに違うハブサーバに中継するように設定することで、負荷を分散させることができる。あるいは、複数のハブサーバにランダムに送るようにすることもできる。前に説明したように、各部署サーバが直接インターネットに送出するように設定してもいい。

■ユーザー用サーバの並列化

ハブサーバは比較的簡単に複数の並列運用を実現できるが、部署用のサーバ、あるいは1台で組織全体の面倒を見るといった構成では、サーバの並列化は難しくなる。これは、ユーザーのためのデータをローカルに保持する必要性があるからだ。受信したメールは、各ユーザーごとにメールボックスにスプールされる。サーバを並列化した場合、複数のサーバの間でメールボックスを一元的に扱う必要が生じるのである。ハブサーバは、受信したメールはすぐに別のMTAに中継してしまうので、ユーザー向けのデータが長時間保持されることはなく、ユーザーもアクセスしないので、このような問題は起こらない。

もう少しあとで説明するが、MUAをMTAとは違うコンピュータ上で使用し、POP／SMTPプロトコルでメールのやり取りをするのであれば、送信用MTAと受信用MTAを違うホストにすることは可能である。しかしこれは負荷分散にはなるが、バックアップの役には立たない。クライアント形式のMUAを使う場合の基本的な問題は、

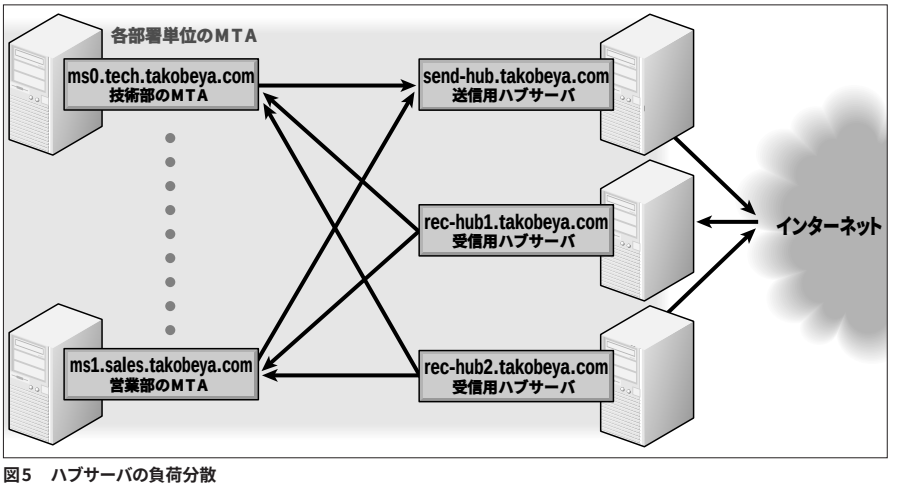


図5　ハブサーバの負荷分散

クライアント側でPOPサーバ、SMTPサーバを1つしか指定できないということである（**図6**）。

このようなクライアントに対してサーバを多重化するためには、同じ名前で複数のサーバにランダムにアクセスでき、そしてこれらのサーバはすべて、ユーザーのメールボックスの正確なコピーを持っていなければならない。したがって、これを実現するのは、簡単な作業ではない。

消極的な対応としては、受信だけ行えるバックアップMTAを用意するという方法がある。これはDNSのMXレコードと2台のMTAホスト（ms-mainとms-back）で実現できる。DNSに2台のMTAホスト用のMXレコードを登録する。この時、プライオリティを違う値にしておく。これにより、通常のメール配信は、プライオリティの高いほうのMTA（ms-main）に対して行われる。低いほう（ms-back）に配信されるのは、ms-mainがダウンしている時だけとなる（**図7**）。

リスト3のようにMXレコードを登録することで、送信側MTAは、メールをms-main.takobeya.comに配信する。ダウンなどにより、ms-mainへの送信に失敗した場合は、ms-back.takobeya.comに配信する。これにより、とりあえず、宛て先側のダウンという理由で、発信者にエラーメールが返されるという事態は避けることができる。

ユーザーが使うMUAは、ms-mainをPOP／SMTPサーバとして設定する。これにより、ms-mainが正常に動作している限り、普通にメールの送受信を行える。ms-mainがダウンしている間は、メールの送受は行えないが、この組織宛に送られてきたメールは、ms-backが受信し、スプールしている。さて、それではms-backが受信したメールは、どのようにして読めばいいのか？　いくつか方法を考えてみよう。

1. **ユーザーのMUAの設定を一時的に変更して、ms-backに接続する。この方法なら、ms-mainがダウンしている間、受信だけでなく、送信も可能である。しかし、方法としては非常にうっとうしい。**
2. **ms-back上に、スプールしたメールをms-mainに転送するスクリプトを用意する。そして、ms-mainが復旧した後、手動で、あるいは自動でこのスクリプトを実行し、ms-backが受信したメールをms-mainに転送する。**
3. **受信したメールを無条件にms-mainに転送するようにms-backを設定しておく。これはエイリアス機能を使えば実現できる。ただしこの方法は、ms-mainのダウンタイムが長くなると、受信したメールを発信者にエラーとして返送してしまうので、この配信不能エラーと判定する時間を十分長くしておく必要がある。あるいは、このような処理を行う専用の配信エージェント（配信できるまでひたすら試行するといったもの）を定義するという方法もある。**

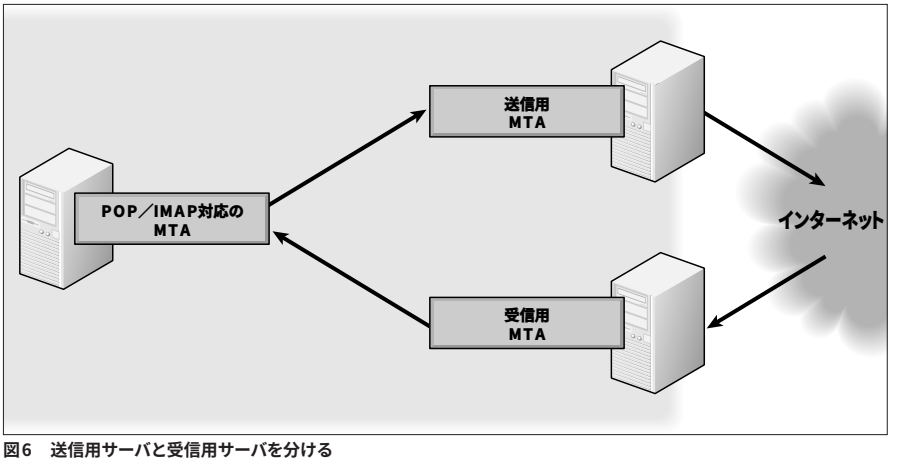


図6　送信用サーバと受信用サーバを分ける

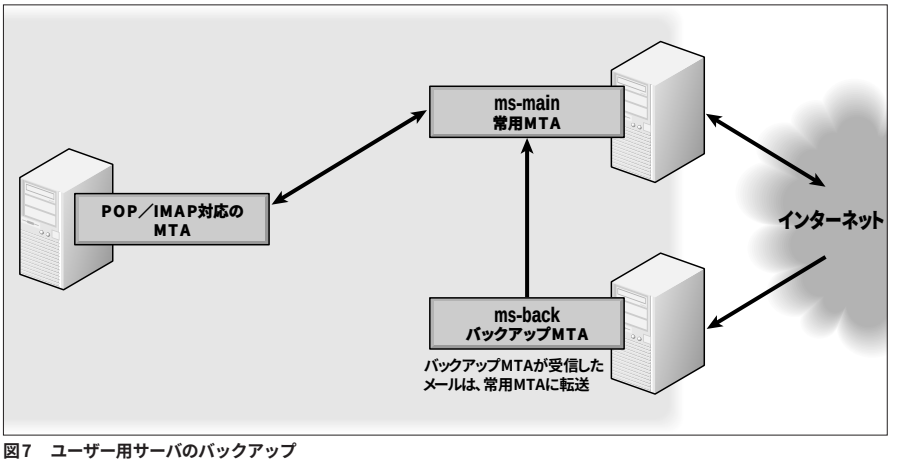


図7　ユーザー用サーバのバックアップ

▼リスト3　takobeya.comゾーンのMXレコード

```
@               IN MX      10 ms-main.takobeya.com.
                IN MX      20 ms-back.takobeya.com.
```

付録CD-ROMに収録した Kondara MNU/Linux 2.0 のインストール

本誌付録CD-ROM収録のKondara MNU/Linux 2.0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デジタルファクトリ株式会社からサポートを受けることはできません。

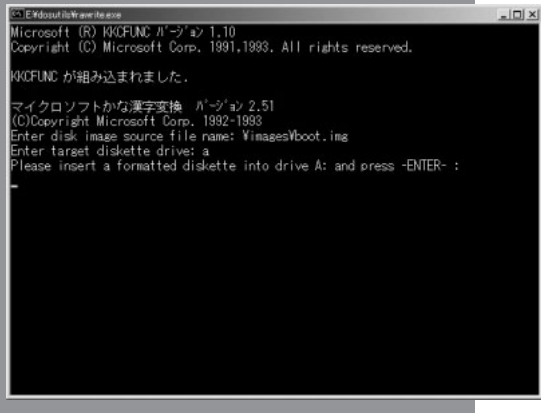
また、CD-ROMのメディアに不良があった場合は、お手数ですが住所、氏名を明記のうえ（linux-cd@ml.ascii.co.jp）宛にご連絡くださるようお願いいたします。なお、Linuxの設定などについてのご質問にはお答えできませんので、あらかじめご了承くださいませ。

インストールの前準備

これからKondara MNU/Linux 2.0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

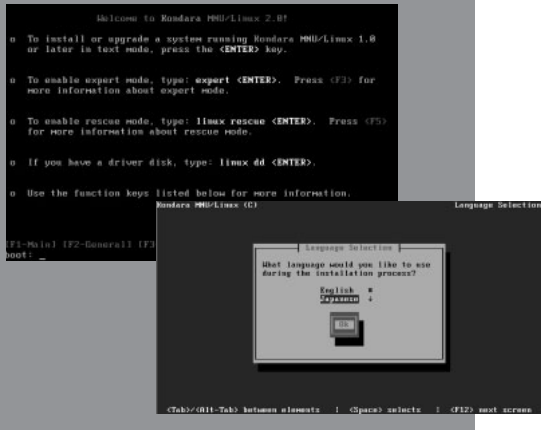
CD-ROMからインストーラを起動できない場合は、以下の手順でインストーラ起動用のフロッピーディスクを作成します（ここでは、フロッピーディスクドライブがA:であるとして解説します）。

（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、テキストモードのインストーラが立ち上がります。Kondara MNU/Linux 2.0のインストーラには、グラフィカルモードが用意されていません。グラフィカルな画面が表示されなくても、CD-ROMの不具合ではありませんから、テキストモードでインストールを進めます。

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。インストーラのテキストモードでは、［Tab］で項目を移動、［Space］で項目をチェックし、［Enter］で［OK］ボタンなどを押して操作します。

キーボードの設定とインストールタイプの選択

デフォルトのキーボードタイプには、[jp106]が選択されています。キーボードに日本語が刻印されているキーボードを使用する際は、デフォルトの[jp106]を選択したまま[OK]を押します。

次にインストールタイプを選択します。[ワークステーション]と[サーバシステム]を選択すると、ハードディスクのデータがすべて削除されたあとにLinux用のパーティションが作成されます。また、[既存システムのアップグレード]は、すでに古いバージョンのKondara MNU/Linuxがインストールされている場合に選択するオプションです。

ここでは、もっとも柔軟に設定できる[カスタムシステム]を選択します。

パーティション作成ツールの選択

Linux用のパーティションを作成するためのツールを選択します。[fdisk]は柔軟にパーティション作成できるのですが、コマンドを使って操作するので初心者にはお勧めできません。

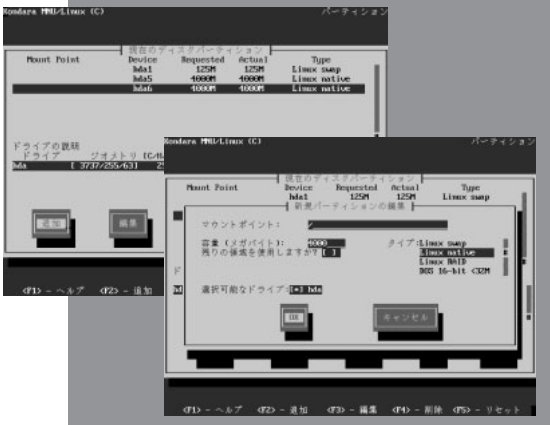
ここでは、fdiskに比べるとやや柔軟性に欠けますが、コマンド操作が不要な[Disk Druid]を選択します。

Linux用パーティションの作成

まず、[追加]を押して、新しくLinux用のパーティションを作成します。[新規パーティションの編集]という別のメニューで、

マウントポイント	容量	タイプ	
「/」		4000	Linux native
なし		128	Linux swap

という2つのパーティションを順番に作成します。フルインストールすると、ディスクを約3Gバイト消費するので、フルインストールする場合は、「容量」に4000と入力して、Linuxシステム用のパーティションを4Gバイト程度作成します。「Linux swap」の「容量」には、マシンが搭載するメモリの1～2倍程度の数字を、Mバイト単位で入力します。上の例では、「Linux swap」に128Mバイトを割り当てています。



パーティションのフォーマットとLILOの設定

手順6でLinux nativeとして作成されたパーティションが表示されます。[*]でチェックされたパーティションがフォーマットされます。[フォーマット中に不良ブロックをチェックする]をチェックすると、ハードディスクに不良部分がないかどうかをチェックしながら、パーティションをフォーマットします。

次に、ブートローダLILOに付けるオプションを指定します。デフォルトでは[リニアモードを使用する(一部のSCSIドライブで必要)]がチェックされていますが、一般的なIDEタイプのディスクを使用する場合はチェックをはずしておきましょう。チェックをはずしたら[OK]を押します。

LILOのインストール先の選択

Linuxのブートローダ**LILO**をインストールする場所を選択します。**Windows 9x／MeとLinux**を共存させる場合は、［**/dev/hda　マスターブートレコード（MBR）**］を選択してハードディスクから**Linux**を起動し、**Windows NT／2000**と共存させる場合は［**/dev/hda7　ブートパーティションの最初のセクタ**］を選択して、あとで作成するレスキュー用のフロッピーディスクから**Linux**を起動します。

ファイアウォールとネットワークの設定

まずはファイアウォールを設定します。どれを選択すればよいのかわからない場合は、デフォルトで選択されている［**中**］でいいでしょう。［**中**］を選択すると外部からの**telnet**接続を受け付けないので、**Linux**マシンをインターネットに接続する場合も安全です。

次にネットワークの設定です。プロバイダなどの**DHCP**サーバを利用して、**IP**アドレスなどのネットワーク情報を取得する場合は、［**bootp/dhcpを使用する**］をチェックします。また、**DHCP**を使用しない場合は、**IP**アドレスや**DNS**サーバの**IP**アドレスなどを各欄に入力します。

なお、この場面ではダイヤルアップ接続を設定できません。**Linux**マシンをダイヤルアップ接続する場合は、インストール後に**PPxP**などを使って設定してください。

マウスとタイムゾーンの設定

まず、マウスを選択します。**PS/2**タイプの**2**ボタンマウスを使う場合は、［**Generic - 2 Button Mouse（PS/2）**］を選択して、［**3**ボタンマウスのエミュレーションを設定しますか？］をチェックします。このオプションをチェックしておくと、**2**ボタンマウスの左右ボタンを同時に押すことで、**3**ボタンマウスの真中のボタンを押す効果を得られて便利です。

次にタイムゾーンを選択します。デフォルトで［**Asia/Tokyo**］が選択されています。たいていのユーザーはこのままでよいでしょう。**Linux**を**Windows**と共存させる場合は、［**ハードウェアクロックをGMTに合わせて設定しますか？**］をチェックしないでおきましょう。

Linux管理者rootのパスワード設定

Linuxには大きく分けて、ソフトウェアのインストールなどシステム管理を行う特権ユーザー［**root**］と、**Web**ブラウズやメールの読み書きなど一般的な作業を行う一般ユーザーの**2**種類が存在します。ここでは、**Linux**管理者のパスワードとして、［**パスワード**］と［**パスワード（再入力）**］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。

12 一般ユーザーの作成

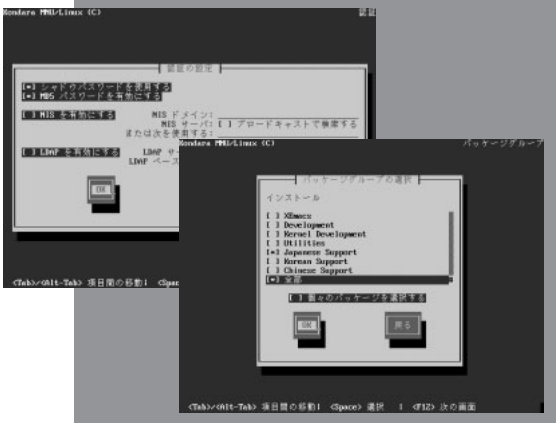
［ユーザーID］には一般ユーザーのログイン名を入力します。［パスワード］と［パスワード（再入力）］には、一般ユーザーのパスワードとして同じものを入力します。［フルネーム］は一般ユーザーのフルネームを入力する欄ですが、特に入力する必要はありません。

次の［ユーザーアカウントの設定］では、［OK］を押して次の場面に進みます。

13 認証の設定とパッケージグループの選択

［認証の設定］では、デフォルトの状態で［OK］を押して次の場面に進みます。

［パッケージグループの選択］で、どのグループを選択して良いのかわからない場合は、［全部］を選択してください。［全部］を選択すると、ハードディスクを3Gバイト程度消費します。

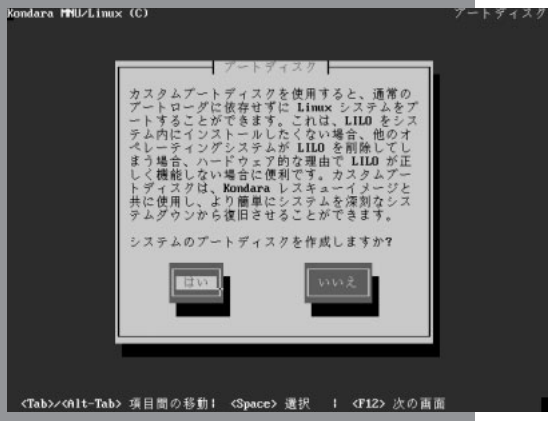


14 ビデオカードの自動認識

［X設定の検出結果］でビデオカードが自動認識され、［インストールの開始］で［OK］を押すと、パッケージのインストールが始まります。

15 パッケージのインストール

［パッケージグループの選択］で［全部］を選択した場合は、K6-II/400MHzと128Mバイトのメモリを搭載したマシンで、パッケージのインストールに約40分かかります。パッケージインストールの途中で［CD-ROM変更］が表示された場合は、付録CD-ROMのDisc 2をセットして［OK］を押します。



ブートディスクの作成

Linuxを起動できるレスキュー用のフロッピーディスクを作成します。このフロッピーディスクは必ず作成しておきましょう。空のフロッピーディスクを1枚セットして［はい］を押すと、フロッピーディスクの作成が始まります。

16

Xの設定

Xの設定を始めます。［ようこそ］の場面では［OK］を押すと、ビデオカードが自動認識されます。［OK］を押して次に進みます。

17

既存のXF86Configのエラー

この場面では、［はい］を押して設定ファイルのシンボリックリンクを張ります。

18

モニタの設定（1）

使用するモニタが一覧に表示されない場合は、［カスタム］を選択して［OK］を押します。［カスタムモニターの設定］では［OK］を押して次に進みます。

19

モニタの設定（2）

［カスタムモニターの設定（続き）］では、使用するモニタのマニュアルを参照しながら、適切なものを選択します。

ビデオメモリとクロックチップの設定

［ビデオメモリ］では、使用するビデオカードが搭載するメモリの量を指定します。

［クロックチップの設定］では、［Clockchipを設定しない（推奨）］がデフォルトで選択されています。このまま［OK］を押して次に進みます。

解像度の設定

ビデオモードには、ビデオカードがサポートする解像度と色数が一覧表示されます。色数に8ビットを選択すると、256色しか表示されません。約65000色で表示される16ビット以上を選択しましょう。解像度には、モニタのマニュアルに記載されている最大解像度以下のものを選択してください。また、［Space］で複数のモードをチェックしておくと、Xを使用する際に、［CTRL］＋［ALT］＋［+］や、［CTRL］＋［ALT］＋［−］で解像度を切り替えられます。

次の場面で［OK］を押すと、Xの表示がテストされます。Xの表示に成功すると、Kondara MNU/Linuxのデスクトップ画面が表示されます。デスクトップ画面に表示される質問に答えてXの表示テストを終了します。

起動するデーモンの選択

［サービス］では、Linuxと同時に起動するデーモンを選択します。［＊］でチェックされているデーモンが起動されます。起動するデーモンに［＊］を入れて、起動しないデーモンからは［＊］をはずします。

［アプリケーションCD］は、商用ソフトが収録されたCD-ROMをセットする場面ですが、本誌には収録されていないので、［キャンセル］を押してインストールを終了します。

お疲れさまでした。

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17回
【Itanium】（アイテニアム）
【Xeon】（じーおん）
【Pentium 4】（ぺんてぃあむ・ふぉー）
【Athlon】（あすろん）
【Athlon MP】（あすろん・えむぴー）

Itanium

【アイテニアム】

おおいた

乱世を極めるCPU業界にインテル社が投入したサーバ向け64ビットCPU。ハードディスクなみに巨大な体躯に加え、フドウ小数点演算を得意とするため、「McKinley山のフドウ」との異名をもつ。ふだんは他に類を見ないほど重装備のヒートシンクが必要なくらい温厚な人柄で知られるが、かつては「鬼のフドウ」と異名をとるほど乱調な開発スケジュールで巷の開発者を恐れさせた。あの“拳王”ラオウ（編注：ビル・ゲイツ？）をして生まれて初めて対応OSを予定どおり出せないのではないかと恐怖を感じさせたCPUとも言われている。事実、現在市場に投入されているItanium対応OSはLinuxばかりである。しかし、そんなフドウもユリア（編注：旧DEC Alpha開発チーム？）との運命的な出会いによって南斗五車星のひとつ・五車山峨斬（編注：わからん。好意的に解釈してEPICアーキテクチャの暗喩か？）の使い手へと成長するのであった。物語の最後では（編注：？？　フォローしきれません）恐れの克服を求めるラオウと闘い、壮烈な死を遂げる。またひとり世を去った漢（おとこ）への想いを胸に、ケンシロウは打倒ラオウの決意を新たにするのであった。

なお、インテルではほかにも南斗五車星シリーズとして「風のヒューイ」（2コマで市場から撤退する可能性を感じさせる次世代Itanium）、「雲のジュウザ」（パイプライン動作が雲のように無形なので、VLIWなのにスケジューリングがどうなるか誰にもわからない開発者泣かせの次々世代IA-64）などを予定しているという噂があるが、「蒼天の拳」の人気（および「北斗の拳」のリバイバル人気）が思ったより盛り上がらなかった場合は、「やましたたろーくん」シリーズや「シティハンター」シリーズにロードマップコードネームが訂正される可能性もある（例：すぐ鼻水が垂れるので結露が心配な“たろー”、アダルトゲームの高速動作に特化された“サエバリョー”など）。そうならないように、みんなで読者アンケートハガキを出して応援しよう！（編注：なんだかわからないけどLinux magazine編集部には送らないでください。お願いします）

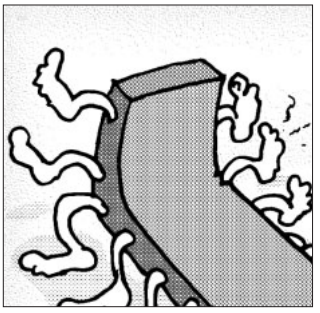
――Oh! Itanium（おー・いたにうむ）：「Itanium発表にあわせてソフトバンク パブリッシングがItanium専門誌を創刊した、っていう話を書こうと思ったんですよ」「……それで？」「もちろん、ビジネスに聡いあの会社ですから単なる創刊じゃ終わりませんよね。そこで“これからは地方の時代だ！　地域情報発信と組み合わせた誌面にしていこう！”と考えるわけです」「……だから？」「コンセプトとしては、CPU＋地域情報。さて、問題はどの地域を選ぶかです。今まであまり注目されてこなかったスポットがいい。というわけで選ばれたのが大分県」「……それで？」「誌名も“Oh! Itanium”（おーいたにうむ）ということでオチがついたねっ！ で、うまく行くハズだったんですよね」「……だから？」「そうしたら本誌8月号90ページに“Itaniumはアイテニアム

と読まなければならない”って書いてあるじゃないですか！これじゃネタが成立しないんですよ。困りますよ！　おまけに“テにアクセント”とか書いてあるし。そんな気をつかって読めるかちゅーねん。ところで大分県って、どこにあるんでしょうね」「……いいから原稿書けよ……」。

Xeon
【じーおん】

一部ユーザーにしか利用されず、ProというよりはManiaと名付けるべきだったとも噂される「Pentium Pro」後継系列のインテルCPU。パッケージ上のL1ブリッジに赤い塗装を施すと通常の3倍速で動作する。サーバ・ワークステーション用途といわれているが、スペースコロニーの管理目的に使用すると南米に墜落させられる可能性があるので初心者は注意が必要だ。

Pentium 4
【ぺんてぃあむ・ふぉー】


むかで

ムーアの法則に従って順調に集積率を上げ続けているCPUだが、レガシーアーキテクチャとの互換性を維持する必要があることなどから性能が必ずしも正比例で向上しないという問題を抱えている。業界の雄・インテルは、これをCPU市場全体を揺るがしかねない課題として認識、新理論を導入して解決をはかろうとしている。その成果として製品化されたのが「Pentium 4」だ。

Pentium 4では「足の数の多いものほど走るのが速い」という自明の事実をもとに、根本的に新しい思想のもと設計が行われている。この“足の数定理”については、古来から数学者によって「ゼノンのパラドックス」として証明がなされてきた。足が2本しかないアキレスと足が4本あるカメに徒競走をさせると、アキレスはいつまでたってもカメを追い抜けないというあれである。また、故事にも足が2本のウサギと徒競走をして勝つカメの話がある。従来のPCアーキテクチャでは、いわゆる「FC-PGA」として370本の足をもつCPUが使われてきた。インテルでは、この足の数の少なさがCPU速度向上のシーリングとなっていると主張。新たに足の数を423本に増やした「PGA423」規格を提唱するとともに対応CPUのPentium 4を開発した。内部アーキテクチャに新味はないが、足の数を増やすだけで理論どおり423÷370　1.14倍の体感的性能向上が見られており、インテルの主張の正しさが証明された。

さらなる計算能力の向上をはかるべく、より足の数を増やした次世代Pentium 4の開発が進められており、完成の暁には「mPGA478」規格に基づく478本の足をもつ強力なCPUが市場に投入されることになる。ただし、現状のPCアーキテクチャでは足の数も早期に頭打ちになることが予想され、対策としてインテルではチップ上部にもピンを生やす「DS PGA（Double-Sided PGA）」の規格化を検討。COMPUTEX TAIPEI 2001で発表している。

DS PGAが実現すれば、CPUの足密度＝速度は一挙に2倍になる可能性もあり、さらに高い放熱効果も期待できるのでファンレスシステムさえ実現するのではないかと観測する専門家もいる。

Athlon
【あすろん】


手洗いの優しさ？？

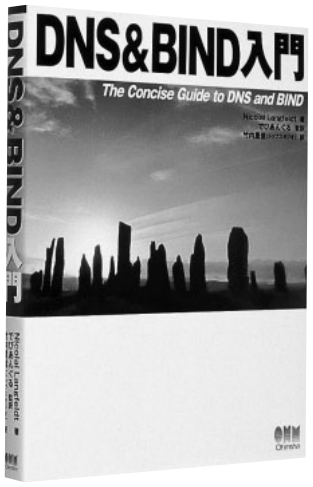
毛糸洗いに自信がもてるAMD製CPU。全自動洗濯機に汚れ物といっしょに放り込んだらスイッチポン。裏面の無数のピンが剣山のような効果を発揮。衣類の繊維質の奥の奥まで入りこんで頑固なドロや油汚れをかきだしてくれる。同時に柔軟作用も期待できるので、「洗濯機で手洗いのやさしさ」が得られる。さらに0.13μメートルの銅配線から放出される銅イオンにより殺菌効果も。お肌の敏感な乳幼児の衣服も清潔に保つことができる。まさにおしゃれ着からカジュアルまで、しっかり洗える万能CPUである。

最近ではさらに研究が進み、over 1GHz製品が通電時に発生する電磁波を応用すれば、ドライクリーニング（水なし電磁波洗浄）が可能なのではないかといわれている。AMDのJerry Sanders会長は、このニーズの高まりが今後期待できるとして旧東独・ドレスデンのFab30におけるギガクラスチップ歩留まりの向上を、当面の目標に掲げた。

Athlon MP
【あすろん・えむぴー】

毛糸洗いに2倍自信がもてるCPU。

Books



DNS＆BIND入門

Nicolai Langfeldt 著／でびあんぐる 監訳／竹内里佳 訳
オーム社
B5変形判／336ページ　本体価格 3000円

常時接続サービスが普及し、独自ドメインの取得が身近なものになってくると、これまでネットワークにユーザーの立場でしか参加していなかった人でも、今度は管理者として臨まなくてはならなくなる機会が増えてくるだろう。ところで、独自ドメインを取得して自宅やオフィスのネットワークをインターネットに接続する場合、名前解決の問題をどうするかというのは、かなりプライオリティの高い決定項目である。プロバイダやネットワークサービス会社に「丸投げ」してしまってもいいのだけれど、コストやシステムの柔軟性などを考えると「ここはひとつDNSも自前で」という気にもなってくる。

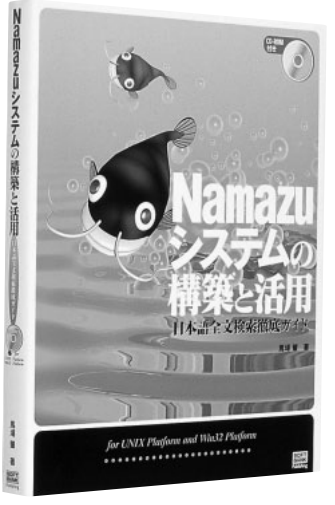
本書は、そんなDNSにまじめに取り組みたい初級管理者に読んでもらいたい1冊だ。基礎的なDNSの概念から、実際の設定例、DNSサーバを運用する際の注意点などが、わかりやすくまとめられている。BIND 8の解説を中心にBIND 4、BIND 9についても触れられている。

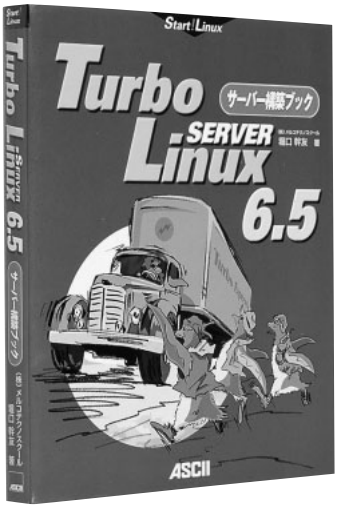
Namazuシステムの構築と活用 ～日本語全文検索徹底ガイド

馬場 肇 著
ソフトバンク パブリッシング
B5変形判／288ページ／CD-ROM1枚付き　本体価格 2800円

国立国会図書館、経済産業省、日本銀行、アップル、NTTドコモ、日産自動車、慶應義塾大、人民日報、民主党、アスキー……これらはオープンソースの日本語全文検索システムであるNamazuを使っているサイトの一例だ。NamazuはHTMLファイルから抽出した単語をインデックス化し、目的の文章を瞬時に見つけ出す。読者諸兄もWebサイトで単語検索の際に必ずお世話になっているはずだ。本書は全文検索システムの基礎知識からNamazuのインストール、Webサーバへの組み込み、各種設定など、サーバ管理者が実際に使えるところまで読者をリードするのは無論のこと、商用の検索システムの紹介や、はては未来の全文検索アルゴリズムにまで話が及ぶ。ただのシステム解説書とは趣を異にした力作である。

ちなみに、Namazuはテキストファイルならばなんでも扱えるので、ハードディスクに何百Mバイトもため込んだメールの検索に使ってみるのも面白い。





Turbo Linux Server 6.5 サーバ構築ブック

堀口幹友 著
アスキー
B5変形判／320ページ／CD-ROM1枚付き　本体価格 2800円

Turbo Linux Server 6.5をベースとしたサーバ構築の入門編。ただし入門編とはいっても、インストールと基本設定の解説に続いて、セキュリティポリシーの決定を含むネットワーク設計と実際のセキュリティの設定に2章分を割くなど、なかなか本格的。また後半では、BIND、Sendmail、Apacheといった定番サーバソフトによるネットワークサービス環境の構築についてポイントを押さえた解説がなされている。

実際に本書の対象読者となるような初めてサーバを構築する管理者にとって、最も不安なのが「何か必要な設定が抜けているのではないか」ということだろう。「わからないこと」は調べようがあるが、「知らないこと」は調べようがないからだ。特にセキュリティについては、知らなかったではすまない場合もある。まずは、本書のような総合的な解説書に目を通し、知っておくべきことを身に付けるようにしたい。

ハンダでGO！ ―自作派へ贈るおもしろPC製作術―

NOBU（のぶ）著
インプレス
A5判／240ページ **本体価格 1600円**

どこかで聞いたようなタイトルだが、本書は月刊「DOS/V POWER REPORT」に1999年から連載されていた同名の記事をまとめたもので、電車とは関係ない。さらにいえば、Linuxとも関係ない。内容を簡単にまとめると、「パーツを買ってきて組み立てるだけじゃつまんないから、いろんな用途に特化したマシンを作っちゃおう。となれば、もちろんマザーボードにハンダごてをあてるよね」といったところであろうか。

製作された究極マシンは22台。どれも個性的でイカしている。おかもちPC、屋台PCに冷蔵庫PC、さらには乗用車PCと何でもケースとして活用してしまう。しかし、そんなのはまだまだ序の口。BIOSのフラッシュROM4個を亀の子配線してスイッチで切り替える、バスを引き出してPCIスロットを増設する、GPIOでラジコン戦車を制御するなど、もうやりたい放題。今世紀最高のPCバカ書籍といっても過言ではない。いや、感動した。

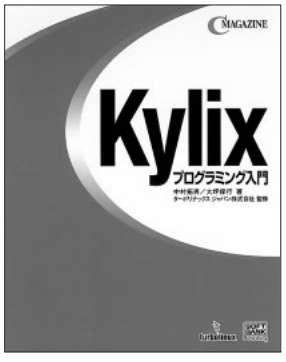
インターネット白書

インターネット協会 監修
インプレス
A4変形判／300ページ／CD-ROM1枚付き
本体価格 4800円

Red Hat Linux 7.1 サーバー構築入門

高原 利之 著
ソーテック社
変形判／288ページ／CD-ROM2枚付き
本体価格 3800円

Kylixプログラミング入門



中村拓男、大坪保行 著
ターボリナックスジャパン株式会社 監修
ソフトバンクパブリッシング
B5変形判／264ページ／CD-ROM1枚付き
本体価格 2700円

Rubyを256倍使うための本 網道編

ただただし、arton 共著
アスキー
B5判／248ページ
本体価格 1200円

読者の声

俺にもいわせろ！

暑い日が続きます。原付ライダーの担当は、熱っせられたアスファルトと車の排気ガスとエンジン熱そして、ジリジリと照りつける太陽光線によって、会社への往復だけで、干からびてしまいそうです。また、例年通り「今年こそ美白に！」と目標を立てていた担当ですが、努力空しくすでに丸焦げになってしまいました……。海に行ったわけでも、山に行ったわけでもないのに、こんなに焦げてしまう担当って、なんて幸せ（泣）。

Webブラウザ

「超入門シェルスクリプト」はすごく参考になります。個人の環境をWindowsからLinuxへ移行中なのですがネットサーフィンなんかはWindowsを利用してしまいがちなので（簡単）ぜひ、LinuxのMozillaの特集をやってください。お願いします。

（よしぴーさん）

今月号のWebブラウザ特集はお役に立ちましたでしょうか？　IEファンの担当は、Internet Explorer for Linuxの登場を待ち望んでいます。

ブートマネージャ

最近、ハードディスクを購入したので、Windows Me等を再インストールしました。いろいろなアプリも引っ越しです。これで以前使用していたハードディスクをLinux専用に使う予定ですが、Linuxだけでも2つぐらいインストールするつもりです。デュアルブート（3？）になるので、専用のソフトウェアを購入するほうがいいのでしょうね。

（不良中年ライダーさん）

フリーのブートマネージャとしてLILO、GRUB、GAG、Extended-IPLなどがあります。LILOは、多くのディストリビューションで採用されていて最も入手しやすいと思います。GRUBはLinuxを再構築する際に、再インストールが不要なので、実験用マシンなどには、特にお勧めです。商用ソフトでは、初心者にやさしいユーザーインターフェイスを持つPartitionMagicや、OSのインストール環境を自動作成できるシステムコマンダーなどがあります。

内緒の話

トマトって果物ですか？

（荒木 智さん）

ドキッ。トマトは、緑黄色野菜です。しかも野菜の生産量第一位であるという（トマト投げ祭りの影響でしょうか？）野菜の中の野菜。しかし、野菜の表紙といえば、Linux magazineのお隣さんであるNETWORK MAGAZINEの専売特許ですので、トマトが野菜であることは内緒にしておいてくださいね。

振り向けば……

夏ですねー。熱暴走が気になる季節になってきました。自宅ではクーラーのない部屋で、パンツ一丁でマシンと格闘しています。17インチCRTの発する熱はスゴイので液晶モニタがいいなーと。でも我が家の財務大臣の許可が……。液晶モニタ欲しいよー。

（佐藤孝保さん）

編集部があるフロアでは星の数ほど（ウソ）のコンピュータ本体とモニタが、常時稼動しているといわれており、冬は暖房要らずです。しかし、真夏のこの時期、話は当然、逆！　ビルの空調がストップする夜中は、編集部員泣かせの魔の時間。締め切り間際、徹夜続きで風呂要らずの編集部員からは、良い匂いが……。通勤ラッシュの逆を進む電車の中で、あなたを匂いで振り向かせるその人の手には、発売前のアスキー本があるかもしれません。

静かマシン

静音化実験面白かったです。騒音に悩まされていたので早速実験してみようと思ってマシンを開いたら、よく見るとファンの速度設定用スイッチがパネルの裏側に！　不覚ながら初めて気が付きました。とりあえずは、振動防止のためゴムをかませたりしてビビリ音をなくすことからはじめようと思います。

（梅本 満さん）

パソコンの騒音問題を考えるのページが特におもしろかったです。パソコンで温度やファンの回転数などを検出して、制御するようなプログラムを、ハードと組み合わせて作ったら面白いかもしれませんね。

（楢山 征仁さん）

「パソコンの静音化」とは、そのへんのPC雑誌と同じような企画だなーと思って読んでたらRAMDISKとは! なるほど、Linuxならではの企画ですね。「こんなことができるのか」と感動しました。

（栗山 公秀さん）

8月号の特集のひとつであるパソコンの騒音問題。これは、パソコンの最大の改善点であると思います。常時接続ができる環境になっても、うるさいパソコンをずっと立ち上げるほど人間は我慢強くないと思います。

僕は去年に自作したのですが、Pentium 4が出た現在でも、性能には不満ありません。でも、この騒音に関しては大きく不安があります。ケースファンなどを止めると確かに静かになるものの、今年の夏は乗り切れないし……。やっぱりデスクトップにもクルーソーみたいなCPUは必要でしょうね。

（めそさん）

にゃ〜〜〜。融けかけてます。CPUにも良くないようで……。というわけで、今度熱対策メインの特集をやってくれませんかねぇねぇねぇ（壊れてる）。できれば部屋の温度問題についても。（^_^；

（本多洋平さん）

8月号「パソコンの騒音問題を考える」はご好評いただております。最初、この企画は、パソコンの静かさに加えて涼しさも考えよう！　というものだったのですが、パソコンを冷房代わりにするのは、さすがに難しいということで（ちょっと違う……）、企画変更したという経緯があったのです。

皆さんのご自宅では、静かマシンは実現しましたか？

セキュリティ事情

会社でLinuxサーバを運用していますが、このサーバに対するウイルスを心配しています。というのも先日100台のWindows 2000 Server／クライアント群にウイルスが侵入。トレンドマイクロ社のServerProtect／ウルスバスターによって被害は最小で済みましたが、Linuxサーバだけは問題なのかどうか不明。

Linux用のウイルス対策ソフトってあるのでしょうか？　前述のトレンドマイクロ社のホームページを見てもよくわからないし。皆さんどうしているのでしょう。

（Hiroriさん）

セキュリティが一番心配！　自分なりに用心はしているつもりなのですが……。実際に、踏み台として知らない間に侵入されて、それが元で賠償責任が発生するようなことがあるのでしょうか？　気になっています。

管理者として安心してLinuxを使っていけるような記事をお願いします。

（本田 武さん）

WebサーバやDNSサーバ、システム自身のログが毎回残されるのですが、これらの見方や活用方法などの特集をしていただけるとありがたいです。Windowsでいうイベントビューアのようなフリーソフトがあれば、それらを含めてお願いします。

（伊藤真和さん）

Webサーバのログを解析して、結果をHTML形式で生成してくれるログ解析ソフトとして、Analogやyaalaなどがあります。自動的にログファイルを解析・編集してくれるなんて、ありがたいソフトがあるものです。しかし、サーバを継続して管理していくなら、サーバ管理者の第六感を身に付けるためにも、自分でログファイルを見れるようにしたほうが、やはり安心ですね。

セキュリティに関しては、ご要望を多くいただいておりますので、近々特集を組めるように企画を練っている途中です。もう少々お待ちください。

Itanium搭載機

Itanium搭載機の値段を見てびっくりしました。こんなもん、個人で買えるように本当になる時代がくるのだろうか？　と思ってしまいました。当分はハイエンドサーバのみでの利用でしょうね。ただ、昔の286から386などに変わってきたように、いつかは個人レベルのものになってくることを祈るしかないでしょう。

（村山一馬さん）

Itaniumが、エンドユーザーにまで普及するような時代が来るのでしょうか？　そういう時代には、どんな利用方法が世の中に登場しているのでしょうか？　皆さんはItaniumマシンがどんな用途で使われるようになると思いますか？　人間と見分けのつかない、感情を持ったロボットと話をしたり、遊んだりできるようになるのでしょうか？